

HITACHI

SOFTWARE MANUAL
OPERATION
RPDP/S10V
For Windows®

S10V
Programmable Controller

SOFTWARE MANUAL
OPERATION
RPDP/S10V
For Windows®

S10V
Programmable Controller

First Edition, October 2003, SVE-3-133(A)

All Rights Reserved, Copyright © 2003, Hitachi, Ltd.

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

No part of this publication may be reproduced in any form or by any means without permission in writing from the publisher.

Printed in Japan.

BI-NR-HS<IC-NS> (FL-MW20, AI8.0)



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 the 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 PAGE INTENTIONALLY LEFT BLANK.

PREFACE

This manual describes the procedure for creating real-time programs, which run under the S10V CPMS, on an Microsoft® Windows® 2000 operating system and Microsoft® Windows® XP operating system.

- The table below shows related software manuals.

Manual number	Manual name
SVE-3-201	CPMS GENERAL DESCRIPTION AND MACRO SPECIFICATIONS

- Note that each of the following terms has a special meaning in this manual:

Term	Meaning
Development machine PCs	AZ/R, POC-TX55 or POC-W, HF-W, PADT S10V

Abbreviations

RPDP: Realtime Program Developing Package for S10V

CPMS: Compact Process Monitor System

PCs: An acronym for a Programmable Controllers. A generic name of PLCs such as the S10 α and S10mini Series products.

PLC: An acronym for a Programmable Logic Controller, which is an industrial electronic device that has a built-in program and provides sequence control.
The S10 α and S10mini Series products are also classified as PLCs.

- This manual consists of two parts: PART 1, “GENERAL DESCRIPTION,” and PART 2, “COMMAND REFERENCE.”

PART 1 explains how real-time programs running on the S10V are developed, and also provides general information on commands used in such development. PART 2 serves as a reference on the commands used in developing real-time programs to be run on the S10V; it describes the functions and options of each command.

Appendixes supply notes on the development of real-time programs that run on the S10V, and present error messages and formats in which results of command execution are displayed.

<Trademark>

Microsoft® Windows® 2000 operating system and Microsoft® Windows® XP operating system are registered trademarks of Microsoft Corporation in the United States and/or other countries.

Note for storage capacity calculations:

- Memory capacities and requirements, file size 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

PART 1 GENERAL DESCRIPTION

CHAPTER 1 OVERVIEW.....	1 - 2
1.1 About RPDP.....	1 - 2
1.2 Commands	1 - 4
CHAPTER 2 PROCEDURE FOR PROGRAM DEVELOPMENT.....	1 - 6
2.1 Entire Flow.....	1 - 6
2.2 Site Environment	1 - 9
2.3 Area Management and Area Division in Main Memory.....	1 - 10
2.4 Area Allocation for Tasks	1 - 14
2.5 Area Allocation for IRSUBs	1 - 15
2.6 Loading Programs and Creating Tasks	1 - 16
2.7 Indirect Link Resident Programs.....	1 - 16
2.8 Global (GLB)	1 - 16
2.9 Value (VAL)	1 - 16
2.10 Indirect Link Global Data	1 - 17
2.11 Programming Guide for GLB, VAL and IRSUB.....	1 - 17
2.12 Constraints on CPMS Program Creation	1 - 26
2.13 Managing Names	1 - 32
CHAPTER 3 INSTALLATION AND RPDP EXECUTION ENVIRONMENT.....	1 - 33
3.1 Installation.....	1 - 33
3.2 RPDP Execution Environment	1 - 34
3.3 RPDP Command Storage Directory.....	1 - 35
3.3.1 Environment variable setup example	1 - 35
3.3.2 Setting an environment variable from the command prompt.....	1 - 35
3.4 Registering an RPDP User Account	1 - 36
3.4.1 Registering a new account.....	1 - 36
3.4.2 Adding the RPDPS10Vusers group as a group to which an existing account belongs.....	1 - 37
CHAPTER 4 COMPILER.....	1 - 38
4.1 Details of C Compiler Options	1 - 38
4.2 Compiling Precautions.....	1 - 40
4.2.1 When compiling with shc commands	1 - 40
4.3 Differences between mcc68k and shc	1 - 42
4.3.1 Command line options	1 - 42
4.3.2 Language specification differences	1 - 44
4.3.3 Function call rules	1 - 44
CHAPTER 5 PROGRAMMING COMMANDS	1 - 45
5.1 Notes on Programming Commands	1 - 45

CHAPTER 6	GENERATION	1 - 46
6.1	Overview.....	1 - 46
6.1.1	Purpose of system generation.....	1 - 46
6.1.2	Features of system generation	1 - 46
6.1.3	Site management method for a plurality of PCs units.....	1 - 48
6.2	System Generation Commands and Flow of System Construction	1 - 49
6.3	PCs System Definition Information.....	1 - 53
6.3.1	User setting definition information	1 - 53
6.3.2	Contents of PCs system definition information	1 - 56
6.3.3	Contents of network definition information.....	1 - 57
6.3.4	Site definition information	1 - 58
6.4	PCs System Definition Information Display	1 - 59
6.5	PCs System Environment Duplication	1 - 59
6.5.1	Duplication unit	1 - 59
6.5.2	Duplication range	1 - 60
6.6	PCs System Environment Deletion.....	1 - 61
CHAPTER 7	ALLOCATOR.....	1 - 62
7.1	Allocating and Deallocating Split Areas	1 - 62
7.1.1	Necessity for split areas	1 - 62
7.1.2	Allocating split areas.....	1 - 63
7.1.3	Deallocating split areas	1 - 66
7.1.4	Assigning names to GLB and VAL	1 - 66
7.2	Value (VAL) Registration and Deletion.....	1 - 67
CHAPTER 8	LOADER.....	1 - 68
8.1	Linking and Loading.....	1 - 68
8.2	Loader Operating Environment	1 - 69
8.3	Search Path of Libraries.....	1 - 73
8.4	Notes on Linking and Loading	1 - 73
CHAPTER 9	BUILDER.....	1 - 74
9.1	Registering and Deleting a Task.....	1 - 74
9.1.1	About the task	1 - 74
9.1.2	Registering a task	1 - 74
9.1.3	Deleting a task.....	1 - 75
9.2	Registering and Deleting a Resident Subprogram	1 - 76
9.2.1	About the indirect link subprogram (IRSUB).....	1 - 76
9.2.2	Registering an indirect link subprogram (IRSUB).....	1 - 76
9.2.3	Deleting an indirect link subprogram (IRSUB)	1 - 77
9.3	Registering and Deleting a Built-in Subroutine.....	1 - 78
9.3.1	About the built-in routine.....	1 - 78
9.3.2	Registering a built-in subroutine.....	1 - 78
9.3.3	Deleting a built-in subroutine	1 - 79
CHAPTER 10	MAP.....	1 - 80
10.1	Purpose of Displaying Allocator Management Table Information.....	1 - 80
10.2	svmap Command Options and Displayed Information.....	1 - 80

10.2.1	Map information targeted for output	1 - 80
10.2.2	Description of map information output	1 - 80
10.2.3	Map information output forms	1 - 80
10.3	Logical Address Specification and Information Display by the svadm Command	1 - 82
CHAPTER 11 STARTUP		
11.1	Overview	1 - 83
11.2	CMU State Transitions	1 - 83
11.2.1	Startup procedure	1 - 83
11.2.2	CMU Control Procedure	1 - 85
CHAPTER 12 svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT FUNCTIONS		
12.1	Overview	1 - 87
12.2	PCs Status and Subcommand Availability	1 - 88
12.3	Basic Functions	1 - 89
12.4	Other Functions	1 - 93
12.5	Debug Support Commands	1 - 94
12.5.1	svelog command	1 - 94
12.5.2	svdhp command	1 - 95
12.5.3	svepunow command	1 - 96
12.5.4	svtimex command	1 - 97

PART 2 COMMAND REFERENCE

CHAPTER 1	SYSTEM GENERATION	2 - 2
CHAPTER 2	ALLOCATOR	2 - 7
CHAPTER 3	LOADER	2 - 16
CHAPTER 4	BUILDER	2 - 34
CHAPTER 5	MAP	2 - 44
CHAPTER 6	STARTUP	2 - 51
CHAPTER 7	svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS	2 - 53

APPENDIXES

APPENDIX A	NAMES USABLE IN PROGRAMS	A - 2
APPENDIX B	LIBRARIES	A - 5

APPENDIX C	SITE MANAGEMENT FILES	A - 8
APPENDIX D	ERROR MESSAGES	A - 12
APPENDIX E	NOTE ON USE OF RPDP	A - 46
APPENDIX F	DISPLAY FORMAT OF svmap	A - 48
APPENDIX G	DISPLAY FORMATS OF md AND sd OF debughr (ONLINE DEBUGGER)	A - 63
APPENDIX H	LIST OF STACK SIZES FOR LIBRARY USE	A - 67

FIGURES

Figure 1-1	Configuration of a System in which the Development Tool is Used	1 - 2
Figure 1-2	Program Development Flowchart.....	1 - 7
Figure 1-3	PCs Site Directory Structure.....	1 - 9
Figure 1-4	Logical Space Managed by CPMS	1 - 10
Figure 1-5	PCs Physical Memory Map	1 - 12
Figure 1-6	Task Arrangement in Logical Space.....	1 - 14
Figure 1-7	Task Arrangement in Logical Space (Multitask).....	1 - 14
Figure 1-8	IRSUB Arrangement in Logical Space.....	1 - 15
Figure 1-9	IRSUB Arrangement (Multi-entries) in Logical Space	1 - 15
Figure 1-10	Write Enable or Disable	1 - 27
Figure 1-11	Comparison of Data Sizes	1 - 30
Figure 1-12	Sample Declaration of a Structure with Alignment Considered	1 - 31
Figure 1-13	Example of Explicitly Declared Free Areas	1 - 31
Figure 1-14	Sample Declaration with Structure Size Considered.....	1 - 31
Figure 1-15	PCs System Overall Configuration.....	1 - 47
Figure 1-16	Site Directory Structure	1 - 48
Figure 1-17	Procedure for Newly Constructing a Controller System	1 - 50
Figure 1-18	Procedure for Modifying a PCs System.....	1 - 50
Figure 1-19	Procedure for copying a site within the same development machine.....	1 - 51
Figure 1-20	RPDP-Related Directory Structure and Site Construction Environment Definition File Configuration	1 - 56
Figure 1-21	Scope of Copying Provided by the Site Duplication Function	1 - 60
Figure 1-22	Sample Layout of Split Areas.....	1 - 63
Figure 1-23	Creating a Load Module and Backup File	1 - 68
Figure 1-24	Load Module Structure	1 - 70
Figure 1-25	Loading processing	1 - 71
Figure 2-1	Function Call and Stack Use Amount	2 - 22
Figure 2-2	Format of the Display Shown by svcomp for a Program or Subprogram	2 - 32
Figure 2-3	Format of the Display Shown by svcomp for GLB	2 - 33
Figure 2-4	Memory Access Range	2 - 70
Figure 2-5	Operation Procedure for Dynamic Display.....	2 - 70
Figure 2-6	Memory Access Range	2 - 79
Figure 2-7	ADT Setup Areas.....	2 - 98

TABLES

Table 1-1	RPDP Commands.....	1 - 4
Table 1-2	Uses of Various Logical Spaces.....	1 - 10
Table 1-3	Logical Space Addresses and Sizes.....	1 - 11
Table 1-4	Assigning names to GLB and VAL.....	1 - 17
Table 1-5	How to Use GLB and VAL.....	1 - 17
Table 1-6	IRSUB Usage	1 - 24
Table 1-7	List of RPDP Execution Environment Variables.....	1 - 34
Table 1-8	Setup Environment Variables.....	1 - 39
Table 1-9	Options for Floating-point Number Handling.....	1 - 40
Table 1-10	Floating-point Number Handling and Corresponding Standard Libraries	1 - 40
Table 1-11	Comparison between mcc68k and shc Command Line Options.....	1 - 42
Table 1-12	List of shc Options	1 - 43
Table 1-13	Language Specification Comparison.....	1 - 44
Table 1-14	System Generation Commands	1 - 49
Table 1-15	Contents of Site Definition Information.....	1 - 53
Table 1-16	User-Defined Site Configuration Definition Information Setup Items	1 - 55
Table 1-17	Site Construction Definition Information about Memory.....	1 - 58
Table 1-18	Relationship between Split Area Use and GAREA Selection	1 - 63
Table 1-19	Conditions for Load Module	1 - 69
Table 1-20	External Reference Combinations.....	1 - 72
Table 1-21	Combinations of Output Information and Selectable Output Forms	1 - 81
Table 2-1	Combinations of svdfa Options.....	2 - 8
Table 2-2	Relationship between the Value Specified for stype and the Alignment Count	2 - 11
Table 2-3	Combinations of svdfs Options.....	2 - 11
Table 2-4	Stack Size Calculation Examples.....	2 - 22
Table 2-5	Relationship between Option Combinations and Outputs	2 - 46
Table 2-6	Functions of svdebug	2 - 55
Table 2-7	Operation on Reception of a Signal	2 - 56
Table 2-8	Task States	2 - 61
Table 2-9	Status Bit Stings	2 - 61
Table 2-10	Explanation of the id, t, and cyct Parameters	2 - 65
Table 2-11	Relationships between Specifiable Values and Options	2 - 69
Table 2-12	Display Formats Depending on the Combination of Options	2 - 69
Table 2-13	Management States of Resources.....	2 - 105
Table B-1	Conditions for Specifying Library Names.....	A - 5
Table D-1	Operation in Reception of a Signal	A - 22
Table D-2	Error Messages	A - 22
Table F-1	Real-Time Source Management Status	A - 50

PART 1 GENERAL DESCRIPTION

CHAPTER 1 OVERVIEW

1.1 About RPDP

The real-time program developing package (S10V RPDP) is a tool for developing programs to be run under the real-time operating system (CPMS) for the S10V. This tool runs on an Windows® 2000 or Windows® XP machine (hereafter collectively called development machines). Figure 1-1 shows the configuration of a system in which the development tool is used.

- S10V RPDP: Realtime Program Developing Package for S10V
- CPMS: Compact Process Monitor System

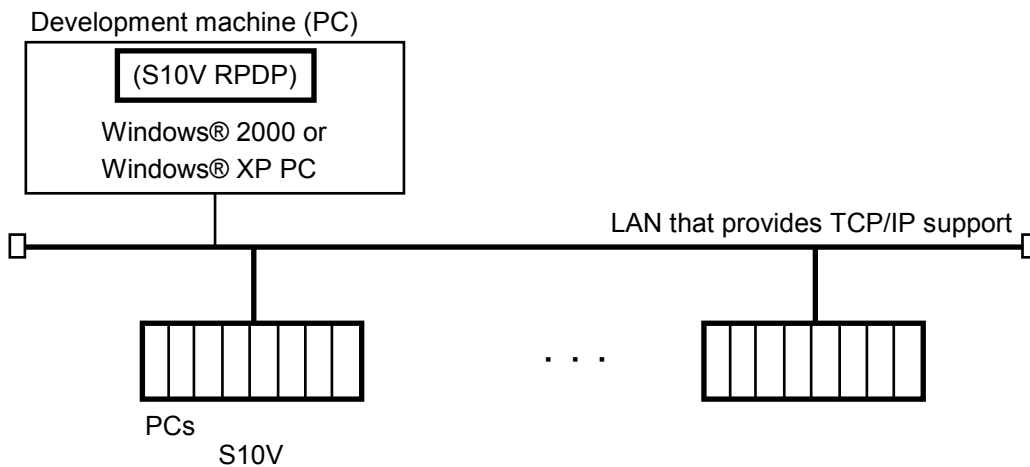
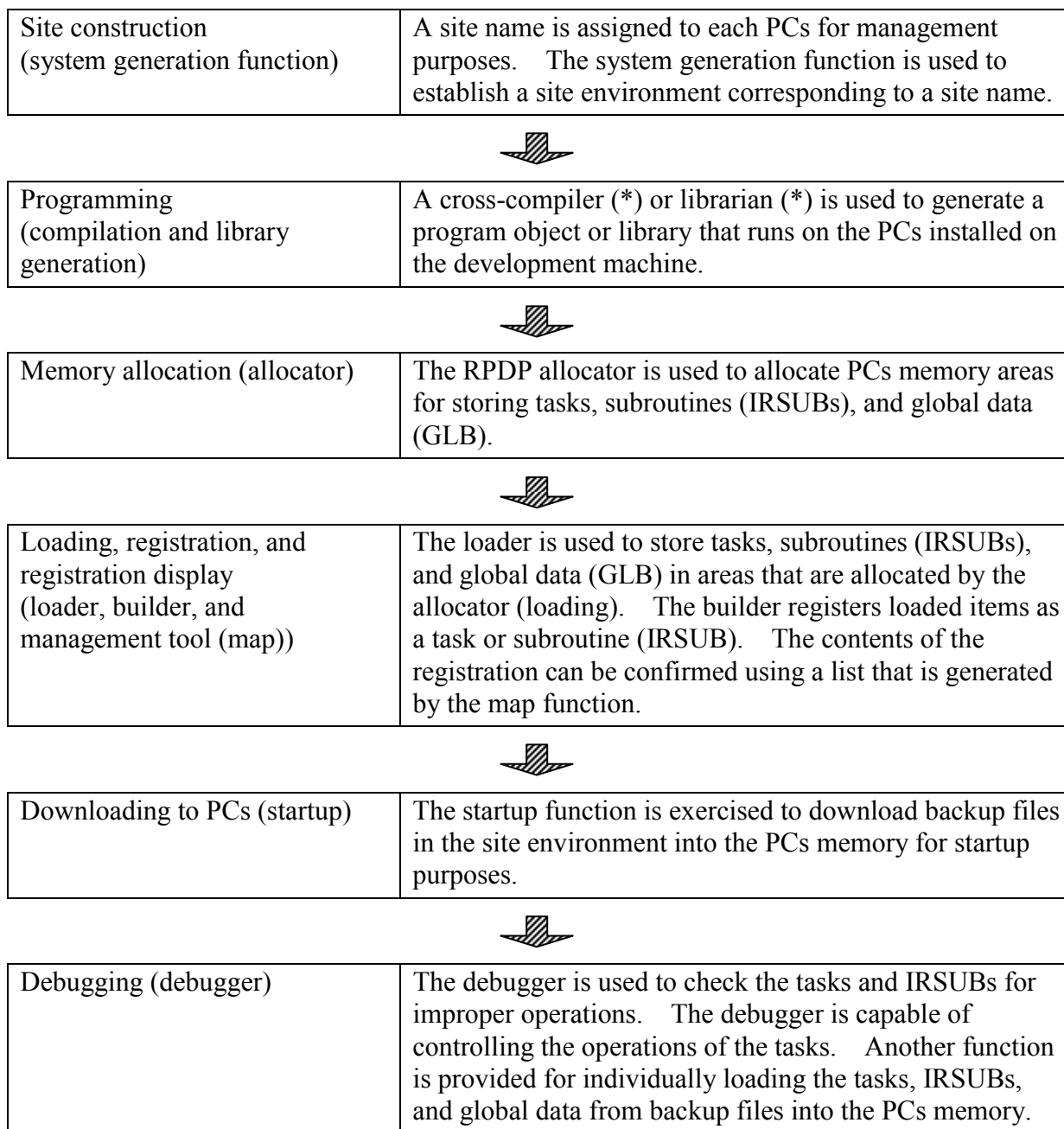


Figure 1-1 Configuration of a System in which the Development Tool is Used

In development of real-time programs that run under CPMS, use the dedicated development system S10V RPDP (simply called RPDP hereafter), enabling programs running under CPMS to use attributes and functions provided for high-speed real-time processing.

The RPDP program development sequence and supported functions are described below:



(*) Hitachi Microcomputer Development Environment System Super Engine C/C++ Compiler Package Version 7 or later is used as the cross-compiler/librarian.

1. OVERVIEW

1.2 Commands

Table 1-1 lists all RPDP commands.

Table 1-1 RPDP Commands (1/2)

Classification	Command name	Function	Reference page		
System generation	svgen	Sets up an environment to construct a site.	2-2		
	svconf	Registers a set of information for a system definition.	2-3		
		Specifies the IP address and memory size.			
	svshconf	Displays a system definition.	2-4		
	svsitecp	Copies a site.	2-5		
svsitedel	Deletes a site.	2-6			
Compiler	shc	Cross compiler (part of the compiler package)	(*)		
Programming commands	optlnk	Librarian (part of the compiler package)	(*)		
	optlnk	Linker (part of the compiler package)	(*)		
Allocator	svdfa	Provides split area allocation and backup file generation.	2-7		
	svdla	Provides split area deletion and backup file deletion.	2-9		
	svdfs	Provides secondary partition area allocation	2-10		
	svdls	Provide secondary partition area deletion.	2-13		
	svdfv	Registers a VAL.	2-14		
	svdlv	Registers or deletes a VAL.	2-15		
Loader	svload	Stores a resource in a backup file and registers it as management information.	2-16		
	svdload	Deletes a resource from the management information.	2-29		
	svcomp	Provides comparison with a registered resource.	2-30		
Builder	svctask	Creates a task.	2-34		
	svdtask	Deletes a task.	2-36		
	svbuild	Creates an indirect link subprogram.	2-37, 2-38		
		Creates a built-in subroutine.			
	svdbuild	Deletes an indirect link subprogram.	2-40, 2-41		
		Deletes a built-in subroutine.			
svirglb	Registers or deletes an IRGLB.	2-42			
Online debugger	svdebug	Online debugger		2-53	
		Start or stop of a task.	qu	Requests start of a task.	2-57
			ab	Inhibits task start.	2-58
			re	Cancel inhibition of task start.	2-59
			ta	Displays the task status.	2-60
			su	Suppresses or suspends task execution.	2-62
			rs	Cancel suppression of task execution.	2-63
			tm	Starts a task cyclically.	2-64
	ct	Cancel cyclic task start.	2-66		

(*) Click [Start], point to [Programs], and start [Hitachi Embedded Workshop2 Online Manuals [SuperH] - Japanese (Ver-Rev)], and then see under “C/C++ Compiler Operating Procedures” and “Optimization Linkage Editor Operating Procedures” in the SuperH RISC Engine C/C++ Compiler/Assembler/Optimization Linkage Editor User Manual.

Table 1-1 RPD Commands (2/2)

Classification	Command name	Function		Reference page					
Online debugger	svdebug	Start or stop of a task.	sht	Displays task cyclic starts.	2-67				
			si	Effects stack initialization.	2-113				
			sp	Displays the amount of stack use.	2-114				
	Memory print or patch	svdebug		md	Displays/changes the memory contents in accordance with a specified address.	2-68			
				sd	Displays/changes the memory contents in accordance with a specified name.	2-71			
				bs	Sets data in specified bit positions.	2-74			
				bg	Displays data existing in specified bit positions.	2-76			
				mcp	Copies the memory contents.	2-78			
				mmv	Moves the memory contents.	2-80			
				mf	Sets a pattern value in the memory.	2-82			
				Breakpoint	svdebug		br	Sets/displays breakpoints.	2-86
							rb	Deletes breakpoints.	2-91
	rd	Displays registers.	2-92						
	rr	Changes the contents of registers.	2-95						
	go	Resumes execution from a breakpoint.	2-96						
	System error display or reset	svdebug		el	Displays error log.	2-84			
				ss	Displays the system status.	2-84			
	Current time setting or display	svdebug		st	Sets the current time.	2-85			
				gt	Displays the current time.	2-100			
	Uploading/downloading/compare	svdebug		ld	Downloads resources individually.	2-101			
				sv	Backs up resources individually.	2-108			
				cm	Compares the contents of a backup file and PCs memory.	2-110			
	Enables/disables the DHP recording function.	svdebug		dr	Enables the DHP recording function.	2-111			
				ds	Disables the DHP recording function.	2-111			
	ADT	svdebug		as	Sets/displays the ADT.	2-97			
				ac	Resets the ADT.	2-99			
				svdhp	Displays the DHP.	2-112			
				svadm	Displays a resource name for an address.	2-47			
				ps	Starts displaying a debug statement.	2-116			
				pe	Finishes displaying a debug statement.	2-116			
				ver	Displays the version of the CMU.	2-117			
	Others	svdebug		help	Displays the subcommands.	2-118			
				q	Terminates the deburrer.	2-119			
!				Executes a command on the development machine at the time of svdebug execution.	2-119				
Management tool	svmap	Displays map information.		2-44					
	svadm	Displays information corresponding to an address.		2-47					
	svsitectl	Controls and displays the site status.		2-50					
Start-up	svrpl	Performs remote loading.		2-51					
	svcpuctl	Controls the statuses of the remote.		2-52					
Activity management	svcpunow	Displays the CPU load ratio.		2-124					
	svtimex	Displays the task activity ratio.		2-125					
Display of error log and DHP traces	svelog	Outputs error log information.		2-120					
	svdhp	Displays DHP trace information.		2-112, 2-122					

2. PROCEDURE FOR PROGRAM DEVELOPMENT

CHAPTER 2 PROCEDURE FOR PROGRAM DEVELOPMENT

2.1 Entire Flow

Figure 1-2 shows the entire flow of the program development procedure.

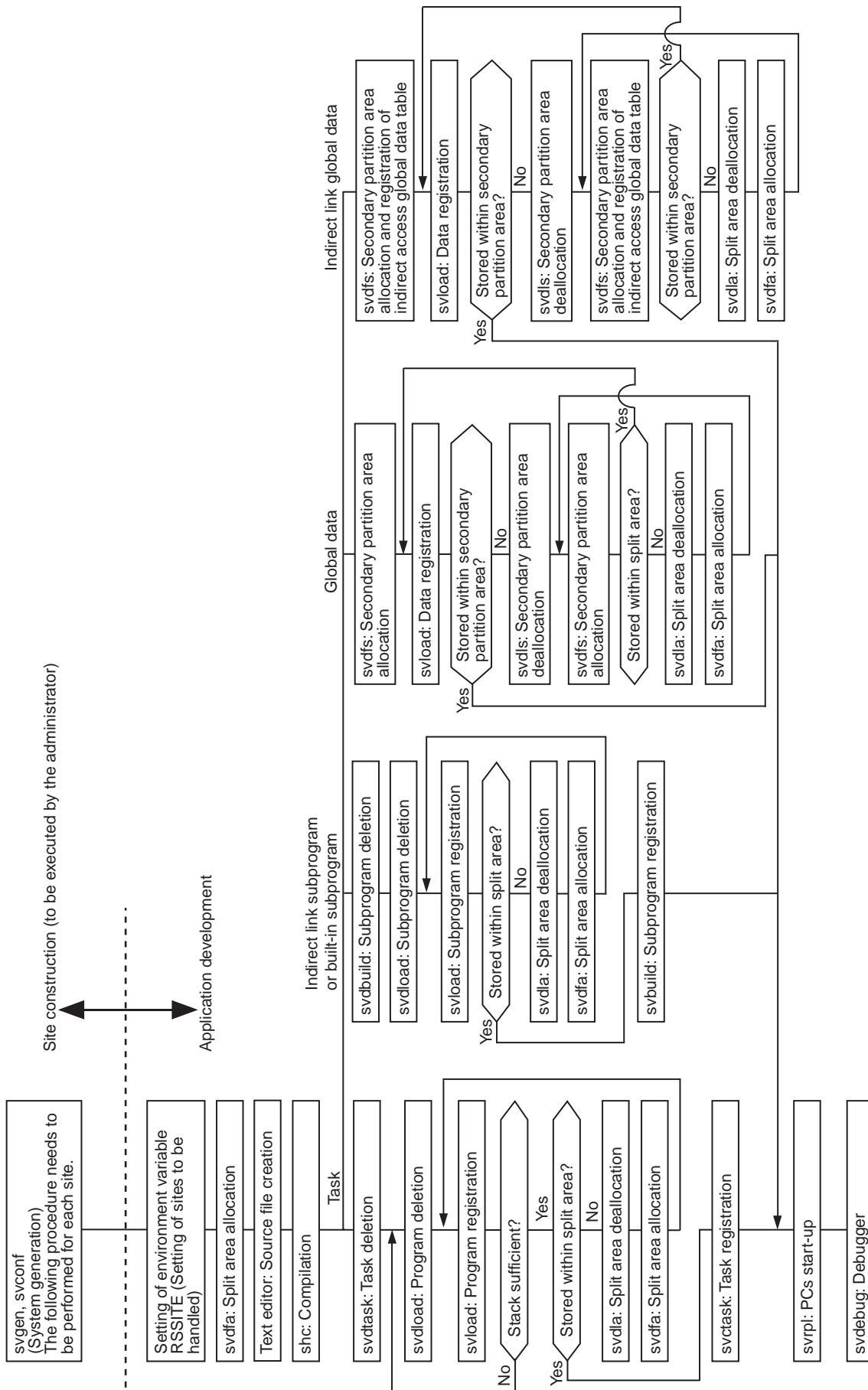


Figure 1-2 Program Development Flowchart (1/2) (from Site Construction to Program Development)

2. PROCEDURE FOR PROGRAM DEVELOPMENT

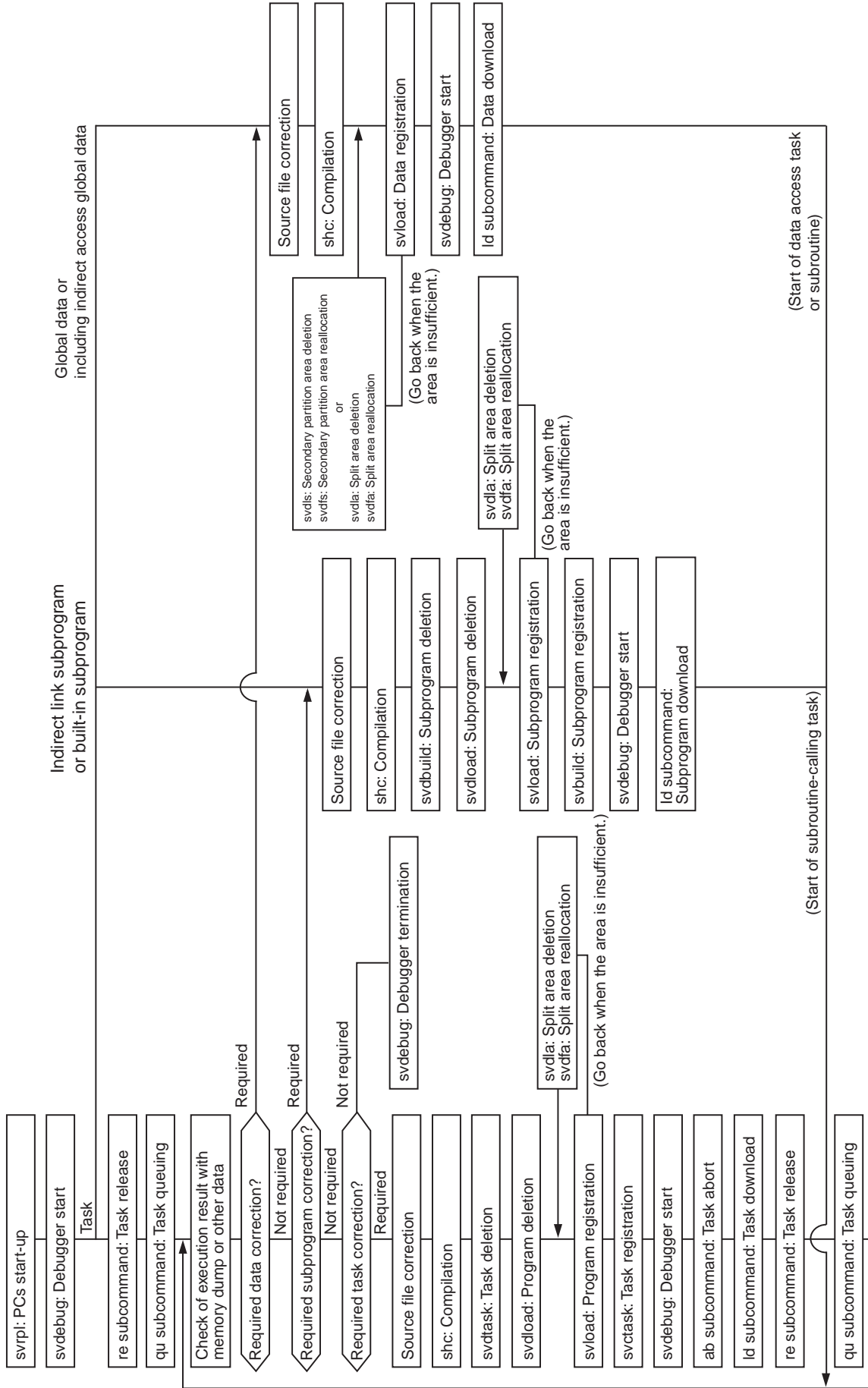


Figure 1-2 Program Development Flowchart (2/2) (from Site Construction to Program Development)

2.2 Site Environment

RPDP assigns a site name to each PCs and manages, on an individual PCs basis, the tasks, subprograms, and global data running on CPMS.

The site name is unique to each PCs. It is to be determined by the user at the time of system generation.

In RPDP system generation, a site directory having a site name is created for each site to contain management files possessed by a site. The management files include a backup file, which is the initial value data file for the PCs internal memory, and files for managing tasks, subroutines, and global data stored in the backup file.

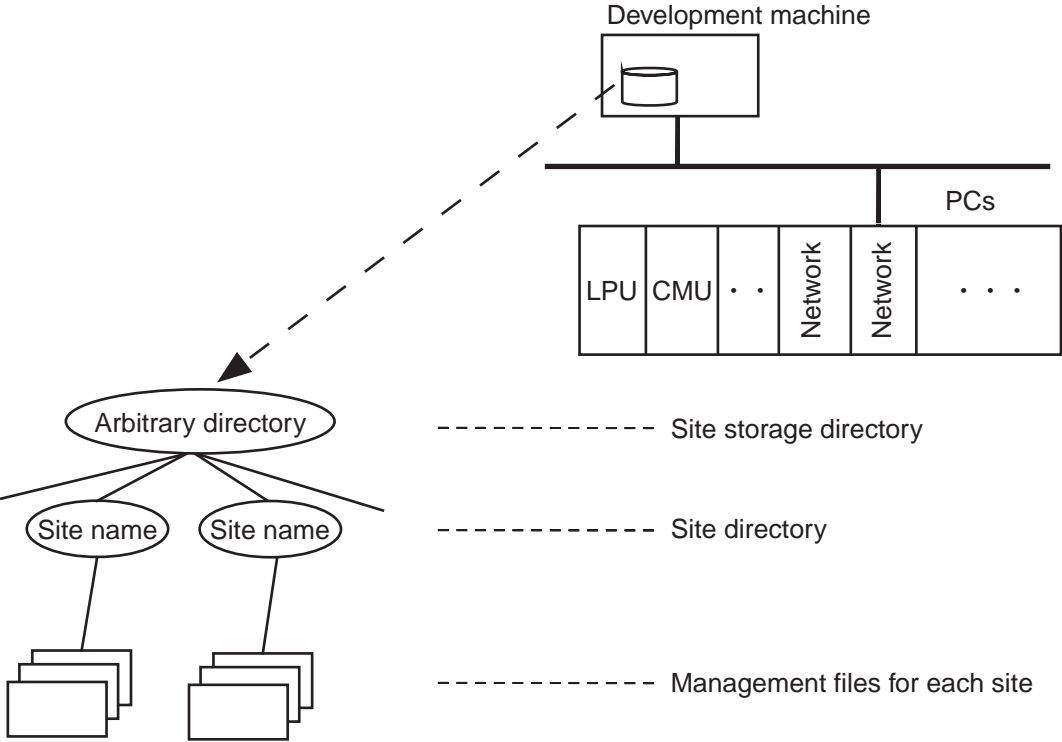


Figure 1-3 PCs Site Directory Structure

2. PROCEDURE FOR PROGRAM DEVELOPMENT

2.3 Area Management and Area Division in Main Memory

On the development machine, RPDP performs area management for main memory in the PCs. The purpose of area management is to allocate programs, subprograms, and data in main memory in an efficient manner, without their duplicated allocation. The memory space subject to area management by RPDP is the logical space managed by CPMS on the physical memory and PCs.

During system generation, physical memory mapping starts with the beginning of the logical space so as to provide memory sizes that are defined for various uses in accordance with GAREA. Figure 1-4 shows the logical space managed by CPMS. Table 1-2 the uses of the logical space.

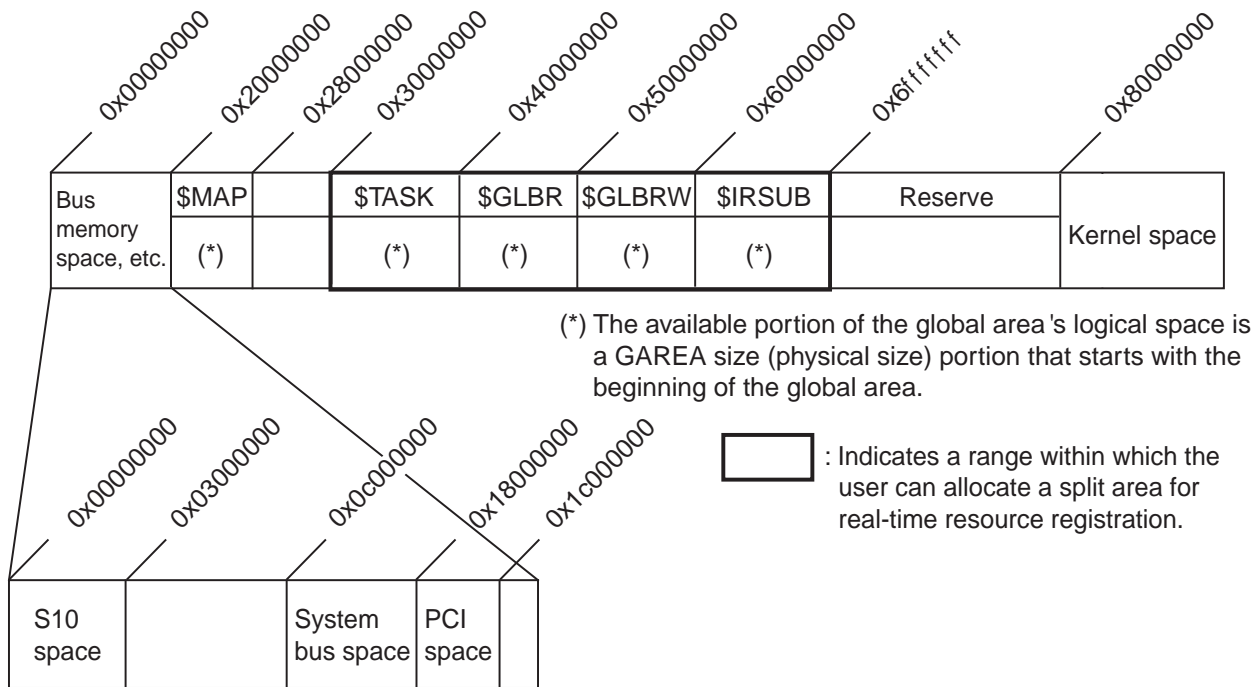


Figure 1-4 Logical Space Managed by CPMS

Table 1-2 Uses of Various Logical Spaces

Logical space's global area name	Use
\$TASK	This area stores tasks (programs).
\$GLBR	This area stores read-only GLB.
\$GLBRW	This area stores read/write GLB.
\$IRSUB	This area stores subprograms.
\$MAP	Stores the following tables, which are managed by RPDP: <ul style="list-style-type: none"> • IRSUB indirect link table (IRSUBT) • Task control block (TCB) • IRGLB indirect link table (IRGLBT) • Built-in subroutine table (USLCB) Stores the management information in the PCs memory as well. Stores the map information (allocator management table) and package version information.

Table 1-3 shows the addresses and sizes of the logical spaces managed by CPMS as indicated in Figure 1-4.

Table 1-3 Logical Space Addresses and Sizes

S10V			
Logical space's global area name (use)	Address	Size	SH4 virtual address space area name
Reserve	0x00000000-0x0000ffff	64 KB	P0 area (U0 area)
S10 space	0x00010000-0x01ffffff	32 MB-64 KB	
Reserve	0x02000000-0x0ffffff	8 MB	
Bus memory space	0x0c000000-0x17ffffff	192 MB	
PCI space	0x18000000-0x1bffffff	64 MB	
Reserve	0x1c000000-0x1ffffff	64 MB	
\$MAP	0x20000000-0x27ffffff	128 MB	
CPMS area	0x28000000-0x2ffffff	128 MB	
\$TASK	0x30000000-0x3ffffff	256 MB	
\$GLBR	0x40000000-0x4ffffff	256 MB	
\$GLBRW	0x50000000-0x5ffffff	256 MB	
\$IRSUB	0x60000000-0x6ffffff	256 MB	
Reserve	0x70000000-0x7ffffff	176 MB	
Kernel space	0x80000000-0x9ffffff	512 MB	P1 area
	0xa0000000-0xbffffff	512 MB	P2 area
Reserve	0xc0000000-0xdffffff	512 MB	P3 area
	0xe0000000-0xfffffff	512 MB	P4 area

2. PROCEDURE FOR PROGRAM DEVELOPMENT

Figure 1-5 shows the details of the physical memory map.

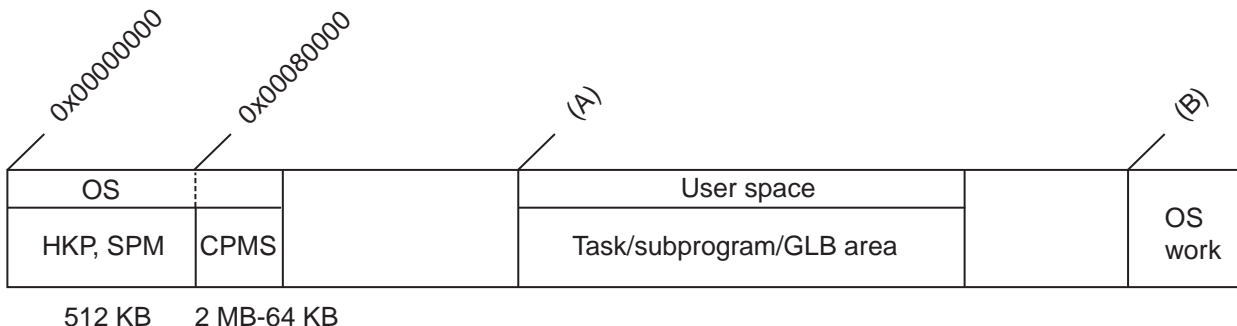


Figure 1-5 PCs Physical Memory Map

SPM: CPMS edition data.

HKP: A space where the hardware KROM Program copied from the ROM runs.

CPMS: An OS program that is copied from the ROM.

Task/subprogram/GLB area:

An area into which tasks, subprograms, and GLB are downloaded.

The user specifies the size of each GAREA at the time of system generation.

The map information, TCB, IRSUBT, IRGLBT, and USLCB are also downloaded into this area.

OS work: A buffer area for use by the OS. It contains a DHP area, error log area, and network buffer.

- Details of task/subprogram/GLB area
 The task/subprogram/GLB area is detailed below:
 Construct a site by specifying the size of each area (GAREA) at the time of system generation.
 (The \$MAP size is fixed at approximately 0.7 MB.)

Task/subprogram/GLB area				
\$MAP	\$TASK	\$GLBR	\$GLBRW	\$IRSUB

\$MAP: Area for storing the management information in the PCs memory.
 \$TASK: Area for storing tasks (programs).
 \$GLBR: Area for storing read-only GLB.
 \$GLBRW: Area for storing read/write GLB.
 \$IRSUB: Area for storing subprograms.

When generating a task or subprogram that runs on CPMS or the data for use by the task or subprogram, first allocate a split area (area) within a global area (GAREA) for task/subprogram/data storage. For GLB (global data), further divide the split area into secondary partition areas (sarea). Tasks and subprograms specify the secondary partition area name in order to access data.

(1) Split area

A split area (area) is to be allocated with svdfa and deallocated with svdla. A plurality of split areas can be allocated within a global area (GAREA) that is defined at the time of system generation.

When a split area is allocated, a backup file adequate for the size of the allocated area will be generated.

(2) Secondary partition area

A plurality of resources can be arranged within a split area allocated by svdfa. Tasks and subprograms are to be arranged within a split area with svload and deallocated with svdload. A GLB secondary partition area (sarea) is to be allocated with svdfs and deallocated with svdls.

To deallocate a split area (area), use svdla.

2. PROCEDURE FOR PROGRAM DEVELOPMENT

2.4 Area Allocation for Tasks

All tasks that run under CPMS are allocated a single logical space. The tasks are to be stored in a split area allocated to \$TASK. A plurality of tasks can be stored in a single split area.

Place text on a page boundary (4 KB boundary). Place data/bss and stack on an 8-byte boundary.

Further, ensure that the text/data and stack/bss for the same task and the OS work are placed on different pages.

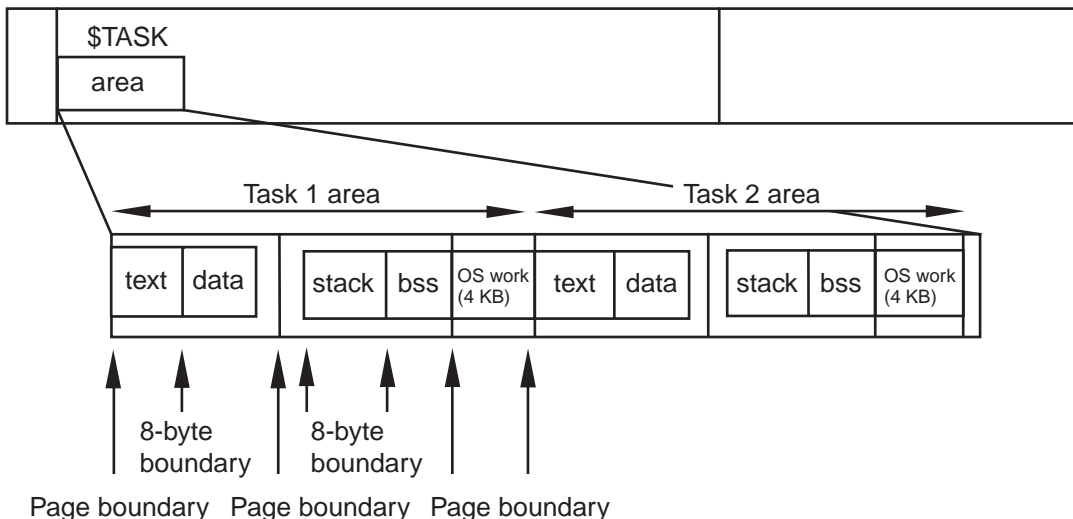


Figure 1-6 Task Arrangement in Logical Space

- Stack arrangement for a multitask
 As regards a multitask, place a stack and bss on different pages. The OS work and stack are to be allocated as needed for all multitasks at the time of loading. The use portion must be specified by the user at the time of task generation. (The figure below relates to a situation where Tasks 1 and 2 are multitasks.)

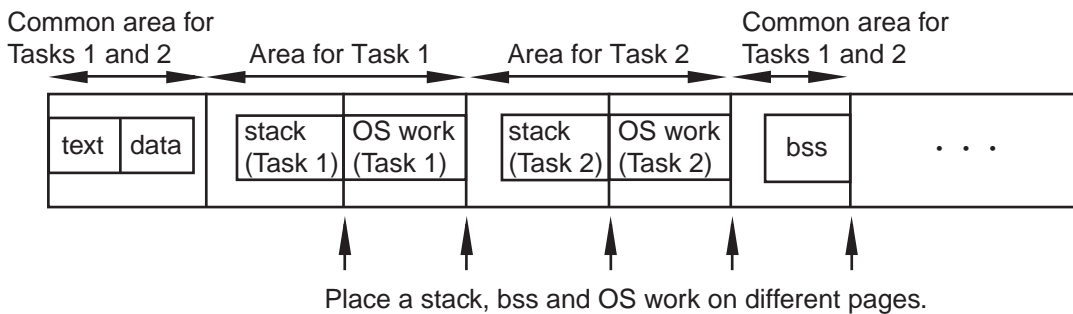


Figure 1-7 Task Arrangement in Logical Space (Multitask)

A multitask is a set of n tasks that are generated for one program with a view toward reducing the memory size required for the program. A multitask is implemented by sharing the loaded program's text, data, and bss divisions and placing the stack division in various areas for all tasks.

It is therefore well to remember that when a task within a multitask writes information into the bss division, the written information is also conveyed to the other tasks within the multitask. As a result, the operation of a task that operates while expecting the initial bss state cannot be guaranteed. As regards a multitask, therefore, do not write data into the bss division.

2.5 Area Allocation for IRSUBs

All IRSUBs running on CPMS are arranged within a single logical space. The IRSUBs are stored in a split area allocated to \$IRSUB. A plurality of IRSUBs can be stored in a single split area.

Place text on a 32-byte boundary and data on an 8-byte boundary.

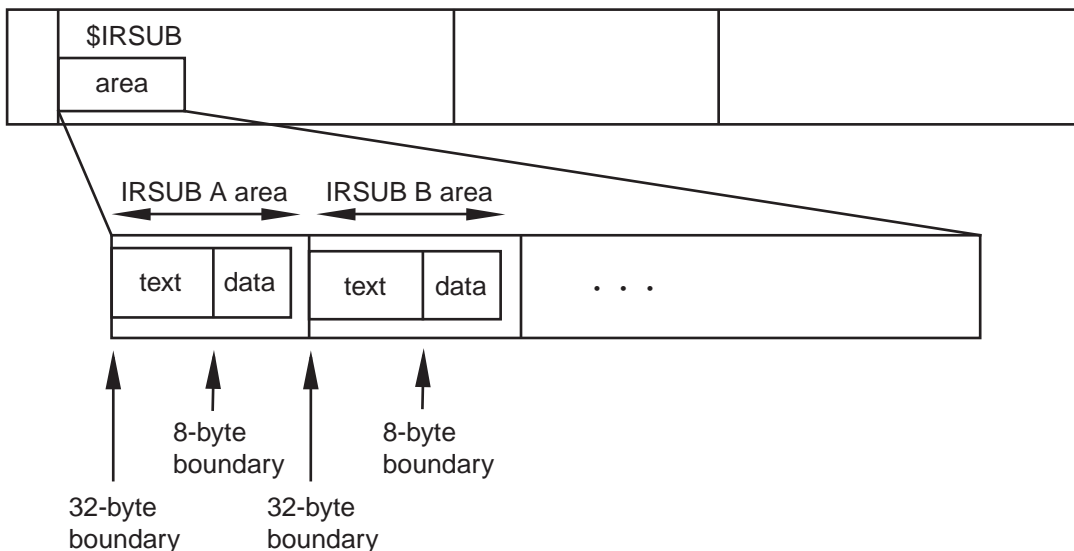


Figure 1-8 IRSUB Arrangement in Logical Space

● Multi-entry arrangement

The same text/data arrangement applies to multi-entry IRSUBs. Multi-entry names and relative entry addresses are retained in a management file of RPDP for management purposes. Figure 1-9 shows how multi-entry IRSUBs (entries named “A,” “B,” and “C”) are positioned.

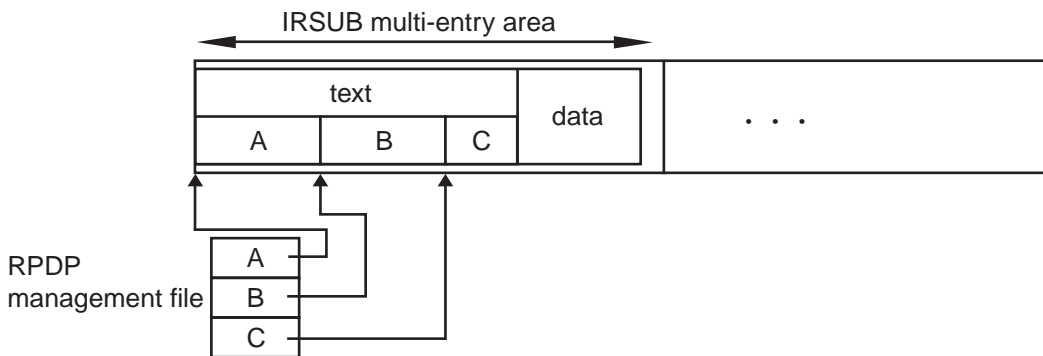


Figure 1-9 IRSUB Arrangement (Multi-entries) in Logical Space

2. PROCEDURE FOR PROGRAM DEVELOPMENT

2.6 Loading Programs and Creating Tasks

The loader (svload) loads load module programs and data in areas or sareas. Loader creates executable modules while referencing information on CPMS resources such as global data and setting the referenced information in load modules. The created executable modules are stored in backup files on the disk in the development machine.

Now, the executable modules are loaded as programs. Register them as tasks using the builder (svctask). svctask sets task attributes in a CPMS-managed table called the task control block (TCB).

2.7 Indirect Link Resident Programs

Suppose that a task consists of several subprograms. Of these, subprograms included in the task are called internal subprograms (ISUBs). In contrast, resident subprograms (RSUBs) are subprograms that are outside the task and resides in main memory so that they are shared by other tasks. RPDP supports indirect link RSUBs (IRSUBs). The IRSUBs are provided with a management table linked to tasks.

The IRSUB main body can easily be changed without changing the tasks linked to the management table.

The loader (svload) updates IRSUBs, while the builder (svbuild) updates the indirect link management table.

2.8 Global (GLB)

CPMS supports GLB so that main memory can be shared by tasks. An area for GLB is allocated by the allocator in advance in the GLB space in the logical space dedicated to CPMS tasks. A name is assigned to the area so that multiple tasks and subroutines can share it. The area is divided into areas by svdfa, each of which is further divided into sareas by svdfs.

2.9 Value (VAL)

The user can register constants to be shared by programs as external names. These external names are called VAL data. VAL can be registered by svdfv and deleted by svdlv. VAL is set by the loader when load modules are loaded in backup files. Therefore, VAL must be set before tasks and subprograms that reference VAL are loaded.

2.10 Indirect Link Global Data

RPDP supports indirect link global data. A management table linked to global data is set for indirect link global data. Updating the linked table facilitates changing global data. The allocator (svdfs) and loader (svload) the global main body is updated. Area definition is achieved by svdfs. Initial value loading is performed by svload. The builder (svirglb) updates the management table for indirect linking.

2.11 Programming Guide for GLB, VAL and IRSUB

This section describes the coding method and linking method for GLB/VAL/IRSUB, which is used by programs and subprograms.

(1) Assigning names to GLB and VAL

Table 1-4 Assigning names to GLB and VAL

Item	Specifications
Maximum number of characters	14 characters (except for _g and _v)
Naming rules	Single-byte alphanumeric characters and _ (underscores). However, the first character must be an alphabetical character. The last character, which indicates the attribute, must be in the following form: GLB: _g VAL: _v
Uniqueness	Identical names cannot be used.

(2) Using GLB, and VAL names

Table 1-2 shows how to use GLB and VAL

Table 1-5 How to Use GLB and VAL

No.	Purpose	C language
1	To declare GLB (on the referencing side)	<code>extern long name_g[size];</code> Explanation name: Global data name (name of an SAREA in the GLB area) size: Global data size
2	To reference GLB (Specification of a secondary partition area name)	<code>extern long name_g[size];</code> <code>main() {</code> <code> long i;</code> <code> i = name_g[index];</code> <code>}</code> Explanation name: Global data name (At the beginning of this coding, add the declaration at No. 1.) Example: <code>extern long name_g[25];</code>
3	To declare GLB (on the referenced side)	There is no need to declare GLB. Set initial values as shown No.4.
4	To set initial values in GLB	<code>long name_g[size] = {1,2,3,.....};</code> Explanation name: Global data name size: Global data size
5	To reference VAL values	<code>extern long name_v;</code> <code>long y = (long) &name_v;</code> <code>main() {</code> <code> long x;</code> <code> x = y;</code> <code>}</code> Explanation name: VAL name

2. PROCEDURE FOR PROGRAM DEVELOPMENT

(3) Notes on GLB data reference

When GLB data is referenced at program creation, data handling varies depending on whether the GLB to be referenced has a defined initial value or not on the same program. Therefore, create a program to reference the GLB data taking the following points into consideration.

- ① When the GLB to be referenced has not been defined in the same program
The above condition refers to the case where the GLB to be referenced in the source program that becomes an object file to be linked by the source program or svload does not give such a definition as shown in Table 1-5, No.3 or No.4. In this case, take (a) to (b) into consideration.

(a) GLB declaration

For GLB declaration, a capacity declaration for each name can be given as shown in Table 1-5, No.1.

The compiler or assembler do not make a rationality check of this capacity with the size of the area reserved by the svdfs command. Accordingly, if the program references an address exceeding the real area, this does not result in an error.

Example: Reference of an address exceeding the declared area

<Allocator>

```
svdfs usrresp0 glb2 100

<c>
extern long glb2_g[100];
      :
glb2_g[100]=.....;
} No error is detected.
```

(b) Reference of relative address

GLB reference can be attained in the form of name $\pm\alpha$ (α denotes a relative byte address.) In this case, the range is $-2^{31} \leq \alpha \leq 2^{31}-1$.

- ② When the GLB to be referenced is defined in the same program
The above condition refers to the case where the GLB to be referenced in the source program that becomes an object file to be linked by the source program or svload gives such a declaration as shown in Table 1-5, No.3 or No.4. In this case, take (a) to (c) into consideration.

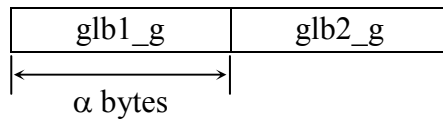
(a) Reference of GLB name only

When referencing only the GLB name with the initial value defined in the program, there are no constraints.

Example: Reference of name only

```
<c>
extern long glb2_g;
long glb1_g[3]={ (long) &glb2_g, 0, 100 };
      :
glb2_g = glb1_g[0];
```

- (b) Reference of relative addresses from the beginning of GLB
 When making reference in the form of name + α , the α value must not exceed the defined range. However, an out-of-range check error is not detected.



Namely, when $\text{glb1_g} + \beta$ is described, $0 \leq \beta < \alpha$ is a requisite.

Example: Reference of relative address

```
<c>
int glb1_g[3]={1,2,3};
int glb2_g[2]={(long)&glb1_g[0],0};... The relative address
                                         is within the range.

a = glb1_g[3] + glb2_g[4];..... The relative address
                                         is out of the range.
```

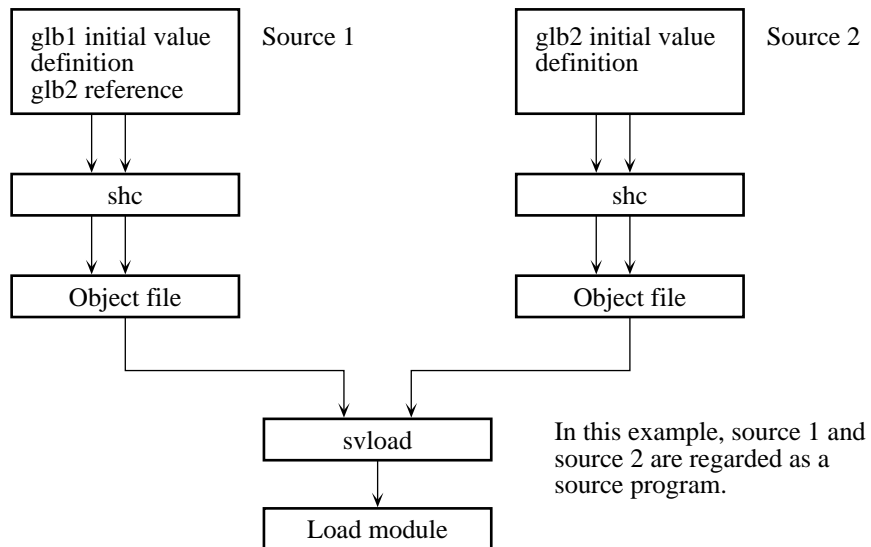
- (c) svload operation
 Suppose that a load module was created from a source program with a text section and GLB initial data. Then, if the load module is loaded by svload simply as a program or subprogram, no initial values are loaded. To load the initial value data, execute the svload after attaching the +D option to it anew. That is, the svload must be executed twice with each different option for a load module. It is recommended to create a file for initial value data only and perform execution separately. It is possible to define multiple GLB initial values in a source program.

2. PROCEDURE FOR PROGRAM DEVELOPMENT

③ Notes on linking

As already described in ① and ②, when linking multiple object files by the svload, they are regarded as a linked file even if they are different source files.

Example: Linking two object files



- ④ When the GLB initial value to be referenced is defined within the same program
 When GLB data is to be referenced at the time of program creation, the GLB initial value to be reference must not be defined within the same program.
 When a GLB initial value definition and its reference are in the same program, the reference is handled as a reference to local data.

(Examples)

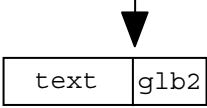
- When a program and GLB data are written within the same program

```
a.c
long glb2_g = 8;

main() {
long x;
    x = glb2_g;
}
```

When the program is as shown at left, the glb2 reference from the program is handled as a reference to the data division.

svload +P -o pname a.obj



glb2 reference



GLB initial value data is loaded as another program.

- When a program and GLB data are separated from each other

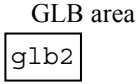
```
a.c
extern long glb2_g;

main() {
long x;
    x = glb2_g;
}
```

```
b.c
long glb2_g = 8;
```

svload +P -o pname a.obj

svload +D b.obj



glb2 reference

2. PROCEDURE FOR PROGRAM DEVELOPMENT

(4) Precautions for GLB data loading

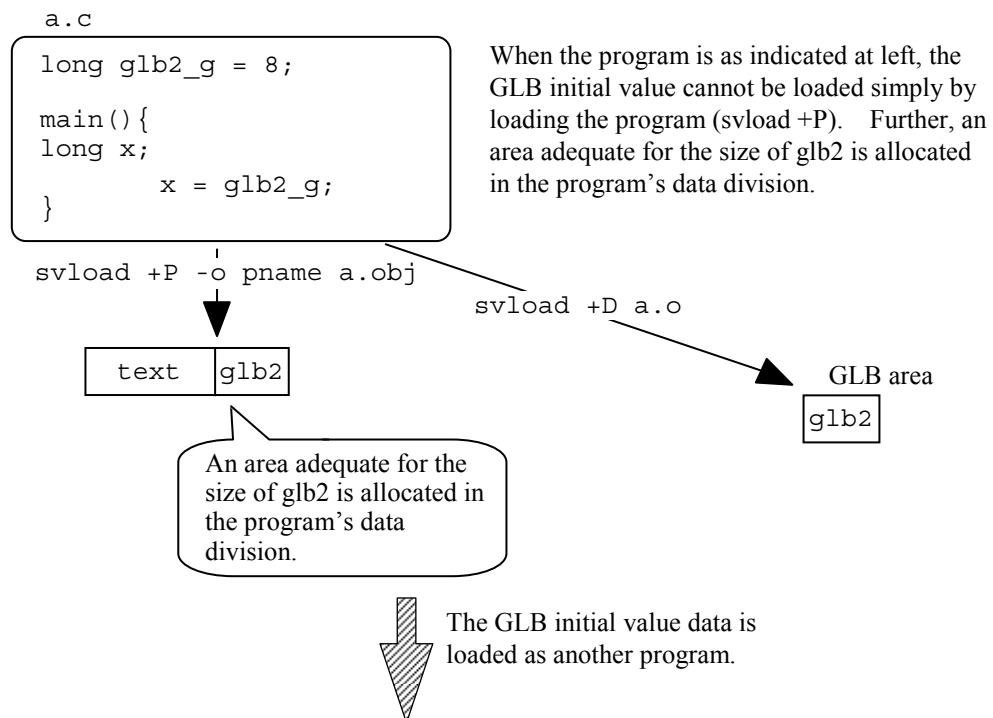
① Coexistence of a program and GLB data

In situations where a program and subprogram having a text division are written together with GLB initial value data in the same program, GLB initial value data will not be loaded even if the loader (svload) loads the program and subprogram as a program (+P) and subprogram (+I). To load GLB initial value data, it is necessary to load it with the +D option specified. In other words, one program is loaded two times with different options specified.

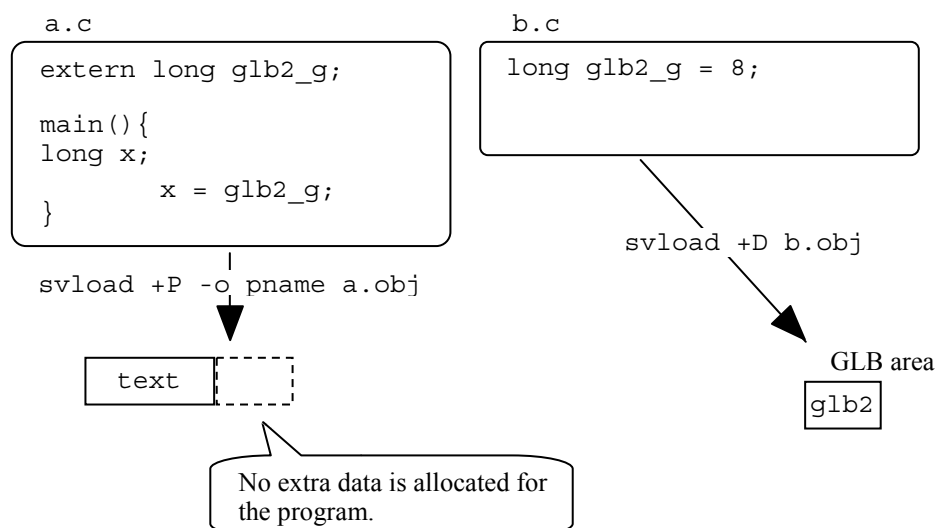
Further, an area adequate for the size of GLB initial value data is allocated in the program/subprogram data division. Therefore, the program for defining the GLB initial value data must merely formulate an initial value definition for GLB.

(Examples)

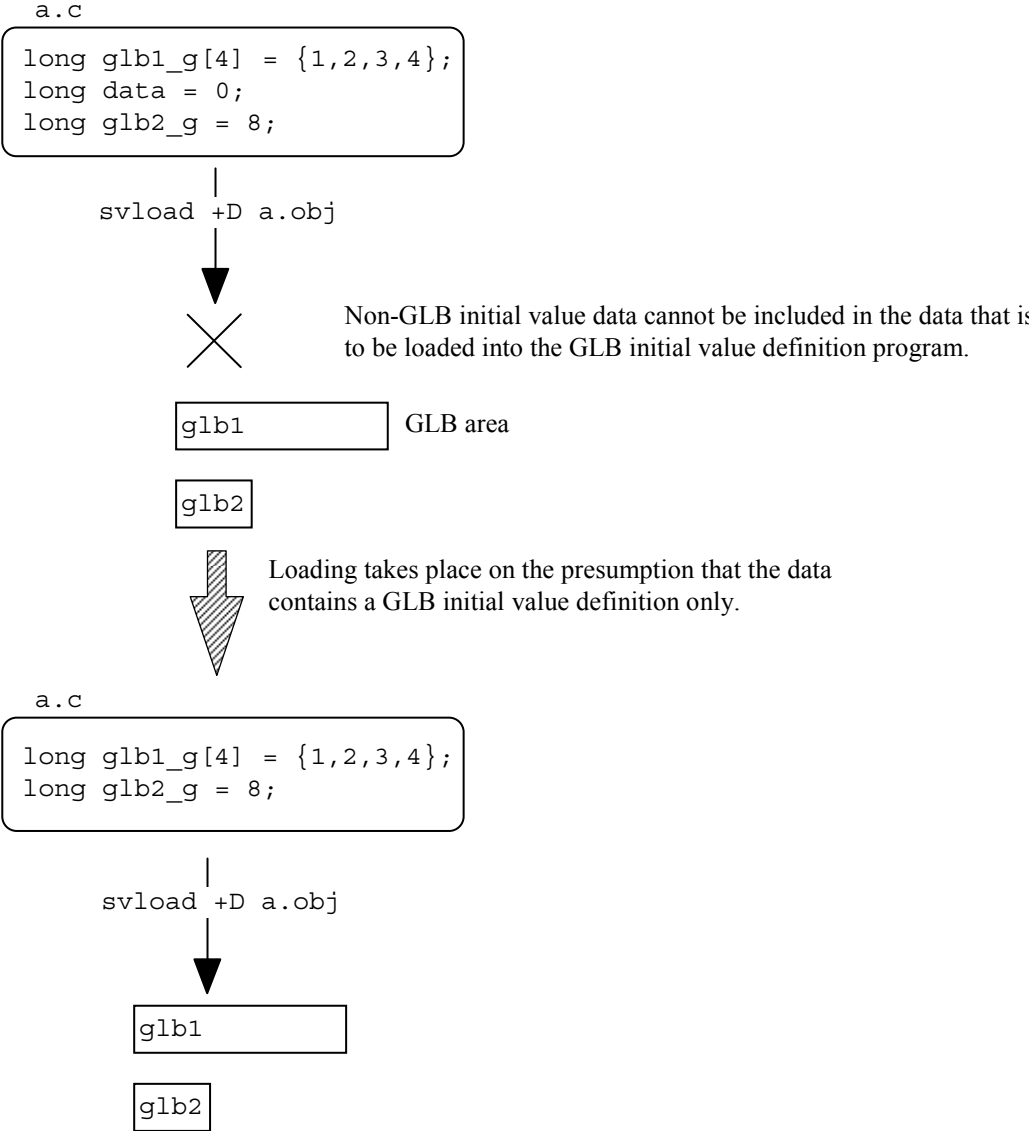
■ When a program and GLB data coexist (are written in the same program)



■ When a program and GLB data are separated from each other



- ② Coexistence of a GLB initial value and non-GLB data
A plurality of GLB initial value data can be defined in a single program.
However, no data other than a GLB initial value can be defined in a GLB initial value definition program.
(Example)
When a GLB initial value definition and non-GLB data coexist in the same program



2. PROCEDURE FOR PROGRAM DEVELOPMENT

(5) IRSUB usage

Table 1-6 IRSUB Usage

Item	C language
IRSUB reference (IRSUB name selection)	<pre>main(){ name(); }</pre> <p>[Explanation] name: IRSUB name</p>
IRSUB reference (irsubad function selection)	<pre>void *irsubad(); main(){ long no; long (*adr)(); no = xxx; adr = irsubad(no); if(adr != 0){ (*adr)(); }else{ /*Unregistered IRSUB processing*/ } }</pre> <p>[Explanation] xxx: IRSUB number The irsubad function is used to reference an address in an indirect link table.</p>

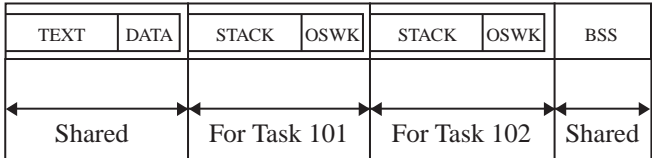
(6) Generating a multitask

When generating a program as a multitask consisting of two tasks, execute the svload/svctask command as indicated below:

```
a.c
extern long glb2_g;

main() {
long x;
    x = glb2_g;
}
```

```
svload +P -o pname -M 2 a.obj
svctask pname task101 101 -r 1
svctask pname task102 102 -r 2
```

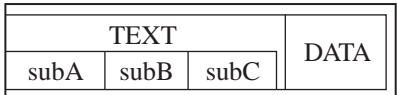


(7) Generating a multi-entry IRSUB

When generating a subprogram as a multi-entry IRSUB containing three entries, execute the svload/svbuild command as indicated below. The following example assumes that a single source file contains three modules.

```
subA.c
subA() {
long X;
    X = 10;
}
subB() {
long X;
    X = 20;
}
subC() {
long X;
    X = 30;
}
```

```
svload +I -o subA -m subB subC subA.obj
svbuild subA -ir -e 101
svbuild subB -ir -e 102
svbuild subC -ir -e 103
```



2. PROCEDURE FOR PROGRAM DEVELOPMENT

2.12 Constraints on CPMS Program Creation

For CPMS program creation, the following constraints are provided.

- (1) An overlay structure is not allowed.
For the CPMS, an overlay structure for tasks or resident subprograms is not allowed. Accordingly, when creating a task or resident subprogram, be careful not to make the program too large.
- (2) A bulk subroutine is not supported.
The CPMS does not support a bulk subroutine that is operated by storing a subprogram on the auxiliary memory and loading it on the main program as required. Use it as an indirect resident subprogram (IRSUB) or an internal subprogram (ISUB) built in a task. (An auxiliary memory cannot be attached to the controller.)
- (3) Notes on indirect resident subprogram (IRSUB) creation
The IRSUB resides on the main memory unit and used in common by multiple main programs. Accordingly, the IRSUB occupies the main memory area independent of the main program that uses it. Because the IRSUB is used by multiple main programs at the same time, make it reentrant.

A non-reentrant program cannot be an IRSUB. The term reentrant means that when the IRSUB is used by a main program, the same IRSUB can be used by another main program.

In the following, a correct procedure for creating an IRSUB is described. A reentrant IRSUB is divided into an invariable part consisting of a text division and a data division and a variable part consisting of work areas. The invariable part is shared with multiple main programs. Each main program reserves its variable part and the IRSUB uses the variable part prepared for the main program. Accordingly, program the variable part to be used by the IRSUB so that the stack area may be referenced. The conditions indicated by (a) and (b) above can be verified by checking whether the division information in the compilation list or linkage map list indicates a B division size of 0.

The IRSUB cannot use a work area (bss division) without an initial value.

When creating a reentrant IRSUB, take the following 3 points into consideration.

- (a) All the work areas should be stack areas.
- (b) When the IRSUB consists of multiple programs, don't use an area common to programs.
- (c) When the defined initial value is a static variable, don't change this value.

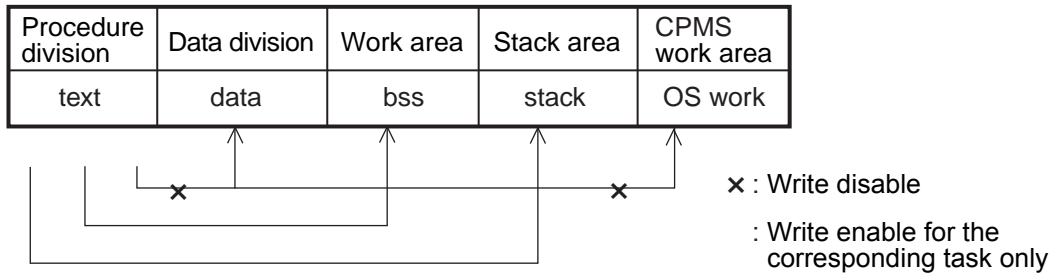


Figure 1-10 Write Enable or Disable

The number ① indicates writing into the stack area. The corresponding task can be written into the stack area.

The number ② indicates writing into the work area. Usually, in the IRSUB, no work area is reserved and writing is inhibited. The corresponding task can be written into the work area.

The number ③ indicates writing into the data division. Tasks cannot be written into the data division.

2. PROCEDURE FOR PROGRAM DEVELOPMENT

The following are notes on reentrant IRSUB creation in each language.

Example 1: Example of C language

```
int b1; ..... ①
int d1 = 10; ..... ②
static int b2; ..... ③
static int d2 = 100; ..... ④
ex() {
static int b3; ..... ⑤
static int d3 = 1000; ..... ⑥
int s1; ..... ⑦
int s2 = 20; ..... ⑧
:
:
}
```

The program that performs writing for b1 declared in ① becomes non-reentrant.

The program that performs writing for d1 declared in ② becomes non-reentrant.

The program that performs writing for b2 declared in ③ becomes non-reentrant.

The program that performs writing for d2 declared in ④ becomes non-reentrant.

The program that performs writing for b3 declared in ⑤ becomes non-reentrant.

The program that performs writing for d3 declared in ⑥ becomes non-reentrant.

If writing is performed for s1 or s2 declared in ⑦ or ⑧ respectively, the reentrant characteristic of the program is not deteriorated. For use as an IRSUB, create it with variables only as shown in ⑦ or ⑧.

The area to which each variable is assigned will be described below.

Usually, s1 is assigned to the stack area. (Note)

b2 is assigned to the bss area.

b3 is assigned to the bss area.

d1 is assigned to the data area.

d2 is assigned to the data area.

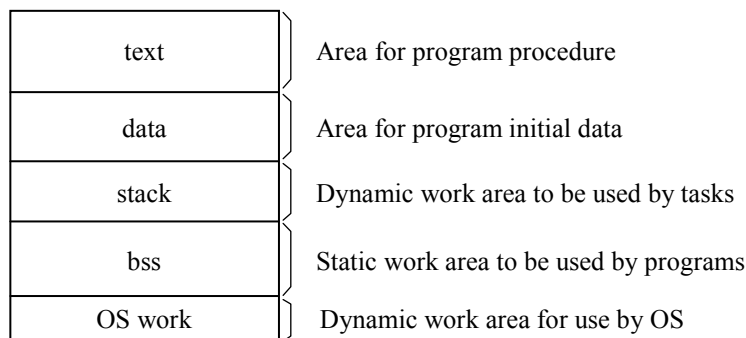
d3 is assigned to the data area.

s1 is assigned to the stack area.

s2 is assigned to the stack area.

Note: In other programs, when the initial value is set in b1, this is assigned to the data area.

- (4) No relocatability exists.
No program or subprogram relocatability exists. A program or subprogram for which the run area is set up cannot be run in another area. To run such a program or subprogram, program or subprogram deletion or re-registration must be executed.
- (5) Up to 8 characters for a name
The number of characters for a program or subprogram name is 8 or less. For a GLB or VAL name, use up to 8 characters.
For the representation of GLB or VAL in C language, `_g` and `_v` may be attached after the name. Namely, up to 10 characters are allowed.
- (6) Name for GLB or VAL
When the name ended with `_g` or `_v` is declared as an external name, it is regarded as a GLB or VAL. Accordingly, for a program that does not use the GLB or VAL, name it with the last character except `_g` or `_v`.
The name ended with `_b` is reserved as one for future extension. Don't use it.
- (7) The external name should be unique.
Name an external name uniquely among all of GLB names, program names, subprogram names and VAL names in the system.
- (8) Some names that must not be used
There are some symbols that cannot be used for a name and some symbols or statements that need care when using them for program creation. For details, see Appendix A, "NAMES USABLE IN PROGRAMS."
- (9) Program structure
The structure of programs that run under the CPMS is as follows.



The sizes of these areas are corrected respectively to a 4-byte integer. Further, each of these areas is arranged so that the start address is a multiple of 8 or 4096. For details on memory allocation, see Section 2.5.

- (10) Constraint on start address
By default, the allocator corrects the GLB area so that the start address is a multiple of 4.

2. PROCEDURE FOR PROGRAM DEVELOPMENT

(11) Treatment of initial value

Note that there is a difference in the treatment of initial value.

Area	CPMS
data	Programmed value
bss	Unfixed
stack	Unfixed

(12) Size of the initial data of GLB

After the compiler completes alignment, the data in the object file may become larger than the size defined in the source program. Use care. An example is shown below. To hasten data access, CPMS uses natural alignment in which fixed addresses are used according to the data type. Since the compiler or linker automatically places data, user code need not consider alignment. As shown below, however, the actual size of the initial data in GLB may exceed the size of the coded structure. Use care.

Size of code: 16 bytes

Size of initial data in memory: 24 bytes

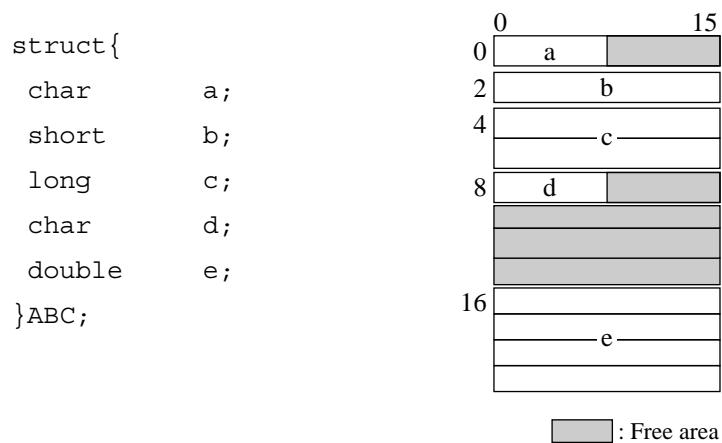


Figure 1-11 Comparison of Data Sizes

Action

- (1) When configuring the structure, consider the order in which data is placed so that free areas are not left.

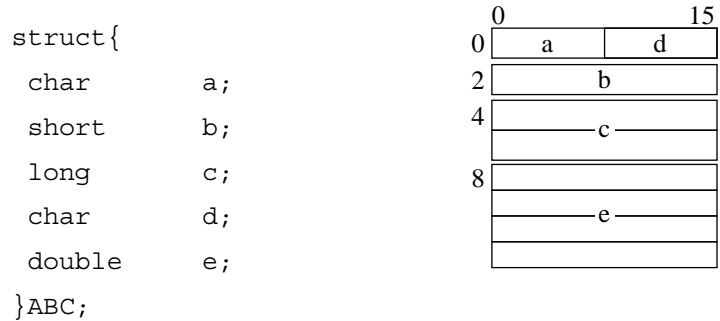


Figure 1-12 Sample Declaration of a Structure with Alignment Considered

- (2) If free areas must be left, explicitly declare that the structure contains them.
(Otherwise, free areas may or may not be allocated, depending on the machine.)

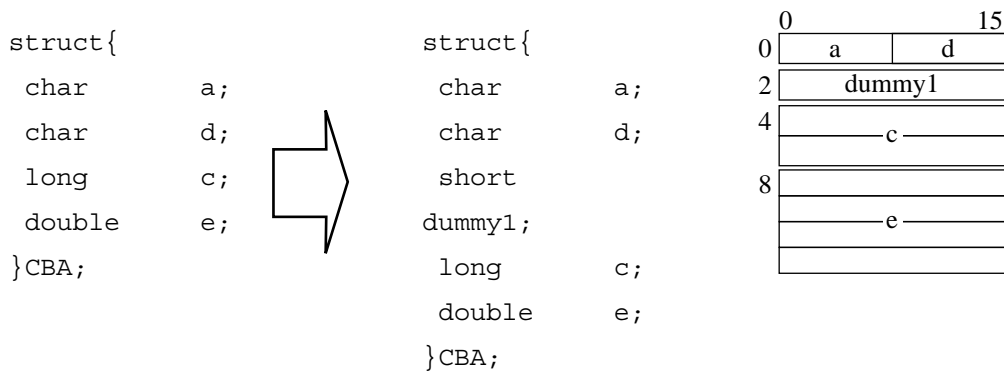


Figure 1-13 Example of Explicitly Declared Free Areas

- (3) The structure size in bytes must be a multiple of the maximum type in the structure. When allocating structure arrays, make sure that the first address of the second or subsequent structure is an accessible address.

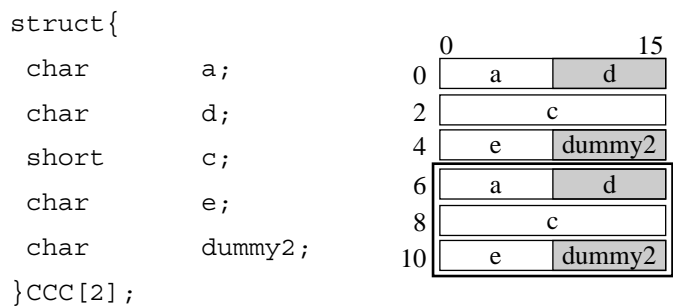


Figure 1-14 Sample Declaration with Structure Size Considered

2. PROCEDURE FOR PROGRAM DEVELOPMENT

2.13 Managing Names

<Constraints on file names for Windows®>

The name of a directory or file to be newly created on a Windows® is case-sensitive (the specified name is used without alteration). When the created directory or file is handled (for example, referenced or deleted), however, its name is not case-sensitive. Examples are shown below.

- Creation of file A → File A is created. When file a is already created, it is overwritten. The file name a remains unchanged.
- Creation of file a → File a is created. When file A is already created, it is overwritten. The file name A remains unchanged.
- Reference of file A → File a or A is referenced.
- Reference of file a → File a or A is referenced.
- Deletion of file A → File a or A is deleted.
- Deletion of file a → File a or A is deleted.

<Constraints on name management by S10V RPD P>

S10V RPD P creates programs and subprograms as files. The following rules should be observed to prevent conflicts in handling the case of file names conforming to the Windows® specifications:

- The svload and svgen commands create files and directories using user-specified file names. When these files and directories are newly to be registered, they are created and managed using the user-specified names without alteration. In subsequent registrations, they are checked for duplication without preserving their case to prevent files or directories from being overwritten. If file AB is already registered, for example, an attempt to register file AB, Ab, aB, or ab results in a duplicate definition error.

The following names can take advantage of this feature:

- svload command
Program names for tasks, names of subprograms (IRSUBs and ULSUBs)
- svgen command
Site names, unit names
- svdebug command
File names specified by the -o or -i option.
File names specified by the -f option of the sv command.

If these names have been checked only for their case in the previous specifications, they need to be reviewed.

- Commands other than indicated above are also case-insensitive.

[In the examples below, a program is already registered by “svload +P -o AB AB.obj...”.]

- (1) Program name display by the svmap command
AB is displayed. (The user-specified name at registration time is displayed as it is.)
- (2) Program registration by the svload command
svload +P -o AB AB. obj }
svload +P -o Ab AB. obj } All these commands result in a duplicate definition error.
svload +P -o aB AB. obj }
svload +P -o ab AB. obj }
- (3) Program deletion by the svdload command
svdload AB +P }
svdload Ab +P } Deletes program AB.
svdload aB +P }
svdload ab +P }
- (4) Task registration by the ctaskhr command of a specified program
svctask AB task_name task_number }
svctask Ab task_name task_number } All these tasks will be registered under
svctask aB task_name task_number } the same program name.
svctask ab task_name task_number }

CHAPTER 3 INSTALLATION AND RPDP EXECUTION ENVIRONMENT

3.1 Installation

Use an installer to install the program product (P.P.) that contains RPDP and is named “S10V_RPDP.” For installation, you must first log in as the Administrator.

Insert the S10V_RPDP disk into its drive, and then execute “SETUP.EXE” in the Disc1 folder from File manager or Windows® Explorer.

When the setup window opens, follow the on-screen instructions.

After completion of installation, you must log in anew.

3. INSTALLATION AND RPDP EXECUTION ENVIRONMENT

3.2 RPDP Execution Environment

To use RPDP, it is necessary to set up the following environment variables.

The default values of the environment variables are automatically set at the time of RPDP installation. However, environment variables set up before RPDP installation will not be overwritten except for PATH and HLNK_DIR.

The environment variables for common use by the system must be set as the system's environment variables from the [Environment Variables] tab, which can be accessed by choosing [Control Panel] and [System] in sequence.

For the meanings of environment variables necessary for shc compiler operation and the details of their settings, refer to the documentation supplied with the shc compiler package.

Table 1-7 List of RPDP Execution Environment Variables

Environment variable	Setting description	Default value	Setting unit	Remarks
SHCPU	Sets the CPU type. You must set "SH4."	SH4	Common to the system.	
SHC_INC	Sets the compiler's include file storage directory.	%SystemDrive%\Hew2\Tools\Hitachi\Sh\7_1_0\include	Common to the system.	No setup operation will be performed if the installed shc compiler package version is other than indicated below: 7.0B, 7.0.01, 7.0.02, 7.0.03, 7.1.00
SHC_LIB	Sets the compiler's installation directory.	%SystemDrive%\Hew2\Tools\Hitachi\Sh\7_1_0\bin	Common to the system.	
SHC_TMP	Sets the directory in which the compiler creates a temporary file.	%SystemRoot%\renix\tmp	Common to the system.	
HLNK_DIR	Sets the svload command's library search path.	%SystemRoot%\renix\s10v\lib;%HLNK_DIR%	Common to the system. An additional setting can be also defined for each user.	Adds %SystemRoot%\renix\s10v\lib to the beginning.
HLNK_TMP	Sets the directory in which the linkage editor creates a temporary file.	%SystemRoot%\renix\tmp	Common to the system.	
PATH	Sets the command storage directory for the compiler and RPDP.	%SystemRoot%\renix\s10v\bin;%SystemDrive%\Hew2\Tools\Hitachi\Sh\7_1_0\bin;%PATH%	Common to the system.	Adds %SystemRoot%\renix\s10v\bin;%SystemDrive%\Hew2\Tools\Hitachi\Sh\7_1_0\bin to the beginning. No setup operation will be performed if the installed shc compiler package version is other than indicated below: 7.0B, 7.0.01, 7.0.02, 7.0.03, 7.1.00
RSSITE	Sets the name of the site that is targeted for operation.	None.	Definable for each user.	
RSUTYP	Sets the user type (s: system; u: user), which becomes the default processing mode when the ALB command's -S option is not specified.	None.	Definable for each user.	

3.3 RPDP Command Storage Directory

The RPDP command storage directory is %windir%\renix\s10v\bin.

3.3.1 Environment variable setup example

This section shows an environment variable setup example.

Example of shc compiler package execution environment setup

```

SHCPU=SH4

SHC_INC=C:\Hew2\Tools\Hitachi\Sh\7_1_0\include

SHC_LIB=C:\Hew2\Tools\Hitachi\Sh\7_1_0\BIN (1)

SHC_TMP=%windir%\renix\tmp

HLNK_DIR=%windir%\renix\s10v\lib;C:\users\subsys\lib (2)

HLNK_TMP=%windir%\renix\tmp

PATH=%PATH%;%windir%\renix\s10v\bin;C:\Hew2\Tools\Hitachi\Sh\7_1_0\BIN (3)

```

- (1) Sets the compiler storage directory to C:\Hew2\Tools\Hitachi\Sh\7_1_0\bin for situations where the shc compiler package (Version 7.1) is installed with the default settings employed.
- (2) Sets RPDP's standard library storage directory to C:\WINNT\renix\s10v\lib and the user's unique library storage directory to C:\users\subsys\lib.
- (3) Sets S10V RPDP's command storage directory to C:\WINNT\renix\s10v\bin and the shc compiler's storage directory to C:\Hew2\Tools\Hitachi\Sh\7_1_0\bin.

3.3.2 Setting an environment variable from the command prompt

To edit an environment variable from the command prompt after logon, use the set command.

(Example)

```

set RSSITE=site1 (1)
set RSUTYP=s (2)

```

- (1) For RSSITE, set a site name of "site1."
- (2) For RSUTYP, set a user type of "s" (system).

3. INSTALLATION AND RPDP EXECUTION ENVIRONMENT

3.4 Registering an RPDP User Account

To log in as a non-superuser to user RPDP, use an account that belongs to a dedicated group (RPDPS10Vusers group). You can create a new account that belongs to the RPDPS10Vusers group or define an existing account as a member of the RPDPS10Vusers group.

3.4.1 Registering a new account

This section describes the procedure for creating a new account as a member of the RPDPS10Vusers group.

- (1) Log in as the Administrator.
- (2) Right-click [My Computer] and then choose [Manage] to open the [Computer Management] window.
- (3) Under “Local Users and Groups,” click “Users.”
- (4) Register a dedicated account by performing the following steps:
 - ① From the [Action] menu, choose [New User].
 - ② When the [New User] dialog box opens, enter a user name and other necessary items of information to register a new account. You can enter any desired user name and password.
 - ③ Select the created user and then click the [Member Of] tab. Click the button and then double-click “RPDPS10Vusers” to add the created user. The “RPDPS10Vusers” group is automatically registered during S10V RPDP installation.
 - ④ Click the button to finish.

3.4.2 Adding the RPDPS10Users group as a group to which an existing account belongs
This section describes the procedure for defining an existing account as a member of the RPDPS10Users group.

- (1) Log in as the Administrator.
- (2) Right-click [My Computer] and then choose [Manage] to open the [Computer Management] window.
- (3) Under “Local Users and Groups,” click “Users.”
- (4) Perform the following steps to add the “RPDPS10Users” group as a group to which an existing account belongs:
 - ① Select the user to be defined as a member of the “RPDPS10Users” group.
 - ② Click the [Member Of] tab. Click the button and then double-click “RPDPS10Users.” The “RPDPS10Users” group is automatically registered during S10V RPDP installation.
 - ③ Click the button to finish.

CHAPTER 4 COMPILER

This chapter gives detailed descriptions of the PC's compiler and assembler. For the command reference, see "PART 2 COMMAND REFERENCE."

As regards compiler/assembler use, RPDP assumes the use of Hitachi Microcomputer Development Environment System SuperH RISC Engine C/C++ Compiler Package Version 7 or later (which is hereinafter referred to as the shc compiler).

4.1 Details of C Compiler Options

This section describes the method of compiling with shc as well as the compiling precautions. For detailed specifications for shc, refer to the documentation supplied with the shc compiler.

- Command format

```
shc [ $\Delta$ <option>...][ $\Delta$ <filename>[ $\Delta$ <option>...]...]
```

(Example) shc Δ test1.c Δ test2.c [Enter]

- Environment variable setup

When you use shc, complete environment variable setup as indicated in Table 1-8.

Table 1-8 Setup Environment Variables

Environment variable	Setup description
path	<p>Add the installed compiler package's executable file storage directory to the environment variable "path."</p> <p>It is necessary to set the paths to the compiler (shc), assembler (asmsh), and optimization linkage editor (optlnk).</p> <p>This environment variable must always be specified.</p> <p>Specified format: path= <executable file pathname> [;<existing pathname>;...]</p>
SHC_LIB	<p>Specify the directory in which the compiler's load module and system include file are stored.</p> <p>This environment variable must always be specified.</p> <p>Specified format: set SHC_LIB= <executable file pathname></p>
SHCPU	<p>Specify the target CPU type. For this system, set SH4 as SHCPU.</p> <p>When this environment variable is not specified, the system assumes that the CPU type is SH1. The CPU type can also be specified using the -cpu option.</p> <p>Specified format: set SHCPU= <CPU></p>
SHC_INC	<p>Specify the compiler's include file storage directory. The system include file search will be conducted in the include option-specified directory, SHC_INC-specified directory, and system directory (SHC_LIB) order.</p> <p>The user include file search will be conducted in the current directory, include option-specified directory, and SHC_INC-specified directory order.</p> <p>Specified format: set SHC_INC = <include pathname> [;<include pathname> ;...]</p>
HLNK_DIR	<p>Specify the input file storage directory for the optimization linkage editor.</p> <p>Files specified by the input option/library option will be searched for in the current directory and HLNK_DIR-specified directory order.</p> <p>The loader's library search path also complies with the HLNK_DIR setting.</p> <p>Specified format: set HLNK_DIR= <input file pathname>[;<input file pathname> ;...]</p>
SHC_TMP	<p>Set the directory in which the compiler creates a temporary file.</p> <p>set SHC_TMP= <directory></p>
HLNK_TMP	<p>Set the directory in which the linkage editor creates a temporary file.</p> <p>set HLNK_TMP= <directory></p>

4. COMPILER

4.2 Compiling Precautions

4.2.1 When compiling with shc commands

- Floating-point number handling

With shc, it is possible to exercise control over floating-point denormalized number handling and rounding with the compilation option.

However, it is well to remember that the standard library to be linked at the time of loading varies the handling method.

The options for providing control over denormalized number handling and rounding as well as the corresponding standard libraries are indicated below (the loader links to libsh4nbmdn.lib if no library is specified at the time of loading):

Table 1-9 Options for Floating-point Number Handling

	Specifications	Option	Default
Denormalized number handling	Handled as 0.	-denormalization=off	Handled as 0.
	Handled as a denormalized number (*).	-denormalization=on	
Resulting value rounding	A nonsignificant portion will be disregarded.	-round=zero	Disregarded.
	A nonsignificant portion will be rounded.	-round=nearest	

(*) The SH4 (SH7751), which is the S10V's CPU, does not support denormalized numbers. Therefore, it handles a denormalized number as 0 during execution.

Table 1-10 Floating-point Number Handling and Corresponding Standard Libraries

	-denormalization	-round	Standard library
Specified option	off	zero	libsh4nbmzz.lib
	on	zero	—
	off	nearest	—
	on	nearest	libsh4nbmdn.lib

- Generating and saving a compilation list (shc)

Generate and save a compilation list because it is necessary, for instance, for calculating the stack size for use by a task. To generate a compilation list, specify the following options.

Specify the `-listfile` option before the C source files to be compiled. If the option is specified after the C source files, the resulting compilation list covers the last file only.

- Specifying the option for compilation list generation

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

If the list file name is not specified, a file will be generated with the extension “.lst” attached to the same file name as the source file name.

(Example)

```
◆ shc △-listfile △test1.c △test2.c [Enter]
The listfile option is valid for both test1.c and test2.c.
◆ shc △test1.c △test2.c △-listfile [Enter]
The listfile option is valid for test2.c only.
```

- Specifying “-fpscr=safe” for built-in subroutine compilation

The shc compiler allows you to specify (`-fpscr`) whether or not to guarantee the FPSCR register’s precision mode before and after a function call. The default setting is “`-fpscr=aggressive`” so that the precision will not be guaranteed before and after a function call. Therefore, a code will be generated after the return from a function call so that the FPSCR register reverts to the single-precision mode.

The S10V cannot perform a floating-point operation within a built-in subroutine. (If an attempt is made to perform a floating-point operation, the CPU comes to a stop due to an “FPU Unavailable” exception.

When a function is called in situations where compilation is conducted with a built-in subroutine (`-fpscr=aggressive`) specified, the FPSCR is accessed to cause an “FPU Unavailable” exception even if a floating-point operation is not being performed.

When compiling a built-in subroutine (or an IRSUB that is to be called by a built-in subroutine), therefore, specify the “`-fpscr=safe`” option.

4. COMPILER

4.3 Differences between m68k and shc

4.3.1 Command line options

Table 1-11 shows the comparison between m68k and shc command line options.

Table 1-11 Comparison between m68k and shc Command Line Options

m68k	Shc	Meaning
-c	- <u>c</u> ode= <i>machinecode</i>	Does not provide linking. Generates an object module.
-Dname	- <u>d</u> efine=name	Defines the name.
-Dname=def	- <u>d</u> efine=name=def	Defines the name in the "def" position.
-E	—	Does not provide compilation. Outputs the preprocessor processing results to the standard output.
-g	- <u>d</u> ebug	Generates debug information.
—	- <u>l</u> istfile - <u>s</u> how= <u>s</u> ource, <u>o</u> bject	Inserts a source file line into an assembler source.
ANSI compliance by default	ANSI compliance by default	Compiles ANSI Standard C compliant programs only.
—	- <u>e</u> ndian= <i>big</i>	Performs compilation in big-endian mode (default is "big").
—	- <u>e</u> ndian= <u>l</u> ittle	Performs compilation in little-endian mode.
—	- <u>s</u> jis (default)	Provides kanji (Shift JIS) support. This option can be specified only when the K&R specification is complied with.
—	- <u>s</u> how=length=n	Specifies the number of lines per source list page.
—	- <u>l</u> istfile - <u>l</u> istfile=filename	Displays a source list.
-Idir	- <u>i</u> nclude=dir	Adds an include file search directory.
-O	- <u>o</u> ptimize=0	Sets the optimization level. <m68k> n=b, c, e, g, i, j, l, R, r, s, t <shc> optimize=0: provides no optimization; optimize=1: provides optimization. You can select an optimization method with -speed, -nospeed, or -size.
-On	- <u>o</u> ptimize=1	
	- <u>s</u> peed - <u>n</u> ospeed - <u>s</u> ize	
-P	- <u>p</u> reprocessor[=file]	<m68k> Performs preprocessor execution only and stores the result in the .i file. <shc> Performs preprocessor execution only and stores the result in the .p file.
-S	- <u>c</u> ode=asmcode	Generates an assembler source. Does not start the assembler or linker.
-Uname	—	Undefines the name.

Underlined () portions of shc options represent an acceptable abbreviated character string. By default, italic characters take effect.

Table 1-12 lists the shc options.

Table 1-12 List of shc Options

Function	Specified format	Meaning
CPU type	<code>-cpu=sh4</code>	Generates an SH-4 object.
List file output	<code>-listfile[=filename]</code>	Generates a compilation list.
Compilation list output form (This option is operative when it is specified together with the <code>-listfile</code> option.)	<code>-show=source nosource</code> <code> object noobject</code> <code> statistics nostatistics</code> <code> include noinclude</code> <code> expansion noexpansion</code>	Specifies whether or not to provide a source list. Specifies whether or not to provide an object list. Specifies whether or not to provide statistical information. Specifies whether or not to provide a post-include-expansion list. Specifies whether or not to provide a post-macro-expansion list.
Character string data storage section	<code>-string=<u>c</u>onst</code> <code> <u>d</u>ata</code>	Outputs a character string to a constant area section (C). Outputs a character string to an initialized data area section (D).
Return value sign/zero extension	<code>-rtnext</code> <code>-nortnext</code>	Provides return value sign/zero extension. Does not provide return value sign/zero extension.
Denormalized number handling	<code>-denormalization=<u>o</u>ff</code> <code> <u>o</u>n</code>	Handles a denormalized number as 0. Handles a denormalized number as a denormalized number.
Floating-point number rounding direction	<code>-round=<u>z</u>ero</code> <code> <u>n</u>earest</code>	Rounds to zero. Rounds to nearest.

4. COMPILER

4.3.2 Language specification differences

Table 1-13 shows the differences between the mcc68k and shc language specifications and the precautions to observe at the time of changeover. The comparison table below shows the specification differences only. Further, the extended capabilities unique to mcc68k and shc are excluded from the table. Programs using extended capabilities unique to mcc68k need to be reviewed.

Table 1-13 Language Specification Comparison

Compared item	mcc68k	shc	Precautions for changeover
Number of effective identifier characters (external)	510 characters	8189 characters	
Number of effective identifier characters (internal)	512 characters	8191 characters	
Alignment	2-byte alignment	Natural alignment (4-byte alignment for double)	Modify a program that uses a constant for monitoring the structure size and arrangement.

4.3.3 Function call rules

For the function call rules, click [Start], point to [Programs], and start [Hitachi Embedded Workshop2 Online Manuals [SuperH] - Japanese (Ver-Rev)], and then see under “Programming” in the SuperH RISC Engine C/C++ Compiler/Assembler/Optimization Linkage Editor User Manual.

CHAPTER 5 PROGRAMMING COMMANDS

5.1 Notes on Programming Commands

As S10V programming commands, the librarian and linker offer optlnk and svload. For optlnk, click [Start], point to [Programs], and start [Hitachi Embedded Workshop2 Online Manuals [SuperH] - Japanese (Ver-Rev)], and then see under “Optimization Linkage Editor Operating Procedures” in the SuperH RISC Engine C/C++ Compiler/Assembler/Optimization Linkage Editor User Manual. For svload, see Chapter 8, “Loader.”

CHAPTER 6 GENERATION

6.1 Overview

6.1.1 Purpose of system generation

In real-time program development for a S10V, the system is constructed according to the system type and operation type of the user. This job is called system generation. To support system generation, system generation commands are provided. The user can use these commands to construct an environment that enables development of real-time programs for the PCs system on a development machine. Also, these commands facilitate extension of the PCs system after the system is constructed.

6.1.2 Features of system generation

At the time of system generation, the user can enjoy the following features:

- Generating a system (site) construction environment
The framework of an S10V development environment (site) can be generated and registered as site management information.
- Generating configuration definition information
The definition information necessary for PCs construction can be set up for each site to construct a development environment.
- Displaying configuration definition information
This function is used to confirm the definition information about a defined PCs.
- Duplicating an existing site
This function is used to make a duplicate of an existing site with a view toward simplifying the PCs construction procedure. The configuration definition information about such a duplicate can be edited to construct a new site.
- Deleting a site
This function is used to delete an unnecessary site.

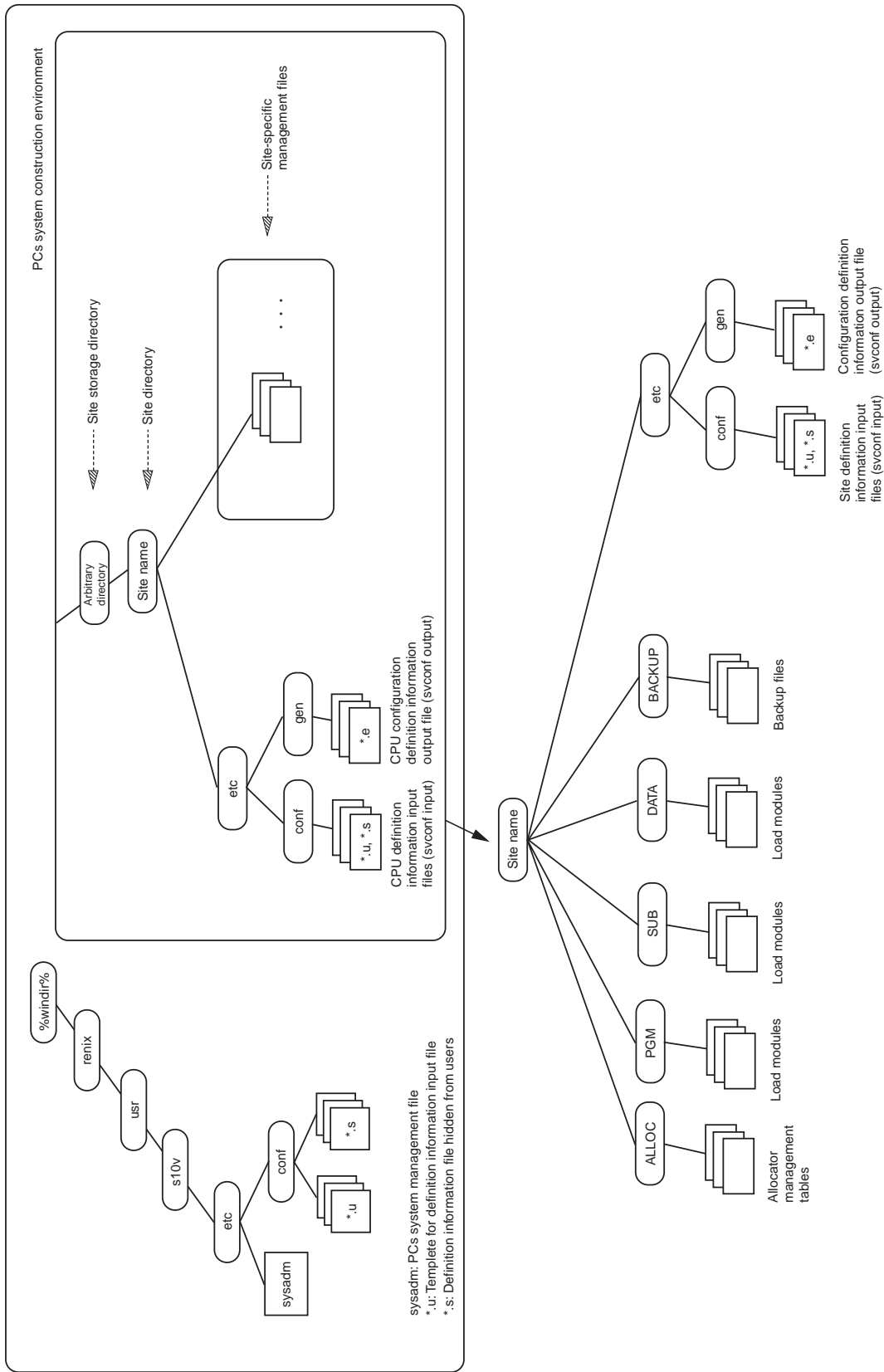


Figure 1-15 PCs System Overall Configuration

6. GENERATION

6.1.3 Site management method for a plurality of PCs units

With RPDP, it is possible to manage the real-time program development environment for a plurality of PCs units in a single development machine environment.

RPDP manages PCs units with a “site name” assigned to each of them

- Site name

One site name is to be assigned to each PCs. The site name requirements are set forth below:

- The site name consists of 1 to 14 characters.
- The site name may consist of single-byte alphanumeric characters and underscores (_). However, it must begin with an alphabetical character.

- Site management method

The information about each site is managed on the file system of the development machine as indicated below. Figure 1-16 shows the RPDP-related directory structure of each model.

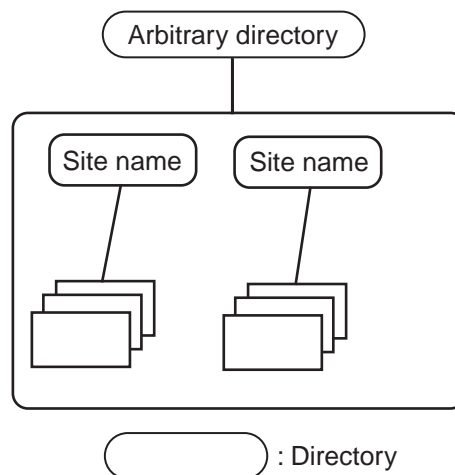


Figure 1-16 Site Directory Structure

The system generation commands create the following directory structure on the development machine:

- Individual site directories will be created under an arbitrary directory that is specified by the SUBSYS_DIR identifier within a user definition file specified by the svgen command.
- Directories and files for use by RPDP will be created within the directory of each site. The files contain the information that relates to the PCs system configuration.

6.2 System Generation Commands and Flow of System Construction

(1) Commands

Table 1-14 lists the system generation commands supported by RPDP.
You need administrator privilege to execute the commands.

Table 1-14 System Generation Commands

Command	Function	Description
svgen	Creates a PCs construction environment.	Constructs directories and sites on the development machine as needed for PCs site management.
svconf	Generates configuration definition information.	Generates a configuration definition information output file in accordance with the contents of a site configuration definition information input file.
svshconf	Displays configuration definition information.	Displays the configuration definition information about a constructed site.
svsitecp	Duplicates an existing site.	Duplicates the environment of a specified site to generate a new site. However, this function does not duplicate the configuration definition information.
svsitedel	Deletes a site.	Deletes a specified site's environment file from the development machine.

(2) Flow of generation

Procedures for system generation are divided into two types: new and modification.

New: Procedure to construct a new PCs system

Modification: Procedure to modify the already-constructed PCs system by changing information defined for the system

6. GENERATION

(3) Procedure for newly constructing a PCs system

Figure 1-17 shows a procedure for newly constructing a PCs system.

Step	Flow	Description
1	<pre> graph TD Start([Start]) --> Step1[Create a site construction environment definition file with an editor.] Step1 --> Step2[Create a system (site) construction environment with svgen.] Step2 --> Step3[Create various site configuration definition information files with an editor.] Step3 --> Step4[Construct a controller system environment with svconf.] Step4 --> End([End]) </pre>	Create a site construction environment definition file for the purpose of site construction environment creation (see Section 6.3.1, “User setting definition information”).
2		When executed, the svgen command creates directories and files necessary for the development machine in accordance with the site construction environment definition file.
3		Create a site configuration definition information file for the purpose of PCs system construction (see Section 6.3.3, “Contents of network definition information,” and Section 6.3.4, “Site definition information”).
4		Execute svconf sitename to construct a site.

Figure 1-17 Procedure for Newly Constructing a Controller System

(4) Procedure for modifying a PCs system

Figure 1-18 shows a procedure for modifying a PCs system.

(a) Modifying the PCs system

Step	Flow	Description
1	<pre> graph TD Start([Start]) --> Step1[Edit various site configuration definition information files with an editor.] Step1 --> Step2[Reconstruct a controller system environment with svconf.] Step2 --> End([End]) </pre>	Edit the site configuration definition information input files for the purpose of PCs system reconstruction (see Section 6.3.3, “Contents of network definition information,” and Section 6.3.4, “Site definition information”).
2		Execute svconf sitename to reconstruct a site.

Figure 1-18 Procedure for Modifying a PCs System

(5) Constructing a site by copying another site

(a) Copying a site within the same development machine

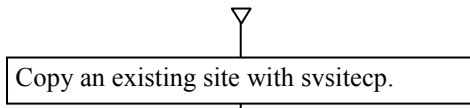
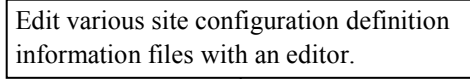
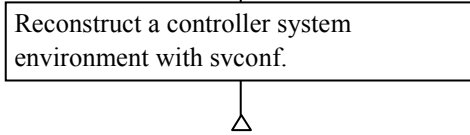
Step number	Flow	Description
1		Copy an existing site.
2		Edit the site configuration definition information files for the purpose of controller system reconstruction (see Section 6.3.3, “Contents of network definition information,” and Section 6.3.4, “Site definition information”).
3		Execute svconf sitename to reconstruct a site.

Figure 1-19 Procedure for copying a site within the same development machine

6. GENERATION

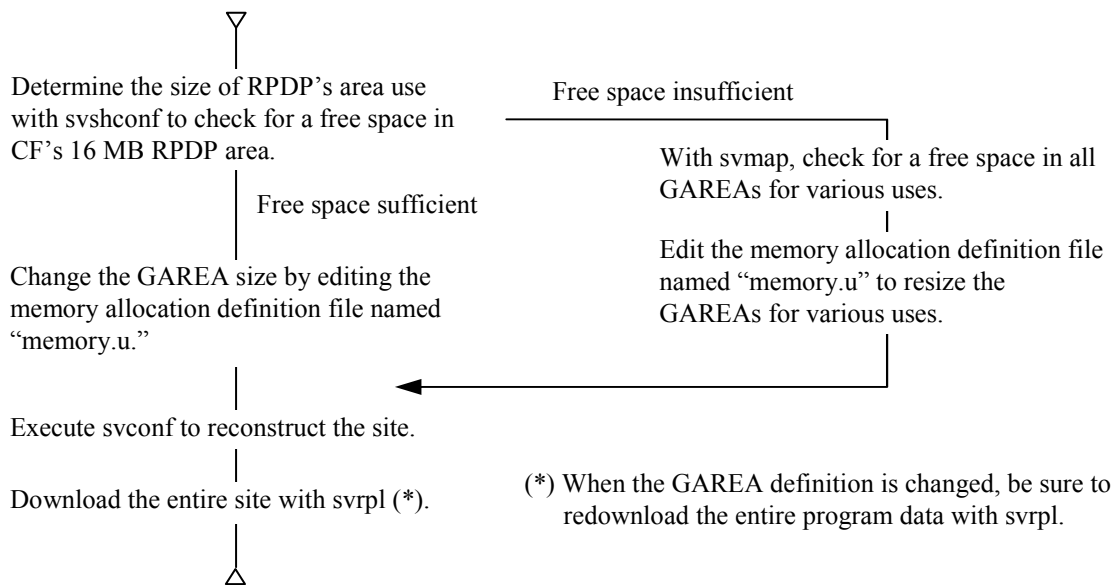
<Precautions for area partitioning>

(1) Increasing the GAREA size when it is insufficient

In S10V RPDP area management, ensure that the physical memory sizes available for various purposes of task and GLB uses are allocated as GAREAs for various uses at the time of site construction. If any GAREA is insufficient after site construction, you can enlarge it as far as there is a free space within the physical memory. When no free space is found in the physical memory, you have to make adjustments by, for instance, reducing the GAREA size allocated for a different use.

When you change the GAREA size, reupload the entire site with svrpl.

The procedure for increasing the GAREA size after site construction is indicated below:



(2) Avoiding GAREA fragmentation

For S10V RPDP area management purposes, the physical memory and logical space are mapped at the ratio of 1:1. Therefore, the range of split area allocation to the logical space for each use, such as task or GLB use, is from the beginning of each logical space to its end that is defined as the GAREA size at the time of site construction. As such being the case, even when the GAREA range contains a large number of small free areas to which no split areas are allocated, no new split area can be allocated until a contiguous free space of an adequate size is available.

To defragment a free area within a GAREA, it is necessary to rearrange split areas. To rearrange the split areas, you must reload the programs and subprograms within the split areas. As regards GLB, you must redefine the secondary partition areas and reload initial value data. When GLB split areas are rearranged, you must reload the programs and subprograms that reference the GLB split areas. As regards GLB, you must redefine the secondary partition areas and reload initial value data. When GLB split areas are rearranged, you must reload the programs and subprograms that reference the GLB split areas.

Consequently, you should exercise care so that a free area within a GAREA does not become fragmented. When changing the size of a split area for deallocation/reallocation purposes, GAREA fragmentation is likely to occur. For prevention of fragmentation, it is recommended that you allocate split areas while taking their future extension into account, thereby minimizing the necessity for split area deallocation/reallocation.

6.3 PCs System Definition Information

The PCs system comprises a development machine, a PCs, and a network and PI/O for interconnecting them. A site is to be allocated to the PCs for construction purposes. Table 1-15 shows the contents of site definition information.

Table 1-15 Contents of Site Definition Information

Type	Definition information	Management unit
Site construction environment definition information	Definition information about the PCs system managed by the development machine. It relates to the site storage directory and site directory specified by the site construction environment definition file.	The entire PCs managed by the development machine is targeted.

6.3.1 User setting definition information

The user setting definition information must be set up by the user at the time of PCs construction. It is roughly divided into the following two items:

- Site construction environment definition information
This information is necessary when generating a site construction environment with the svgen command. The user must write this information in an arbitrary file.
 - Site configuration definition information
This information is necessary when generating the PCs system's real-time development environment within a site environment with the svconf command. A template file is generated in the site storage directory when the site environment is created with the svgen command.
- (1) Site construction environment definition information
This information is necessary when generating a site construction environment with the svgen command. The user must write this information in an arbitrary file.

■ Setup items

Item	Description
Site storage directory	Specify the path to the directory that stores a site directory. Enter a full pathname, which includes a drive letter.
Site name	Specify the site name.

6. GENERATION

■ Entry format

```
SUBSYS_DIR=△site-storage-directory  
SITE_NAME=△site-name
```

Keywords are indicated in bold italics. Ensure that each entry contains a keyword. The triangular mark represents one or more spaces or a tab.

• Definition example

A system definition example is shown below:

```
SUBSYS_DIR= c:\sitedir  
SITE_NAME= pcs10
```

■ Site naming rules

Enter a site name consisting of no more than 14 characters. Single-byte alphanumeric characters and “_” (underscores) can be used to enter a site name.

(2) Site configuration definition information

This information is necessary for generating a site environment with the `svconf` command. It must be written in the site configuration definition information file. A template for the site configuration definition information file is generated under `etc\conf` within the site directory when the site configuration environment is generated with `svgen`.

Table 1-6 lists the site configuration definition information.

For details, see Section 6.3.3, “Contents of network definition information,” and Section 6.3.4, “Site definition information.”

Table 1-16 User-Defined Site Configuration Definition Information Setup Items

Type	Definition information	Setup item	Identifier	Input attribute (range)	Default value	Input file name
Network definition information	Although this information formulates a network adapter definition, fixed values are used. Set network adapter definition information on an individual interface basis.	Slot number	SLOT	Decimal number (fixed at 0)	—	adapter.u
		Station number	STATION_NO	Decimal number (fixed at 1)	—	
		Interface number	IF_COUNT	Decimal number (fixed at 1)	—	
		UCB number	UNO	Decimal number (fixed at 1)	—	
		Interface number	IF_NUM	Decimal number (fixed at 1)	—	
		IP address	IP_ADDR	IP address in X1.X2.X3.X4 form	—	
		Subnet mask	NETMASK	IP address in X1.X2.X3.X4 form	—	
		Broadcast address	BRD_ADD	IP address in X1.X2.X3.X4 form	—	
		Task GAREA size (KB)	TASKSZ	Decimal number (a multiple of 16)	—	
		Read-only GLB GAREA size (KB)	GLBRSZ	Decimal number (a multiple of 16)	—	
Processor definition information	Set the information about the memory for use by the site.	Read/write GLB GAREA size (KB)	GLBRWSZ	Decimal number (a multiple of 16)	—	memory.u
		Subprogram GAREA size (KB)	SUBSZ	Decimal number (a multiple of 16)	—	

<Notes>

- Set the IP address for the S10V unit with the S10V tool. By default, the IP address is set at 192.192.192.1. Address 192.168.0.1 is not available because it is an E-port address for the R600.
- RPDP commands cannot be used via the NCP-F or LANCP. They are available only when they use Ethernet via the CMU.
- RPDP can define memory sizes of up to 16 MB.

6. GENERATION

6.3.2 Contents of PCs system definition information

(1) PCs system definition information

The PCs system definition information is created and updated in accordance with the site construction environment definition file, which is a parameter for the svgen command. The “site storage directory name” and “site directory name,” which are written in the site construction environment definition file by the user, are stored in the sysadm file (within %windir%\renix\usr\s10v\etc).

[Setup items]

Item	Description
Site storage directory	Specify the path to the directory that stores a site directory. Enter a full pathname, which includes a drive letter.
Site name	Specify the site name.

[Setup example]

The example below shows a site construction environment definition file setup for creating the following directory structure on the development machine:

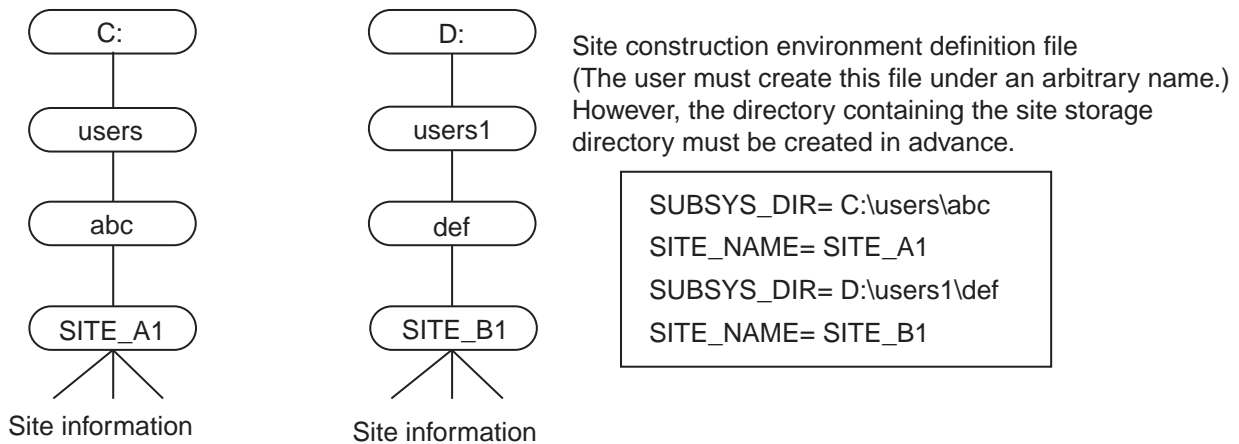


Figure 1-20 RPDP-Related Directory Structure and Site Construction Environment Definition File Configuration

6.3.3 Contents of network definition information

(1) Adapter information (adapter.u)

This item of information formulates a CMU definition.

[Setup items]

- SLOT (slot number)
Set the slot number for network adapter definition.
The slot number for the CMU is fixed at 0.
Be sure to set the slot number because no default value is predefined.
- STATION_NO (station number)
Set the rotary switch information about the hardware's station number.
The station number for the CMU is fixed at 1.
Be sure to set the slot number because no default value is predefined.
- IF_COUNT (interface count)
Set the number of interfaces (channels) to be used.
The default interface count is 1.
The interface count for the CMU is fixed at 1.

(2) Interface startup information

This item of information defines each network adapter interface.

Define a set of the following setup items:

[Setup items]

- UNO (UCB number)
Set the UCB number to be defined for an interface. The UCB number for the CMU is fixed at 1.
Be sure to set the UCB number because no default value is predefined.
- IF_NUM (interface number)
Set the interface number to be used. The default interface number is 1.
The interface number for the CMU is fixed at 1.
- IP_ADDR (IP address)
Set the local IP address in dotted notation.
Be sure to set the IP address because no default value is predefined.
- NETMASK (subnet mask)
To divide the network into subnetworks, enter a subnet mask in dotted notation for address occupation range definition purposes.
Be sure to set the subnet mask because no default value is predefined.
- BRD_ADD (broadcast address)
The address for indicating a broadcast to the network must be set in dotted notation.
Be sure to set the broadcast address because no default value is predefined.

6. GENERATION

<Setup example for specifying no subnetwork>

Address class	IP address	Subnet mask	Broadcast address
Class A	0 to 126.X1.X2.X3	255. 0. 0. 0	0 to 126.255.255.255
Class B	128 to 191.X1.X2.X3	255.255. 0. 0	128 to 191. X1.255.255
Class C	192 to 223.X1.X2.X3	255.255.255. 0	192 to 223. X1. X2.255

The X1, X2, and X3 values are between 0 and 255.

Descriptions of IP addresses for various classes

- For Class A, the network address section consists of 0 to 126, and the host address section consists of X1.X2.X3.
- For Class B, the network address section consists of 128 to 191.X1, and the host address section consists of X2.X3.
- For Class C, the network address section consists of 192 to 223.X1.X2, and the host address section consists of X3.

6.3.4 Site definition information

(1) Information about memory

The S10V defines the information about memory and stores it in the memory.u file in the “site-storage-directory\site-name\etc\conf” directory. The detailed information about memory use is to be set in the file.

[Setup items]

The table below shows the setup items.

Table 1-17 Site Construction Definition Information about Memory

Type	Definition information	Setup item	Identifier	Input attribute (range)	Default value	Input file name	
Processor definition information	Memory information	Set the information about the memory used by the site.	Task GAREA size (KB)	TASKSZ	Decimal number (a multiple of 16)	—	memory.u
		Read-only GLB GAREA size (KB)	GLBRSZ	Decimal number (a multiple of 16)	—		
		Read/write GLB GAREA size (KB)	GLBRWSZ	Decimal number (a multiple of 16)	—		
		Subprogram GAREA size (KB)	SUBSZ	Decimal number (a multiple of 16)	—		

Note: The total size of memory information up to 16 MB.

6.4 PCs System Definition Information Display

The PCs system definition information display function displays the PCs system definition information that is generated by the svgen or svconf command.

For the display operation and the contents of the resulting displayed information, see the svshconf command reference.

6.5 PCs System Environment Duplication

6.5.1 Duplication unit

The PCs system environment duplication function generates a copy of the PCs system environment on an individual site basis. Even when a site is copied in this manner, the configuration definition information is not copied. Be sure to correct the definition information in order to achieve reconstruction.

For the site duplication operation, see the svsitecp command reference.

6.5.2 Duplication range
 The site duplication function copies the following scope:

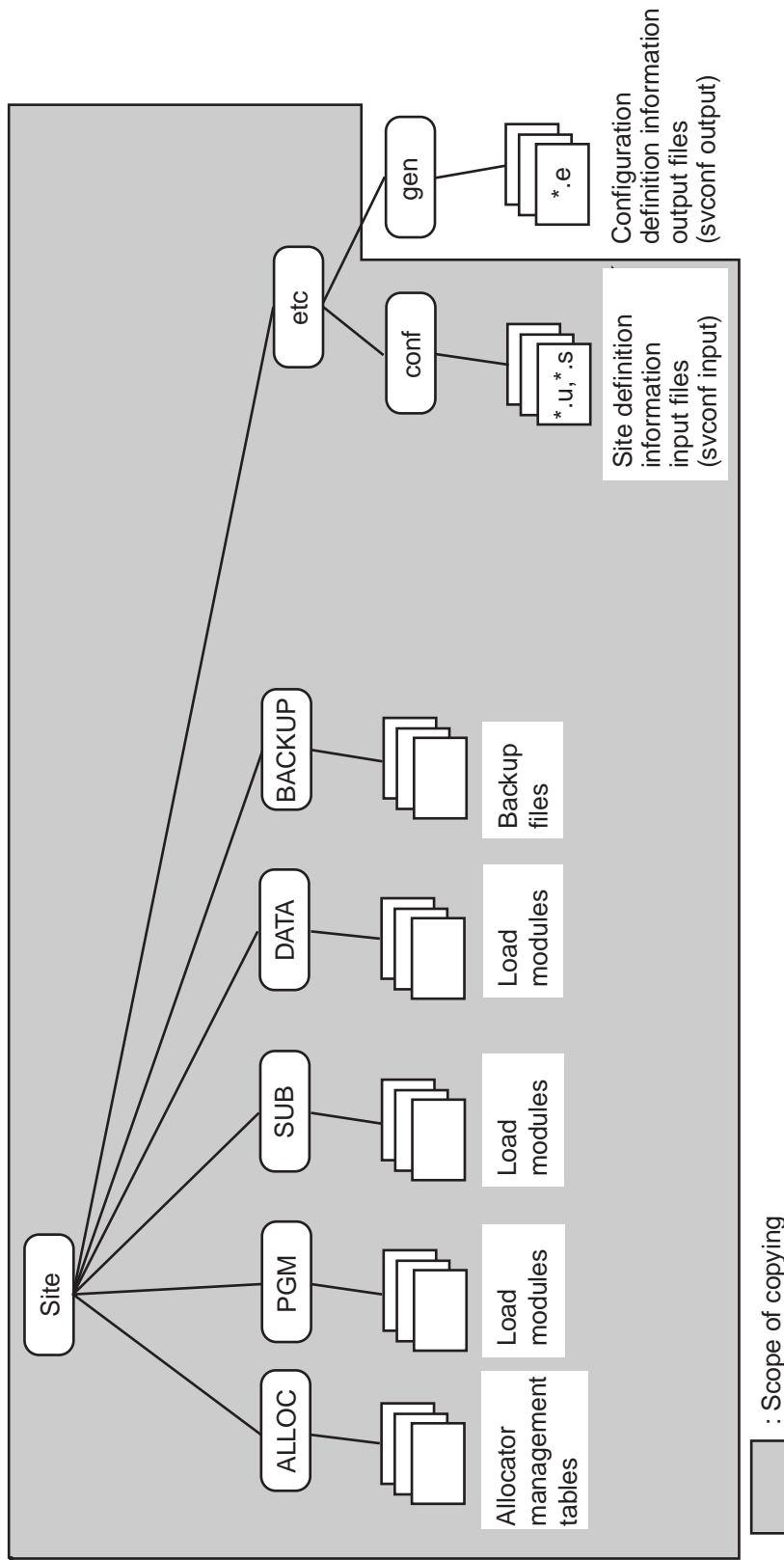


Figure 1-21 Scope of Copying Provided by the Site Duplication Function

6.6 PCs System Environment Deletion

The PCs system environment deletion function deletes files within a specified site environment and deletes the management information from the controller system management file.

For the site deletion operation, see the `svsitedel` command reference.

CHAPTER 7 ALLOCATOR

7.1 Allocating and Deallocating Split Areas

7.1.1 Necessity for split areas

Storage areas must have been allocated for the tasks, subprograms, GLB, and other shared resources used by the real-time program before its development.

To speed up processing, the real-time system allows access to various resources through the addresses where they are stored. To achieve this, the storage addresses of resources must be consistent during program execution. The allocator divides areas in the computer into user-specified areas. The system keeps track of these areas so that they are retained at the specified addresses. In the system design stage, check how much data is required against the target to which the system is applied and determine the size and position of GLB. In most real-time systems, design of data is more important than creation of individual programs, largely affecting the comprehensive performance of the system.

Allocate areas at two stages.

- (1) Define split areas (AREAs) for tasks, subprograms, and GLB.
- (2) Use svdfs further to divide AREAs for GLB into secondary partition areas (SAREAs), if necessary.

When split areas and secondary partition areas are defined by the allocator, their names, attributes, positions, sizes, and other information are recorded as area management information. The real-time program can use names defined in the information to reference or call shared resources such as GLB.

Since areas for shared resources are allocated in a fragmentary manner, they can be allocated separately from those for ordinary resources. This is useful especially when redefining allocated split areas, because the need for redefining other split areas is minimized.

7.1.2 Allocating split areas

Split areas are arranged within a predefined global area (GAREA) in accordance with their use. Table 1-18 shows the relationship between split area use and GAREA selection.

The use of the split area to be defined must be specified with the svdfa command's option at the time of split area allocation.

Table 1-18 Relationship between Split Area Use and GAREA Selection

Split area use	GAREA selection	svdfa option
Task (program)	\$TASK	-p
Read-only GLB data	\$GLBR	-gr
Read/write GLB data	\$GLBRW	-gi, -gw
Subprogram	\$IRSUB	-s

When a split area is allocated, the area adequate for a specified size is acquired within a global area. However, it is not reflected in the PCs memory until the allocated split area is downloaded.

Further, when a split area is allocated, an initial value setup file (backup file) for the real-time resource to be placed within the split area is generated.

As regards a split area for GLB without an initial value, however, no backup file will be generated.

The contents of the backup file are initialized to zero (0).

The following commands are supported to allocate split areas:

- svdfa Allocates a split area (AREA).
- svdfs Allocates a secondary partition area (SAREA).

Figure 1-22 shows a sample layout of split areas.

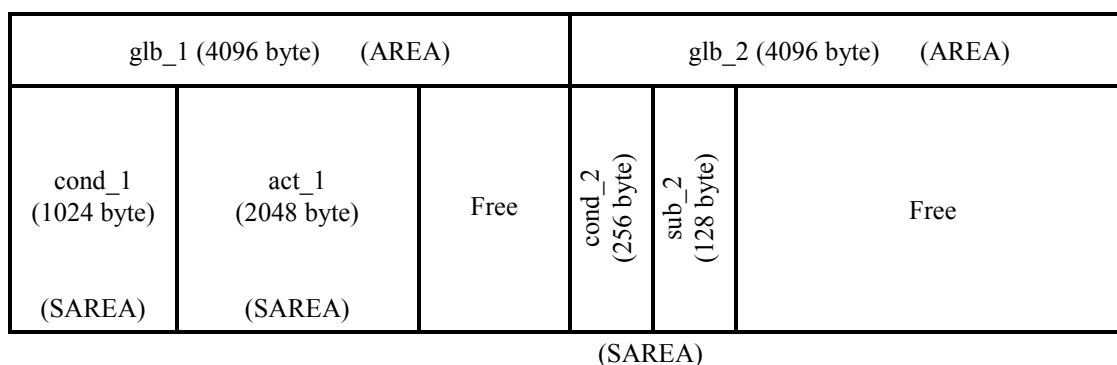


Figure 1-22 Sample Layout of Split Areas

Allocation example for the above layout

```
#svdfa glb_1 4096 -gw
#svdfs glb_1 cond_1 1024
#svdfs glb_1 act_1 2048

#svdfa glb_2 4096 -gw
#svdfs glb_2 cond_2 256
#svdfs glb_2 sub_2 128
```

7. ALLOCATOR

Allocate the split areas according to the layout in Figure 1-22.

```
$svdfa glb_1 4096 -gi
$svdfa glb_2 4096 -gi
$svdfs glb_1 cond_1 1024
$svdfs glb_1 act_1 2048
$svdfs glb_2 cond_2 256
$svdfs glb_2 sub_2 128
$svmap -g -a -e
** allocator map **                                     2002/11/19 09:15:56

site name = test2

< area >
garea/aname                raddr      size      laddr      kind  bkupfile
:
:
$GLBRW/glb_1               + s 00000000 00001000 50000000 glbi  glb_1.bkf
$GLBRW/glb_2               + s 00001000 00001000 50001000 glbi  glb_2.bkf
$GLBRW/                     00002000 000fe000 50002000

< sarea >
:
:
garea/aname/sname          raddr      size      laddr
$GLBRW/glb_1/cond_1       + s 00000000 00000400 50000000
$GLBRW/glb_1/act_1        + s 00000400 00000800 50000400
$GLBRW/glb_1/              00000c00 00000400 50000c00
$GLBRW/glb_2/cond_2       + s 00000000 00000100 50001000
$GLBRW/glb_2/sub_2        + s 00000100 00000080 50001100
$GLBRW/glb_2/              00000180 00000e80 50001180
:
:
** map output end **
$
```

In the real-time program, the names `cond_1`, `act_1`, `cond_2`, and `sub_2` above are assigned to the defined split areas so that these shared resources can be used.

```
$notepad sample.c
```

```
extern int cond_1_g[256];
extern int act_1_g[512];
extern char cond_2_g[256];
extern short sub_2_g[64];
main()
{
short abc;
cond_1_g[10] = 0;
act_1_g[20] = 30;
cond_2_g[255] = 'A';
abc = sub_2_g[0];
}
```

When changing the size or attribute of an allocated split area, take care not to make the new size smaller than the total size of the defined secondary partition areas. If the new size is made smaller than the total size of the secondary partition areas, secondary partition areas cannot be allocated in the specified split area.

7. ALLOCATOR

7.1.3 Deallocating split areas

The following commands are supported to deallocate split areas:

- svdla Deallocates a split area (AREA).
- svdls Deallocates a secondary partition area (SAREA).

For example, the glb_1 and glb_2 that were allocated in the previous section could be deleted from the GLB, as shown below. Note that these deletion operations deletes a desired split area along with any secondary partition areas in it. As for glb_2, for example, if glb_2 is deleted, secondary partition areas cond_2 and sub_2 defined in it are also deleted at the same time.

```
$svdls cond_1
$svdls act_1
$svdla glb_1
$svdla glb_2
$svmap -g
** allocator map **                               2002/04/07 15:19:13

site name = test_CP
:
< global,irglb >
:

** map output end **
$
```

7.1.4 Assigning names to GLB and VAL

The names of GLB and VAL must be unique in the system.

- The maximum number of characters in a name (the number of bytes) is limited to 14.
- Each name must begin with a letter or underscore (_).
- The second and later characters are a combination of letters, numbers, and underscores.

Real-time programs using names under these rules are also restricted by the following naming rules in C used:

Restrictions in C

- The maximum number of characters in a name (the number of bytes) is limited to 14.
- Each name must begin with a letter.
- The second and later characters are a combination of letters, numbers, and underscores.
- Each name is suffixed with the following character string. Characters in the suffix are not counted as part of the name.
 - For GLB: _g
 - For VAL: _v
- When declaring GLB or VAL, be sure to use an external variable with extern specified.

7.2 Value (VAL) Registration and Deletion

The allocator registers or deletes a constant called a value (VAL), which is common to all programs. The svdfv and svdlv commands are used for VAL registration and deletion, respectively.

CHAPTER 8 LOADER

8.1 Linking and Loading

Object modules (.obj files) created with an shc command are linked using the svload command to integrate them into a single program, which is then loaded using the GLB, IRSUB, and other items of allocator management information. The resulting load module is then written into a backup file (see Figure 1-23).

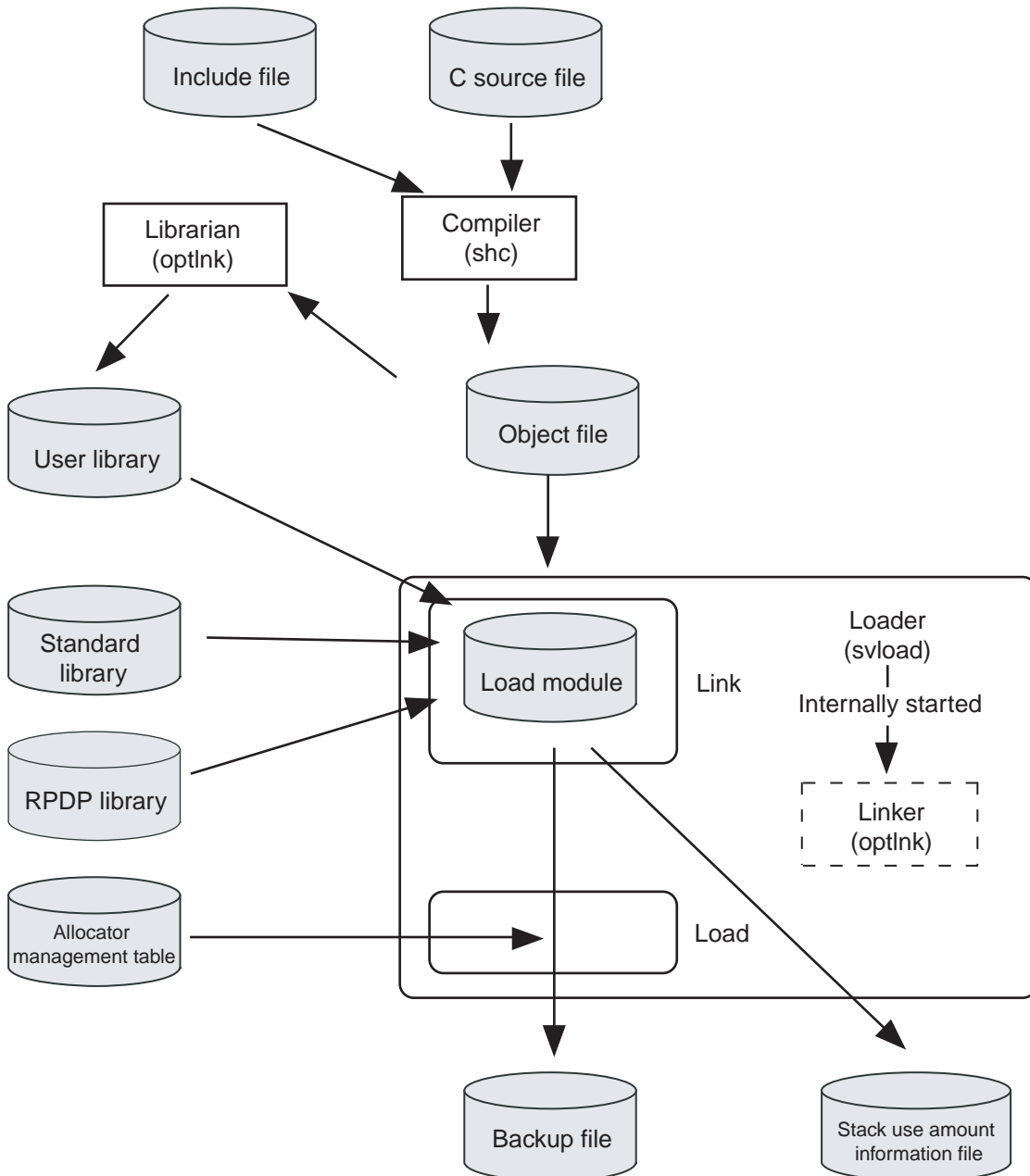


Figure 1-23 Creating a Load Module and Backup File

8.2 Loader Operating Environment

As regards the program to be entered into the loader, ensure that the load module resulting from linkage in the loader satisfies the conditions set forth in Table 1-19.

Table 1-19 Conditions for Load Module

Option	Load module		
	TEXT	DATA	BSS
Program registration	>0	—	—
Subprogram/builtin subroutine registration	>0	—	—(*)
Data registration	—	>0	—

TEXT: Executable portion

DATA: Data with initial value

BSS: Area without initial value

—: Processable when size = 0 or > 0.

> 0: Error except when size > 0

(*) The subprogram's BSS division is write-protected.

Ensure that the subprogram does not have BSS.

8. LOADER

Figure 1-24 shows the structure of a load module that is generated by the loader.

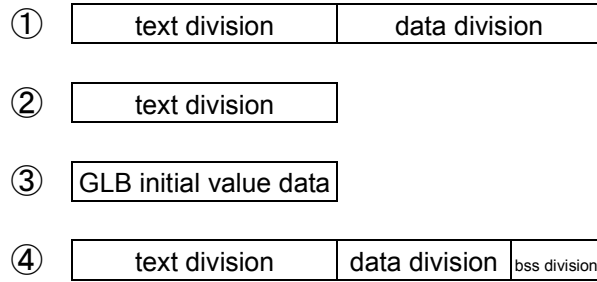


Figure 1-24 Load Module Structure

<Explanation of Figure 1-24>

① is a load module of a program or subprogram, having a text division and a data division. This load module can be loaded as a program or subprogram.

② is a load module of a program or subprogram, having only a test division. It can be loaded as ①.

③ is a load module of a GLB initial value setting program. It can be loaded as data.

④ is a load module of a program having a text division, a data division, and a bss division. It can be loaded as a program. Ensure that a subprogram does not have a bss division.

(1) Loader processing

The loader's loading process is explained below with reference to load module structures ③ and ④ shown in Figure 1-24.

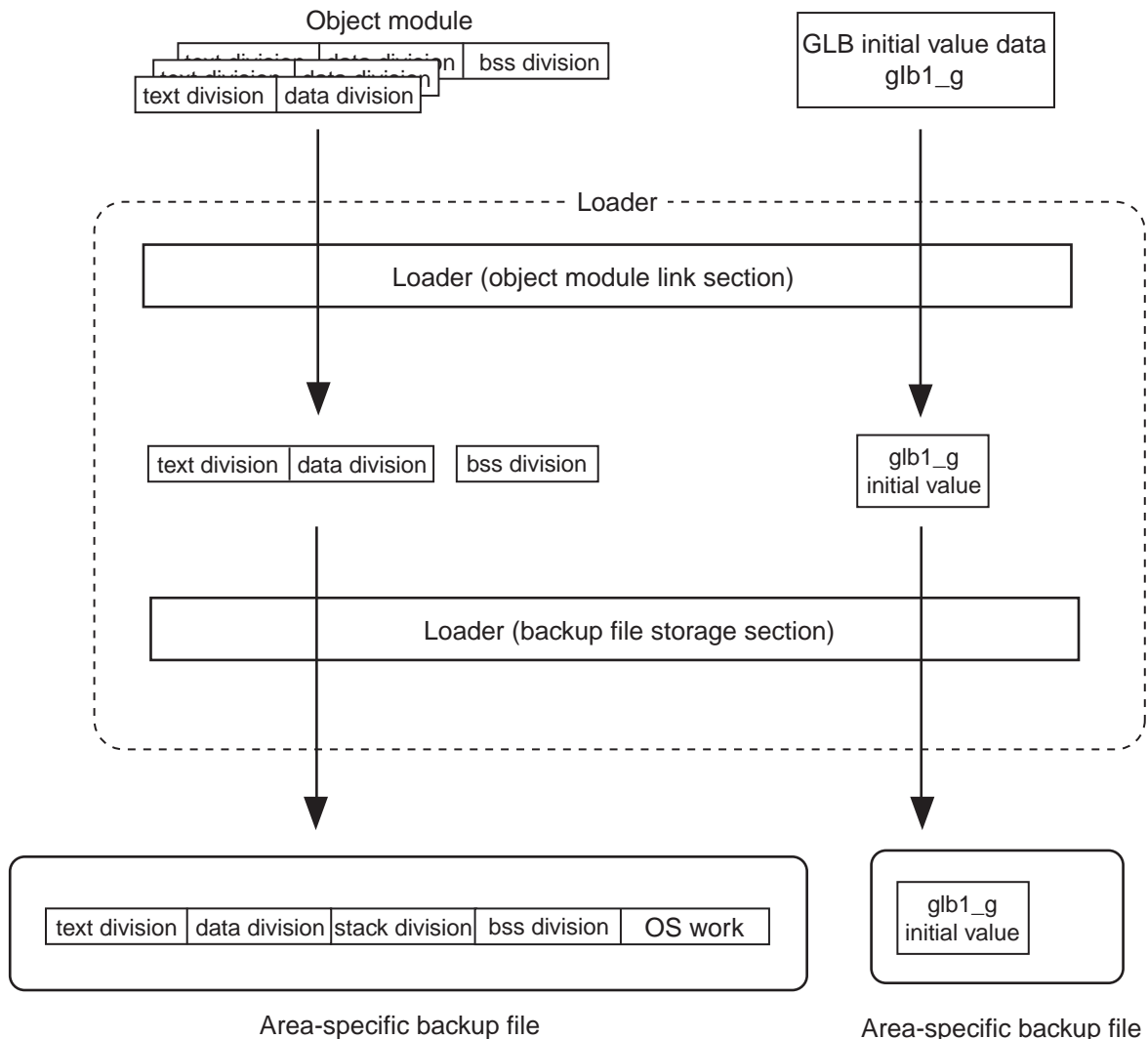


Figure 1-25 Loading processing

<Explanation of Figure 1-25>

- ① The global initial value data section is loaded into a place corresponding to a secondary partition area that is registered in the allocator-managed table by svdfs.
- ② The program is loaded, as a load module, into an area that is specified by a loader command. The loader is stored in a backup file with a stack area specified by the load module and the OS work attached.

8. LOADER

(2) Uniqueness of name

Between the system and user, the same name must not appear as a program name, subprogram name, built-in subroutine name, global name or value name.

(3) system/user external reference check

User's information cannot be referenced from the system. System subprograms can be only referenced from the user. Table 1-20 shows the combinations that can be referenced.

Table 1-20 External Reference Combinations

Referenced \ Referencing		Subprogram		Global		Value	
		S	U	S	U	S	U
Program	S	√	nr	√	nr	√	nr
	U	√	√	nr	√	nr	√
Subprogram	S	√	nr	√	nr	√	nr
	U	√	√	nr	√	nr	√
Global	S	√ (*1)	nr	√ (*2)	nr	√ (*3)	nr
	U	√	√ (*1)	nr	√ (*2)	nr	√ (*3)

√: Can be referenced S: System nr: Cannot be referenced U: User

(*1) An IRSUB number is embedded in the global area. Therefore, the subprogram must be an IRSUB that is built.

(*2) The referenced global address is embedded in the global area.

(*3) The VAL value is embedded in the global area.

8.3 Search Path of Libraries

The loader's library search path (library search sequence specified by the -l option) agrees with the input file search sequence for the optimization linkage editor in the shc compiler package.

The input file search sequence for the optimization linkage editor is as indicated below:

- (1) Current directory
- (2) Directory specified by the environment variable HLNK_DIR

For the environment variable HLNK_DIR, a plurality of paths can be set. To specify two or more paths, separate them with semicolons.

8.4 Notes on Linking and Loading

When linking and loading real-time programs, make sure that the GLB and VAL used in them have been allocated. When IRSUBs are used, make sure that they have been built.

CHAPTER 9 BUILDER

9.1 Registering and Deleting a Task

9.1.1 About the task

A load module loaded in by the loader (svload) can be created as a task by svctask of the builder so that it readies for operation. svctask creates task information that specifies a task name, task number, task execution level, and other parameters. svctask then combines that task information with the task information managed by the operating system.

9.1.2 Registering a task

Below is an example of registering a program as a task.

```

$ svdfa areal 0x3000 -p
$ shc sample.c
$ svload +P -w 1024 1024 -a areal -o sample sample.obj
$ svmap -p -t
** allocator map **                               2002/04/07 15:20:38

site name = test_CP

< task-program >
tn tname          tnox rmtn lvl  sp    pname          st mtn texttop
lastaddr tsize  dsize  ssize (part ) bsize  extra  oswork

:

                                sample      + s ls 0001 30000000
30003000 000068 000000 000400(000400) 000000 000000 001000

:
** map output end **
$ svctask sample sample_1 10 -l 25
$ svmap sample_1 -t
** allocator map **                               2002/04/07 15:21:38

site name = test_CP

< task-program >
tn tname          tnox rmtn lvl  sp    pname          st mtn texttop
lastaddr tsize  dsize  ssize (part ) bsize  extra  oswork

:

    10 sample_1      + u 000a 0001 19 30002000 sample      + s cs 0001 30000000
30003000 000068 000000 000400(000400) 000000 000000 001000

:
** map output end **
$

```

“sample” specified in svctask is the name of an load module to be used as a resource for the task. When the task is started, the specified load module is executed as the main program of the task.

In this example, the priority of the task is set to 25, and the task number is set to 10.

After the task is created, use the task number or task name to identify the task.

9.1.3 Deleting a task

To delete a registered task, use `svdtask` with its task name specified.

Below is an example of deleting a registered task.

```
$ svdtask sample_1
$ svmap sample_1 -t
** allocator map **                               2002/04/07 15:21:09

  site name = test_CP

< task-program >

** map output end **
  svmap : Specified name is undefined ( sample_1 )
$ svdload sample +P
$ svdla area1
$
```


9. BUILDER

9.2 Registering and Deleting a Resident Subprogram

9.2.1 About the indirect link subprogram (IRSUB)

An indirect link subprogram does not require task reregistration even when it is replaced after task registration.

An indirect link subprogram is created in the same way as a subroutine of a program. Unlike a subroutine, an indirect link subprogram is shared by multiple tasks, so it must be re-entrant. For this reason, do not declare static variables in the indirect link subprogram.

When an indirect link subprogram is registered again, only the address corresponding to the registered number is changed. Tasks need not be registered again even if they were registered before the re-registration of indirect link subprograms.

9.2.2 Registering an indirect link subprogram (IRSUB)

After loading an executable module using the loader (svload), register it with the builder (svbuild) as an indirect link subprogram (IRSUB). Below is an example of registering a program as an indirect link subprogram.

Create an indirect link subprogram. Use a notepad or another editor.

```
$notepad sub_a.c
```

```
sub_a()
{
    return;
}
```

Compile the indirect link subprogram.

```
$shc sub_a.c
```

Register the indirect link subprogram in the split area named areal using the registration number 10.

```
$ svdfa areal 4096 -s
$ svload +I -a areal -o sub_a sub_a.obj -w 0
$ svbuild sub_a -ir -e 10
$ svmap sub_a -s -ir
```

```
** allocator map **
```

2002/04/07 15:23:15

```
site name = test_CP
```

```
< IRSUB >
```

irno	entname	st	laddr	subname	offset	texttop	bsslast
tsize	dsize	bsize	extra	ssize (part)			
10	sub_a		+ s	ib 60000000	sub_a	000000	+ s 60000000 60000004
000004	000000	000000	000000	000000	000000	(000000)	

```
** map output end **
```

9.2.3 Deleting an indirect link subprogram (IRSUB)

To delete a registered indirect link subprogram (IRSUB), use `svdbuild`. To `svdbuild`, specify the name of the subprogram to be deleted and the `-ir` option.

Below is an example of deleting a registered indirect link subprogram.

```
$ svdbuild sub_a -ir
$ svdload sub_a +I
$ svmap -s -ir
** allocator map **                                2002/04/07 15:22:20

site name = test_CP

< IRSUB >
** map output end **
$
```

9.3 Registering and Deleting a Built-in Subroutine

9.3.1 About the built-in routine

This section explains the built-in subroutine. In case of a hardware-detected exception or software-detected event, the built-in subroutine can be incorporated into the system as the user's original processing.

The built-in subroutine is loaded in by the loader (svload) and incorporated in by the builder (svbuild) as part of event handling programs supported by the operating system.

In this system, some entry points where to incorporate built-in subroutines are provided. Different points are associated with different events. When incorporating built-in subroutines, select entry points according to the events by which their processing is initiated.

Up to four built-in subroutines can be registered for incorporation at each entry point. When an event occurs, the registered built-in subroutines are called out and executed sequentially in the ascending order of their entry numbers. Entry numbers specify the order in which registered built-in subroutines are executed.

9.3.2 Registering a built-in subroutine

Below is an example of registering a built-in subroutine. In this example, the program named uabs_usr is a subroutine that is executed when a task is aborted. Register the subroutine at the entry point having the point name ABS. To link and load the built-in subroutine, use the loader (svload) with the +U option specified.

```

$ shc uabs_usr.c -fpscv=safe

Register a built-in subroutine.

$ svdfa area1 4096 -s
$ svload +U -a area1 -o uabs_usr uabs_usr.obj -w 512
$ svbuild uabs_usr ABS 1 -ul
$ svmap -s -ul
** allocator map **                               2002/04/07 15:26:41

site name = test_CP

< ULSUB >
pnt typ ent subname          texttop bsslast tsize dsize bsize
extra ssize (part )
abs  os   1 + uabs_usr      + s 60001000 60001004 000004 000000 000000
000000 000200(000200)

** map output end **
$

```

9.3.3 Deleting a built-in subroutine

To delete a registered built-in subroutine, use `svdbuild`. To `svdbuild`, specify the name of the subroutine to be deleted, the entry point name, and the `-ul` option.

Below is an example of deleting a registered built-in subroutine.

```
$ svdbuild uabs_usr ABS -ul
$ svdload uabs_usr +U
$ svmap -s -ul
*** allocator map **                               2002/04/07 15:27:41

  site name = test_CP

< ULSUB >

** map output end **
$
```

CHAPTER 10 MAP

10.1 Purpose of Displaying Allocator Management Table Information

Allocator management table information is displayed in order to help the user pursue real-time program development smoothly.

- Information on the storage areas for tasks, subprograms, GLB, and other shared resources is displayed to facilitate creation of individual programs and system design development.
- Allocator management table information on the system loaded in the controller is displayed to facilitate debugging.

10.2 svmap Command Options and Displayed Information

The svmap command display format for each option is shown in “APPENDIX F DISPLAY FORMAT OF svmap.” The underlined portion within the display format is the data displayed by the svmap command.

10.2.1 Map information targeted for output

The following map information is to be output:

- (1) Map information about resources managed by the development machine
- (2) Map information about resources downloaded into the PCs

10.2.2 Description of map information output

The following items of information are to be output as the map information:

- (1) Header and footer
- (2) Global area information
- (3) Split area information
- (4) Secondary partition area information
- (5) Program information
- (6) Subprogram information
- (7) Task information
- (8) Global information
- (9) VAL information
- (10) IRSUB entry information
- (11) IRGLB entry information
- (12) ULSUB entry information
- (13) Information about the physical memory's free space

10.2.3 Map information output forms

The map information can be output in the following forms:

- (1) Hierarchical map output
- (2) Listing in the order of addresses
- (3) Listing in the order of names
- (4) Listing in the order of numbers
- (5) Specified name output

The hierarchical map output form is used to hierarchically output the map information about resources arranged in a logical space on an individual global or split area basis. The listing forms are used to output specified information in the order of addresses, names, or numbers. Further, the name of a resource can be specified to output the information about that name.

Table 1-21 shows the combinations of output information and selectable output forms.

Table 1-21 Combinations of Output Information and Selectable Output Forms

Output form Output description	Hierarchical output	Listing in the order of addresses	Listing in the order of names	Listing in the order of numbers	Specified name output
Global area information	√	√	ns	ns	√
Split area information	√	√	√	ns	√
Secondary partition area information	√	√	√	ns	√
Program information	ns	ns	√	ns	√
Subprogram information	ns	ns	√	√	√
Task information	ns	ns	√	√	√
Global information	ns	ns	√	√	√
VAL information	ns	ns	√	ns	√
IRSUB entry information	ns	ns	√	√	√
IRGLB entry information	ns	ns	√	√	√
ULSUB entry information	ns	ns	√	√	√

√: Can be specified ns: Cannot be specified

10.3 Logical Address Specification and Information Display by the svadm Command

The svadm command displays a name and other information for the specified logical address. An address can be specified in a command-driven or menu-driven manner. The displayed information is useful in debugging.

(1) Command-driven

svadm cogical address

When a logical address is specified in the parameter, a name and other information are displayed in the format shown below, where XXX is the data displayed by the svadm command.

- (a) Format for displaying the space of the text and data sections for the program, the GLB space and the subprogram space

```
name = XXXXXXXXX type = XXXXXXXXXXXXX raddr = XXXXXXXXX
```

- (b) Format for displaying the space for the bss and stack sections for the program

```
name = XXXXXXXXX type = XXXXXXXXXXXXX raddr = XXXXXXXXX pgmno = XXXX
```

(2) Menu-driven

When a logical address is not specified in the parameter, the entry of a logical address is asked interactively. A name and other information are displayed in the same format as the command-driven manner.

— Specification of an address in a menu-driven manner —

```
#svadm [Enter]

++ address information display start --> site(XXXXX) ++
addr : addr [Enter]

  Displayed information

addr : q [Enter]
++ address information display end ++
#
```

CHAPTER 11 STARTUP

11.1 Overview

The PCs starts up when the initial data file (backup file) for the PCs main memory (SDRAM) within the development machine is downloaded into the PCs main memory. The backup file generated by the allocator is downloaded into the main memory of a specified site within the PCs to start up the specified site.

11.2 CMU State Transitions

11.2.1 Startup procedure

Startup is achieved when the `svrpl` command is executed from the development machine. For the procedure to be performed at the development machine, see “CHAPTER 6 STARTUP,” in “PART 2 COMMAND REFERENCE.” The main functions and uses are described below:

<Functions>

The main functions of the `svrpl` command are as follows:

- Starts up a specified site.
- Downloads a backup file.
- Specifies at startup whether or not to set the CMU time.
- Specifies after completion of backup file downloading whether or not to make a user task available.
- Stops a user task at the time of `svrpl` command execution without acknowledging whether or not to make the user task unavailable.

11. STARTUP

Startup examples in which various functions are exercised are given below:

- Downloading all backup files and restarting the CMU after such a download

```
>svrpl -u test
**** svrpl start (site = test) ****
CMU status(test)=RUN
Do you stop UserTask(test) ? (yes/no)= yes
Remote loading start(site = test)
address : 0X0000d000-0X0000d0ff
.
address : 0X00b00000-0X00dfffff
....
start to modify allocator management tables

finished to modify allocator management tables
Remote loading end
Reset start(test)
**** svrpl end ****
```

- Downloading all backup files into the CMU without acknowledging whether or not to make a user task unavailable, and restarting the CMU after such a download (When the -s option is specified to skip the acknowledgment sequence, no message appears on the display.)

```
>svrpl -u test -s
```

11.2.2 CMU Control Procedure

CMU control is to be exercised by executing the `svcpuctl` command from the development machine. For the procedure to be performed at the development machine, see “CHAPTER 6 STARTUP,” in “PART 2 COMMAND REFERENCE.” The main functions and uses are described below:

<Functions>

The main functions of the `svcpuctl` command are as follows:

- Executes the `svcpuctl` command without acknowledging whether the CMU should be started/stopped (whether or not to start execution).
- Displays the site status.

11. STARTUP

Startup examples in which various functions are exercised are given below:

- Performing time setup for the CMU while at the same time issuing a run request to the CMU

```
>svcpuctl -u test -time  
status CMU(test)=STOP  
Input CMU(test) status ? (stop/run)= run  
Do you really request OK ? (yes/no)= yes  
CMU(test) = RUN
```

- Issuing a stop request to the CMU

```
>svcpuctl -u test  
status CMU(test)=RUN  
Input CMU(test) status ? (stop/run)= stop  
Do you really request OK ? (yes/no)= yes  
CMU(test) = STOP
```

- Issuing a run request to the CMU without acknowledging whether or not to stop a user task, and performing time setup for the CMU
(When the -s option is specified to skip the acknowledgment sequence, no message appears on the display.)

```
#svcpuctl -u test -s -run -time
```

- Displaying the CMU status

```
#svcpuctl -u test -ss  
status CMU(test_CP)=STOP
```

CHAPTER 12 svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT FUNCTIONS

12.1 Overview

svdebug is a command that enables debugging of programs running on the PCs from the development machine. The command has the following basic functions:

Classification	Subcommand	Function
Start or stop of a task	qu ab re ta su rs tm ct sht	Requests start of a task. Inhibits task start. Cancels inhibition of task start. Displays the task status. Suspends or suppresses task execution. Cancels suppression of task start. Starts a task cyclically. Cancels cyclic task start. Displays task cyclic starts.
Memory print or patch	md sd bs bg mcp mmv mf	Prints or patches addressed memory. Prints or patches memory specified by a name. Sets bit data. Displays bit data. Copies the contents of memory. Moves the contents of memory. Sets pattern data in memory.
System error display	el ss	Displays system errors. (executes the svelog command). Displays the system status. (executes the svcpucl command).
Current time setting or display	st gt	Sets the current time. Displays the current time.
Breakpoint setup/reset	br rb rd rr go	Sets/displays breakpoints. Resets breakpoints. Displays registers. Rewrites the contents of registers. Resumes execution from a breakpoint.
ADT setup/reset	as ac	Sets/displays the ADT. Resets the ADT.
Uploading/downloading/compare	ld sv cm	Backup file → PCs memory transfer PCs memory → backup file transfer PCs memory/backup file comparison
DHP recording function enable/disable	dr ds	Enables the DHP recording function (executes the svdhp command). Disables the DHP recording function (executes the svdhp command).
Others	si sp ps pe ver smd svadm svdhp help q !	Initializes the stack memory. Displays the amount of the stack capacity used. Starts displaying a debug statement. Finishes displaying a debug statement. Displays the version of the CMU. Prints/patches the memory for all areas. Displays information for addresses (executes the svadm command). Displays the DHP (executes the svdhp command). Lists subcommands. Terminates the debugger. Executes a command on the development machine.

12. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT FUNCTIONS

12.2 PCs Status and Subcommand Availability

While user task execution is inhibited, svdebug cannot use the following subcommands. The current status can be referenced with the svdebug ss or svcpuctl command.

Classification	Unavailable subcommands
Task start/stop	qu, ab, re, ta, su, rs, tm, ct, sht
Current time setup/display	st, gt
Breakpoint setup/reset	br, rb, rd, rr, go
ADT setup/reset	as, ac
Uploading/downloading/compare	ld (*), sv, cm
Stack initialization/use amount display	si, sp

(*) ld -g and ld -G are available.

svdebug provides the above functions in the form of subcommands. When svdebug is started, a prompt is displayed together with the target site name. From the site name, the current site to be debugged can be checked. At the prompt, enter an appropriate subcommand to begin debugging. To terminate debugging, enter subcommand q. svdebug will then terminate. An example is shown below.

```
$ svdebug
++ debugger start ++
px1> q
++ debugger end ++
```

Access to real-time resources is restricted by the owner type and user type. Be sure to set an appropriate user type before starting svdebug.

Processing of subcommands of svdebug can be suspended by a signal.

12.3 Basic Functions

(1) Subcommands to download and upload programs and data

The allocator, loader, and builder (ALB) incorporate programs and data into backup files. Some means must be used to reflect the contents of backup files in memory in the PCs. The `ld` subcommand can be used to load the contents of backup files into memory in the PCs, and the `sv` subcommand uploads the contents of main memory in the PCs to backup files. The `cm` subcommand can also be used to compare the contents of the backup file and those of corresponding main memory.

In the example below, the status of the task having task number (TN) 5 is checked and the contents of the controller's main memory are updated. GLB (named `indata`) is also updated.

```
$ svdebug
++ debugger start ++
px1> ta 5
tn=5 (0x05) tname=ta5 task state=DORMANT (0x00000000)
tcb top=0x82006280 fact=0x00000000 level=00 (27)
task top=0x02000000 stack=0x409ff000-0x40a00000

px1> ld -t ta5
address : 0x20000280-0x200002bf
address : 0x30000000-0x30000067
address : 0x20000280-0x200002bf
px1> ld -g indata
address : 0x50000400-0x50000bff
px1>
```

As described above, operation by ALB is not reflected directly in the PCs. This means that there may be a mismatch in the backup file between the development machine and the PCs.

- Main memory image files are present in the backup file on the development machine but not on the PCs.
This happens when main memory image files are registered with ALB, but they are neither loaded in the controller in batches by `svrpl` nor reflected by the `svdebug ld` subcommand.
- Main memory image files are present both in the backup file on the development machine and on the PCs.
There are two possible states: (1) Identical main memory image files are present on both the development machine and PCs, and (2) different main memory image files are present on both. State (1) is brought immediately after main memory image files registered with ALB are loaded in the controller in batches by `rplhr` or reflected by the `svdebug ld` subcommand. In state (2), main memory image files are registered and deleted repeatedly with ALB.

12. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT FUNCTIONS

- Main memory image files are not present in the backup file on the development machine but present on the PCs.
This happens when main memory image files are registered with ALB and loaded in the PCs in batches by svrpl or reflected by the svdebug ld subcommand but these files are deleted from the backup file with ALB.
- Main memory image files are not present either in the backup file on the development machine or on the PCs.
This happens when main memory image files are registered with ALB and loaded in the PCs in batches by svrpl or reflected by the svdebug ld subcommand, but these files are deleted from the backup file with ALB and also from the PCs.

The states of the development machine and PCs can be checked by the svmap command or the ld subcommand of the svdebug command. Match their states by batch loading or individual loading, as necessary. In particular, when resources are deleted from the development machine, make sure that they are also deleted from the controller.

(2) Task control subcommands

In general, a program downloaded into the PCs is started when it receives a start request from another task. For more convenience during debugging, however, tasks should be started or stopped according to an operator's instructions. svdebug supports subcommands that start or stop tasks.

In the example below, a task having the task number (TN) 5 is started while its status is being checked.

```
$ svdebug
++ debugger start ++
px1> ta 5
tn=5 (0x05) tname=ta5 task state=DORMANT (0x00000000)
tcb top=0x82006280 fact=0x00000000 level=00 (27)
task top=0x02000000 stack=0x409ff000-0x40a00000

px1> re 5
OK (0)
px1> qu 5
OK (0)
px1> ta 5
tn=5 (0x05) tname=ta5 task state=IDLE (0x00000000)
tcb top=0x82006280 fact=0x00000000 level=00 (27)
task top=0x02000000 stack=0x409ff000-0x40a00000

px1>
```

(3) Memory print and patch subcommands

svdebug supports subcommands that display or change the contents of memory. These subcommands can be used to perform test while changing the contents of memory such as GLB to desired values.

Main memory, the backup file, or both can be specified for patching or display purposes.

The example below is to start a task that sets data in GLB (named indata) containing input data, processes the data, and writes the resulting data to GLB (named outdata). Finally the task checks the processing result.

```
$ svdebug
++ debugger start ++
px1> sd
1 name : indata -g
2 storage (s,m,*) : m
3 baddr : 0
4 raddr : 0
0x50000000(0x000000) 00000000 : 0x1000
0x50000004(0x0000004) 00000000 : e
4 raddr : *1
1 name : indata -g
2 storage (s,m,*) : m
3 baddr : 0
4 raddr : 0
0x50000000(0x000000) 00001000 :
0x50000004(0x0000004) 00000000 : e
4 raddr : e
px1> qu test
OK(0)
px1> sd
1 name : outdata -g
2 storage (s,m,*) : m
3 baddr : 0
4 raddr : 0
0x50001000(0x000000) 00002000 : e
4 raddr : e
px1>
```

(4) Time setting and time display subcommands

The PCs has a built-in clock. svdebug can also be used to set a time in the built-in clock. The example below is to check the time currently set in the PCs and then change the time.

```
$ svdebug
++ debugger start ++
px1> gt
2002.04.04.20:27:32
px1> st 2002.04.04.20:30:00
OK(0)
px1>
```


12. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT FUNCTIONS

(5) System error display and system status display subcommands

During testing or debugging, it may be necessary to see the system status or the nature of a system error. S10V RPDP supports commands (svlog and svcpuctl) that display system errors or the system status. svdebug can start these commands as subcommands (el and ss) of svdebug.

(6) DHP enabling and disabling and DHP trace display subcommands

The PCs supports the debugging helper (DHP) function to facilitate debugging. The DHP function can be used to analyze the running of the operating system and user tasks. S10V RPDP supports command (svdhp) that display DHP traces, enable DHP logging, or disable DHP logging. svdebug can start these commands as subcommands of svdebug.

(7) Other subcommands

In addition to the subcommands described above, some other subcommands are also supported. The si or sp subcommand simplifies the determination of the capacity of the stack used during task execution. The svadm subcommand finds the SAREA name and IRSUB name corresponding to an address. The ! subcommand can be used to execute commands on the development machine without having to terminate svdebug. The help subcommand displays a simple explanation.

12.4 Other Functions

svdebug supports some command line options for users' convenience. The following options can be used to tailor the operation of svdebug to user's purposes.

- i Outputs keyed-in data to the output file. The output results of subcommands are not logged.
- o Outputs the date and results of operations to the specified file. This option is useful for logging operations the user has done.
- r Executes subcommand lines in the specified command file. This option is useful for performing a sequence of operations with a single operation. The file created with the -i option specified can be used as an input.
- s Directly executes a subcommand specified in this option. This option is useful for creating command procedures.
- u site Specifies the name of the site to be processed by the debugger.

12.5 Debug Support Commands

12.5.1 svelog command

The svelog command fetches error log from the error log buffer in the PCs via the network, and displays the error log sequentially from the most recent error. For each error, error log shows (1) the time the error occurred, (2) the error code (EC) indicating the cause of the error, (3) the task number (TN) of the task that caused the error, (4) the contents of the program counter (PC) that indicates the location of the error, (5) iarv0 to iarv8 and iarvn1 to iarvn8 (contents of the program around the location indicated by the PC) that indicates processing that would have been performed, and (6) other information. This command also displays register information and DHP traces to allow for more detailed analysis.

Typical operation examples are shown below.

Example 1: To display error log at the site named siteA in the simplified format

```
svelog -u siteA -f s
```

Even when the -f and s options are omitted, the same result is obtained.

Example 2: To display all error log at the site named siteA

```
svelog -u siteA -f m
```

Example 3: To display error log and DHP traces at the site named siteA

```
svelog -u siteA -f l
```

Example 4: To display error logs and DHP traces at the site named siteA and also store the information displayed on the screen in the file named abc

```
svelog -u siteA -f l -o abc
```

12.5.2 svdhp command

The svdhp command fetches the current DHP trace from the DHP trace buffer in the PCs via the network and displays the trace. The displayed DHP traces include the DHP logging time, DHP trace points, and data required for analysis. The command also stops or restarts DHP logging.

Typical operation examples are shown below.

Example 1: To display DHP traces at the site named siteA for the specified number of counts (10)

```
svdhp -u siteA +10
```

When a count (+10 in this example) is omitted, all DHP traces are displayed.

Example 2: To stop DHP logging at the site named siteA

```
svdhp -u siteA -off
```

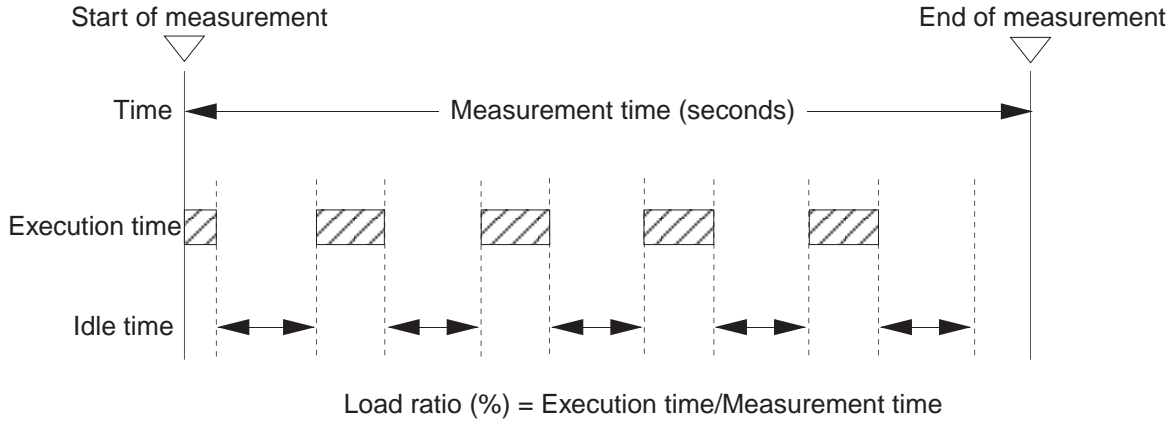
Example 3: To restart DHP logging at the site named siteA

```
svdhp -u siteA -on
```

12. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT FUNCTIONS

12.5.3 svcpunow command

The svcpunow command displays a load ratio of the PCs at the specified site. The load ratio is a percentage of the time from the start to the end of measurement during which the CMU is used for program executions (total execution time).



A typical operation example is shown below.

Example: To display the load ratio measured for 10 seconds at the site named siteA

```
svcpunow -u siteA -t 10
1996/07/03 09:56:00 SITE=siteA ** 10 second wait **
CMU(siteA) load ratio = 12.34%
```

(*1) (*2)

NOTE: Enter the command at the underlined portion.

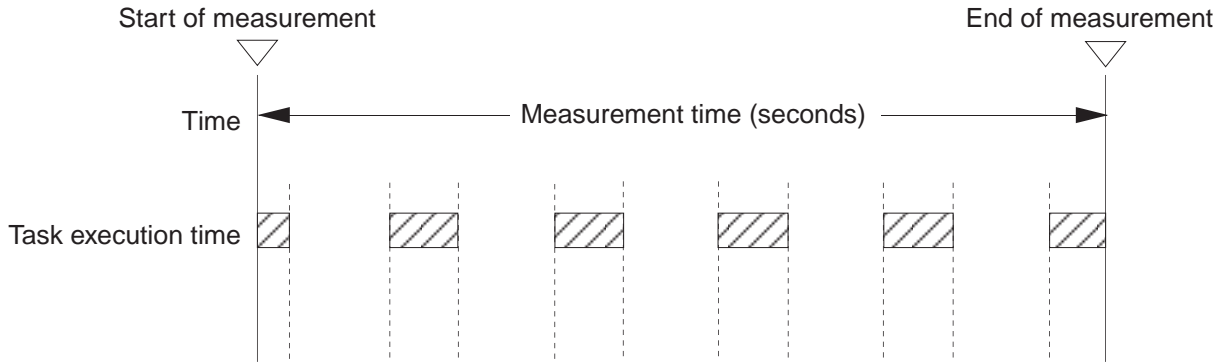
(*1) Site name

(*2) PU load ratio

When the measurement time (-t 10 in this example) is omitted, the load ratio measured for one second is displayed.

12.5.4 svtimex command

The svtimex command displays activity information for a task registered in the PCs. The activity information consists of the PU load ratio of the task, the number of executions, execution times, and an average execution time.



PU load ratio (%) of task = Total task execution time/Measurement time
 Average PU execution time of task = Total task execution time/Number of task executions

Examples of typical operations are shown below.

Example 1: To measure the activity of the following task for 10 seconds:
 Task name = taska, Task number = 123

```
#svtimex -u siteA taska -t 10
1996/07/03 13:30:24 SITE=siteA ** 10 second wait **
taska(123) load ratio=3.00% execute count=10 total time=0.030sec average time=0.003sec
```

(*1) (*2) (*3) (*4) (*5)

NOTE: Enter data at the underlined portion.

- (*1) Task name (task number)
- (*2) PU load ratio of the task
- (*3) Number of task executions
- (*4) Total task execution time
- (*5) Average PU execution time of the task

When a measurement time is not specified (i.e. “-t 10” is omitted), the load ratio is measured for one second.

12. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT FUNCTIONS

Example 2: To measure the activities of the following tasks interactively for 3600 seconds:

Task name = taska, Task number = 123

Task name = taskb, Task number = 124

Task name = task22c, Task number = 111

```
#svtimex -u siteB
SITE=siteB Task measuring period[sec] = 3600
Task name or number = taska
Task name or number = taskb
Task name or number = 111
Task name or number = [Enter]
1996/07/17 13:30:24 SITE=siteB ** 3600 second wait **
taska(123) load ratio=3.00% execute count=36000 total time=108.000sec average time=0.003sec
taskb(124) load ratio=2.50% execute count=18000 total time=90.000sec average time=0.005sec
task22c(111) load ratio=0.01% execute count=360 total time=0.360sec average time=0.001sec
```

NOTE: Enter data at the underlined portion.

PART 2 COMMAND REFERENCE

1. SYSTEM GENERATION

CHAPTER 1 SYSTEM GENERATION

NAME

svgen

SYNOPSIS

svgen fname

DESCRIPTION

svgen generates a system construction environment (site) defined within a specified system construction environment definition file, and registers it in the PCs system management file (sysadm). Further, svgen generates a user definition information input file template in the generated environment. Administrator privilege is needed for command execution.

ARGUMENT

fname Specifies a definition file that containing the definition of the system construction environment to be generated.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: The definition file was incorrect.
- 2: The system construction environment was not successfully generated.
- 3: The PCs system management file could not be updated.

System environment definition file structure

```
SUBSYS_DIR=△site-storage-directory  
SITE_NAME=△site-name
```

Keywords are indicated in bold italics. Ensure that each entry contains a keyword. The triangular mark represents one or more spaces or a tab.

- Site storage directory
Specify the site storage directory by writing its full pathname, which includes a drive letter. Ensure that the site directory name does not exceed 255 characters in length.
- Site name
Specify the site name.
Enter a site name consisting of no more than 14 characters. Single-byte alphanumeric characters, _ (underscores), and + (plus sign) can be used to enter a site name.

NAME

svconf

SYNOPSIS

svconf sitename

DESCRIPTION

svconf constructs a PCs system environment in accordance with the definition information in the user definition information input file is generated in etc\conf within the site environment. Administrator privilege is needed for command execution.

ARGUMENT

sitename Specifies the name of the site to be constructed.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: The definition file was incorrect.
- 2: The definition information output file was not successfully generated.

1. SYSTEM GENERATION

NAME

svshconf

SYNOPSIS

svshconf sitename [-f fname]

DESCRIPTION

svshconf outputs a list of definition information about a specified site.

ARGUMENT

sitename Specifies the name of the site whose definition information is to be displayed.

OPTION

-f fname Specify the name of the file to which the result is to be output.
By default, the result will be delivered to the standard output.

TERMINATION CODE

Returns one of the following termination codes:

0: Normal termination.

1: The definition information was not successfully read.

OUTPUT FORMAT

The output format is shown below:

```
<<<Show Subsystem directory>>>
  c:\users\s10v\kwsys          --- Site directory information

<<< Show Site name >>>
  test

<<< Site Information (test) >>>
                                     --- Memory information
TASK_SIZE      512KB                Servers as the area information about the site.
GLBR_SIZE      512KB
GLBW_SIZE      512KB
IRSUB_SIZE     512KB

<<< I/O Information (test) >>>      --- I/O information

Adapter Information : LANCE
SLOT 0
STATION_NO 1
IF_COUNT 1
UNO 1 IF_NUM 1 IP_ADDR 192.192.192.1 NETMASK 255.255.255.0 BRD_ADD 192.192.192.255
```

NAME

svsitecp

SYNOPSIS

svsitecp siteX1 siteY1

DESCRIPTION

svsitecp copies the PCs system on an individual site basis.
Administrator privilege is needed for command execution.

ARGUMENTS

siteX1 Specifies the copy source site name.
siteY1 Specifies the name of the copy destination site directory by entering its full
pathname, which includes a drive letter.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: The specified site name contained an unacceptable character or consisted of more than 14 characters.
- 2: The specified copy destination site name was a duplicate of an existing site name.
- 3: The specified copy destination site directory was incorrectly named. Or, you did not have the right to access it.
- 6: The site was not successfully copied.
- 7: The PCs system management file could not be updated.

1. SYSTEM GENERATION

NAME

`svsitedel`

SYNOPSIS

`svsitedel sitename`

DESCRIPTION

`svsitedel` deletes the PCs system of a specified site from the development machine. Administrator privilege is needed for command execution.

ARGUMENT

`sitename` Specifies the name of the site to be deleted.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: The site was not successfully deleted.
- 2: The PCs system management file could not be updated.

NOTE

If Windows Explorer or the like displays the targeted site directory when you execute this command for site/directory deletion, you cannot erase the site directory because the associated file is open and the command generates the following message and abnormally terminates:

“svsitedel: Can't Delete Directory (directory pathname) (EC0613)”

If the above situation is encountered, stop referencing the associated file with Windows Explorer or the like, and then achieve deletion by executing the `svsitedel` command with the `-d` option specified.

CHAPTER 2 ALLOCATOR

NAME

svdfa

SYNOPSIS

svdfa aname size [option]

DESCRIPTION

svdfa allocates a specified split area within a specified global area and generates a backup file.

ARGUMENTS

aname Name of the split area to be deleted.
 size Size of the split area to be allocated. Specify a multiple of 4,096 bytes. Otherwise, a warning message is displayed and the specified size is rounded up to the nearest multiple of 4,096 bytes.

OPTIONS

-p An area to store of the task is allocated.
 -s An area to store a subprogram is allocated.
 -gi Allocates a GLB area with an initial value in a read/write global area.
 -gw Allocates a GLB area without an initial value in a read/write global area.
 -gr Allocates a GLB area with an initial value in a read-only global area.
 -S Specifies the use of the system's access rights. If this option is not specified, the preselected environment variable RSUTYP takes effect. By default, this variable represents the user's access rights (RSUTYP=u).
 -u site Specifies the site name (site) to be processed by the allocator. If this option is not specified, the site name is specified according to the preselected environment variable RSSITE, which has no default value.
 -f adr In adr, specify the address of the split area to be allocated, relative to the beginning of the global area. The address must be a multiple of 4,096. Otherwise, a warning message is displayed and the specified address is rounded up to the nearest multiple of 4,096. When this option is omitted, an area is automatically allocated, that is, the first free area found is allocated.

NOTES

- When none of -p, -s, -gi, -gw, and -gr is specified, -p is assumed.
- When a secondary partition area is allocated with the svdfs command in the split area allocated with -gi or -gr specified, the secondary partition area is zero-cleared. Table 2-1 shows option combinations.

2. ALLOCATOR

Table 2-1 Combinations of svdfa Options

Parameter	Area type					
	Task	Sub-program	Read/write GLB with an initial value	Read/write GLB without an initial value	Read-only GLB with an initial value	
aname	√	√	√	√	√	
size	√	√	√	√	√	
Options	-p	sp (default)	ns	ns	ns	
	-s	ns	√	ns	ns	
	-gi	ns	ns	√	ns	
	-gw	ns	ns	ns	√	ns
	-gr	ns	ns	ns	ns	√
	-S	sp	sp	sp	sp	sp
	-u site	sp	sp	sp	sp	sp
-f adr	sp	sp	sp	sp	sp	

√: Required sp: Can be specified ns: Cannot be specified

- The table below shows the relationships between the user types and the owner types for areas to be allocated:

User type	Owner type for area to be allocated
System	System
User	User

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: An improperly specified option was encountered.
- 2: The GAREA did not have an adequate free space.
- 3: The maximum registration count was exceeded.
- 4: The management table could not be updated.

NAME

svdla

SYNOPSIS

svdla aname [option]

DESCRIPTION

svdla deletes a split area allocated by svdfa and a backup file.

ARGUMENT

aname Name of the split area to be deleted.

OPTIONS

- s This option specifies that the access right type is “system.” When this option is omitted, the default (set in the RSUTYP environment variable in advance) is used.
- u site In site, specify the name of the site to be handled by the allocator. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

NOTES

- When the specified split area includes a secondary partition area (read/write global area or read-only global area), the secondary partition area is also deleted.
- The table below shows the relationships between the user types and the owner types for areas to be deleted:

User type	Owner type for area to be deleted	
	System	User
System	√	√
User	nd	√

√: Can be deleted nd: Cannot be deleted

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: An improperly specified option was encountered.
- 2: The user cannot delete a system area.
- 3: The management table could not be updated.

2. ALLOCATOR

NAME

svdfs

SYNOPSIS

svdfs aname sname size [option]

DESCRIPTION

The svdfs command allocates a global secondary partition area in the split area allocated with svdfa.

Initializes the allocated area to zero (0).

ARGUMENTS

aname Specifies the name of the split area to be divided into secondary partition areas.
sname External name (global name) of the secondary partition area to be allocated.
size Size of the secondary partition area to be allocated (in bytes).

OPTIONS

-S This option specifies that the access right type is "system." When this option is omitted, the default (set in the RSUTYP environment variable in advance) is used.

-u site In site, specify the name of the site to be handled by the allocator. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

-l adr In adr, specify the address of the secondary partition area to be allocated, relative to the beginning of the split area. The address must be a multiple of 4. Otherwise, a warning message is displayed and the specified address is rounded up to the nearest multiple of 4. When this option is omitted, an area is automatically allocated, that is, the first free area found is allocated.

-a align In align, specify the number of alignments with a value of 2 raised to the nth power ($0 \leq n \leq 12$). The secondary partition area is allocated based on this number. The default is 2.

-t svtype Effects alignment in accordance with the data type specified by svtype. Table 2-2 shows the relationship between the value specified for stype and the alignment count.

-e idxnum When using the allocated secondary partition area as an indirect link global area, specify the entry number to be assigned to the secondary partition area in idxnum. When this option is omitted, no entry number is assigned. idxnum can be specified in the range of 1 to the value of the IRG_MAXNUM.

NOTES

- When secondary partition areas allocated without specifying the -e option are used as indirect link global areas, svirglb command can be used to assign an entry number to the desired secondary partition area.
- Determine the number of alignments according to the size declared for the data.
- The -l, -a, -t options are mutually exclusive. When they are specified together, an error occurs.
- Table 2-3 shows the option combinations that can be specified.

Table 2-2 Relationship between the Value Specified for stype and the Alignment Count

svtype	Data type	Alignment count
1	char	0 (1 byte)
2	short	1 (2 bytes)
3	long	2 (4 bytes)
4	struct	3 (8 bytes)
5	float	2 (4 bytes)
6	double	3 (8 bytes)
7	long double	4 (16 bytes)

Table 2-3 Combinations of svdfs Options

Parameter	Area type					
	Task	Sub program	Read/write GLB with an initial value	Read/write GLB without an initial value	Read-only GLB with an initial value	
aname	ns	ns	√	√	√	
sname	ns	ns	√	√	√	
size	ns	ns	√	√	√	
Options	-S	ns	ns	sp	sp	sp
	-u site	ns	ns	sp	sp	sp
	-l adr	ns	ns	sp	sp	sp
	-a align	ns	ns	sp	sp	sp
	-t svtype	ns	ns	sp	sp	sp
	-e index	ns	ns	sp	sp	sp

√: Required sp: Can be specified ns: Cannot be specified

- The table below shows the relationships between the user types and the owner types for secondary partition areas to be allocated:

User type	Owner type for secondary partition area	
	System	User
System	√ (System)	√ (User)
User	na	√ (User)

√: Can be allocated na: Cannot be allocated
The type in parentheses is the owner type of the secondary partition area allocated.

2. ALLOCATOR

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: An improperly specified option was encountered.
- 2: The user cannot create an sarea within the system area.
- 3: The maximum registration count was exceeded.
- 4: The management table could not be updated.

NAME

svdls

SYNOPSIS

svdls aname [option]

DESCRIPTION

The svdls command deleted the secondary partition area allocated with svdfs.

ARGUMENT

aname External name of the secondary partition area to be deleted.

OPTIONS

- s This option specifies that the access right type is “system.” When this option is omitted, the default (set in the RSUTYP environment variable in advance) is used.
- u site In site, specify the name of the site to be handled by the allocator. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

NOTE

The table below shows the relationships between the user types and the owner types for areas to be deleted:

User type	Owner type for area to be deleted	
	System	User
System	√	√
User	nd	√

√: Can be deleted nd: Cannot be deleted

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: An improperly specified option was encountered.
- 2: The user cannot delete a system area.
- 3: The management table could not be updated.

2. ALLOCATOR

NAME

svdfv

SYNOPSIS

svdfv *ename* *value* [*option*]

DESCRIPTION

The svdfv command registers information on external reference of value information.

ARGUMENTS

ename External name to be registered.
value Value of the external name ($-2^{31} \leq \text{value} \leq 2^{31} - 1$)

OPTIONS

-s This option specifies that the access right type is “system.” When this option is omitted, the default (set in the RSUTYP environment variable in advance) is used.
-u *site* In *site*, specify the name of the site to be handled by the allocator. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

NOTES

- The value specified in *value* is handled as integer-type data. When the value is not within the range of -2^{31} to one less than 2^{31} , an error occurs.
- The value used during execution depends on the type of the value name (specified in *ename*) in the language used.
- The table below shows the relationships between the user types and the owner types for value areas to be allocated:

User type	Owner type for value area
System	System
User	User

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: An improperly specified option was encountered.
- 2: The maximum registration count was exceeded.
- 3: The management table could not be updated.

NAME

svdlv

SYNOPSISsvdlv *ename* [*option*]**DESCRIPTION**

The svdlv command deletes external reference information registered with dfvhr.

ARGUMENT

ename Specifies the external name to be deleted.

OPTIONS

- s This option specifies that the access right type is “system.” When this option is omitted, the default (set in the RSUTYP environment variable in advance) is used.
- u *site* In *site*, specify the name of the site to be handled by the allocator. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

NOTE

The relationship between the user type and deletion value owner type.

User type	Deletion value owner type	
	System	User
System	√	√
User	nd	√

√: Can be deleted nd: Cannot be deleted

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: An improperly specified option was encountered.
- 2: The user cannot delete the system's VAL.
- 3: The management table could not be updated.

CHAPTER 3 LOADER

NAME

svload

SYNOPSIS

svload [option] file

DESCRIPTION

The svload command links the specified object file or library, registers it in the development environment using a name representing a program, subprogram, or data, and finally stores the file or library in the backup file.

ARGUMENT

file Specify one or more object files or libraries to be linked.

OPTIONS

- S This option specifies that processing mode is “system.” When this option is omitted, the default (set in the RSUTYP environment variable in advance) is used.
- u site In site, specify the name of the site to be handled by the loader. When this option is omitted, the site set in the RSSITE environment variable is used.
- C n In n, specify the first address of the area for storing the program or subprogram. The address must be a multiple of 64. Otherwise, a warning message is displayed and the specified address is rounded up to the nearest multiple of 64.
- p n In n, specify the relative address in the area at which to start loading. This option is valid for a program or subprogram. When this option is omitted, the file is automatically registered. This option and -C option are mutually exclusive. Do not specify them together. When specifying the address, ensure that it is a multiple of 4096 for a program or a multiple of 32 for a subprogram. When the specified address is other than a multiple of 4096 or 32, the system displays a warning message and raises the specified value to a multiple of 4096 or 32.
- a area In area, specify an area to which to load the program or subprogram. When the file is to be loaded is a program or subprogram, this option must be given.
- +P This option specifies that the file is loaded as a program (task).
- +I This option specifies that the file is loaded as an indirect link subroutine (IRSUB).
- +U This option specifies that the file is loaded as a built-in subroutine (ULSUB).
- +D This option specifies that the file is loaded as global data. The data type depends on the attribute of the split area where the data is stored.
- M n A multi-task program is created. The value n represents the number of tasks in the multi-task program. Ensure that the value n is between 2 and 128.

- m n [n . .] This option specifies multi-entry loading of an IRSUB. In n, specify an entry name as an entry point. This option is valid only when +I is specified.
- Z The sizes of the text, data, bss, and stack sections of the load module are output. This option does not register a program.
If this option is specified in conjunction with the -P option, the command generates a linkage map list without loading.
If this option is specified in conjunction with the -w n [,m] option, the command makes it possible to decide on the necessity for caller side reloading (stack size enlargement) when the stack size used by the called IRSUB increases. In such an instance, specify the values n and m so that they are equal to those specified for the last registration.
- d Does not delete the load module file after storage in the backup file. The load module file is generated under PGM, SUB, or DATA within the site directory.
- s Generates a stack use amount information file. The stack use amount information file is generated in the PGM/SUB directory within the site directory.
- l lib In lib, specify the library to be linked. libcpms.lib and libsh4nbmdn.lib are automatically linked.
- P [file] Outputs the program's linkage map list. If "file" is not specified, the command generates a file under PGM, SUB, or DATA within the site directory in such a manner that the resulting file name is formulated by adding ".map" to the end of the name of the program or subprogram to be loaded. For initial value data loading, the file name is formulated by adding ".map" to the end of the sarea name of the beginning of the loaded data.
- o obj In obj, specify the name of the program to be created. When the specified program name represents a subprogram, the specified name is used as the subprogram name. Make sure that the subprogram name and the function name in the program are identical.
- U sym During linking, the symbol specified in sym is handled as undefined.
- E n Achieves storage with a redundant byte count (n) taken into account at the time of program/subprogram linkage. This option is useful when the program or subprogram is likely to be replaced with a larger one in the future.
- r Checks whether a program/subprogram can replace the contents of a specified area. When specifying the storage address, ensure that it is identical with the program/subprogram address prevailing before replacement.

3. LOADER

-w n [m]

Specifies the stack area size in bytes.

This option is mandatory for a program/subprogram.

In the “n” position, specify the size of the stack area to be used locally.

In the “m” position, specify the stack size to be allocated. If “m” is not specified, the resulting stack size will be determined by adding the local stack size (n) to the maximum stack size for use by the called IRSUB.

If the stack size (m) to be allocated is smaller than the value n plus the called IRSUB’s maximum stack size, the command generates a warning message.

Ensure that the values n and m are between 0 and 8388608 (0x800000) and a multiple of 8. If either of the specified values is other than a multiple of 8, the command displays a warning message and raises it to a multiple of 8 for processing purposes.

The maximum stack size value for a built-in subroutine is 512 bytes.

LIBRARY SEARCH PATH

The loader's library search path (library search sequence specified by the -l option) agrees with the input file search sequence for the optimization linkage editor in the shc compiler package.

The input file search sequence for the optimization linkage editor is as indicated below:

- (1) Current directory
- (2) Directory specified by the environment variable HLNK_DIR

For the environment variable HLNK_DIR, a plurality of paths can be set. To specify two or more paths, separate them with semicolons.

STACK SIZE

When the program uses a stack area, specify its size.

SYSTEM/USER EXTERNAL REFERENCE CHECK

The system cannot reference the user information. The user can reference the system's subprograms only. The table below shows combinations that can be referenced.

Referenced Referencing		Subprogram		Global		Value	
		S	U	S	U	S	U
Program	S	√	nr	√	nr	√	nr
	U	√	√	nr	√	nr	√
Subprogram	S	√	nr	√	nr	√	nr
	U	√	√	nr	√	nr	√
Global	S	√	nr	√	nr	√	nr
	U	√	√	nr	√	nr	√

√: Can be referenced S: System nr: Cannot be referenced U: User

NOTE: When global data references a subprogram, an indirect link table number corresponding to its name is embedded in the global data. When global data references global data, an absolute address is embedded in the global data. When global data references a value, the value is embedded in the global data.

3. LOADER

NOTES

- Since IRSUBs and multitasks are reentrant programs, they cannot have a BSS area. If they have a BSS area, a warning message appears.
- If a plurality of global data exist within the program to be loaded, the local label address solution will not be implemented. In this instance, divide the global data into a plurality of files for loading purposes.
- Task execution starts at the beginning of the program. It does not always starts with the main. When loading the program, first specify the main routine's object file.
- Option combinations are indicated below:

		-o obj	-a area	-w n	-s	-S	-u site	-C n	-p n	-M n	-d	-Z	-P file	-E n	-r	-m n	-l
Program	+P	⊙	⊙	⊙	√	√	√	√	√	√	√	√	√	√	√	ns	√
IRSUB	+I	⊙	⊙	⊙	√	√	√	√	√	ns	√	√	√	√	√	√	√
ULSUB	+U	⊙	⊙	⊙	√	√	√	√	√	ns	√	√	√	√	√	ns	√
Data	+D	⊙	ns	ns	ns	√	√	ns	ns	ns	ns	√	√	ns	ns	ns	√

⊙: Mandatory √: Optional ns: Cannot be specified

TERMINATION CODE

Returns one of the following termination codes:

0: Normal termination.

Other than 0: Abnormal termination.

STACK SIZE CALCULATION PROCEDURE

The overall stack area use amount can be determined by calculating the stack use amounts of all functions comprising the program and considering a function call.

(1) Calculating the stack area used by a function

The size of the stack area used by a function can be determined from the “frame size” in the object list generated by the compiler.

A concrete example is shown 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 ; <u>frame size=12</u>
	00000000	2FE6		MOV.L	R14,@-R15	
	00000002	4F22		STS.L	PR,@-R15	

In the above example, the size of the stack area used by function h is 12 bytes, which is the “frame size” value under “COMMENT” in the object list.

3. LOADER

(2) Calculating the overall stack size from a function call

The size of the stack area to be used can be calculated from a function call.

Figure 2-1 shows how to calculate the stack use amount from a function call.

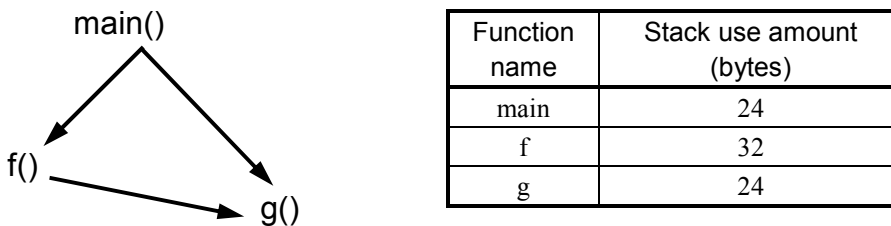


Figure 2-1 Function Call and Stack Use Amount

When function g is called via function f as shown above, the stack area size is as indicated in Table 2-4.

Table 2-4 Stack Size Calculation Examples

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

As indicated above, it is necessary to calculate the stack size for a function at the deepest call level and allocate at least the stack area adequate for its maximum size.

When using a standard library function, it is also necessary to consider the stack size used by the library function. For stack sizes used by standard library functions, see “APPENDIX H LIST OF STACK SIZES FOR LIBRARY USE.”

The stack size for a recursively called function must be calculated by multiplying the function’s stack size by the maximum recursive call count.

Even when no library function is used with a source program, an execution routine required for program execution may be linked. The stack size used by the execution routine can be confirmed with a stack analysis tool indicated on the next page, which describes the stack use amount determination procedure.

STACK USE AMOUNT CONFIRMATION PROCEDURE

When the -s option is specified at the time of program/subprogram loading, the stack use amount information file can be generated.

The amount of overall stack use by a program/subprogram can be determined by analyzing the stack use amount information file generated by the loader with the stack analysis tool supplied with the compiler package.

- Generating the stack use amount information file

When the -s option is specified at the time of program/subprogram loading, the system generates the stack use amount information file.

The stack use amount information file is generated in the PGM/SUB directory within the site directory. The name of this file is formulated by attaching “_sni” to the program/subprogram name.

(Example)

```
svload +P -o pgm01 -a tskarea -w 4096 pgm01.obj -s
```

In the above example, site-directory\PGM\pgm01_sni is generated.

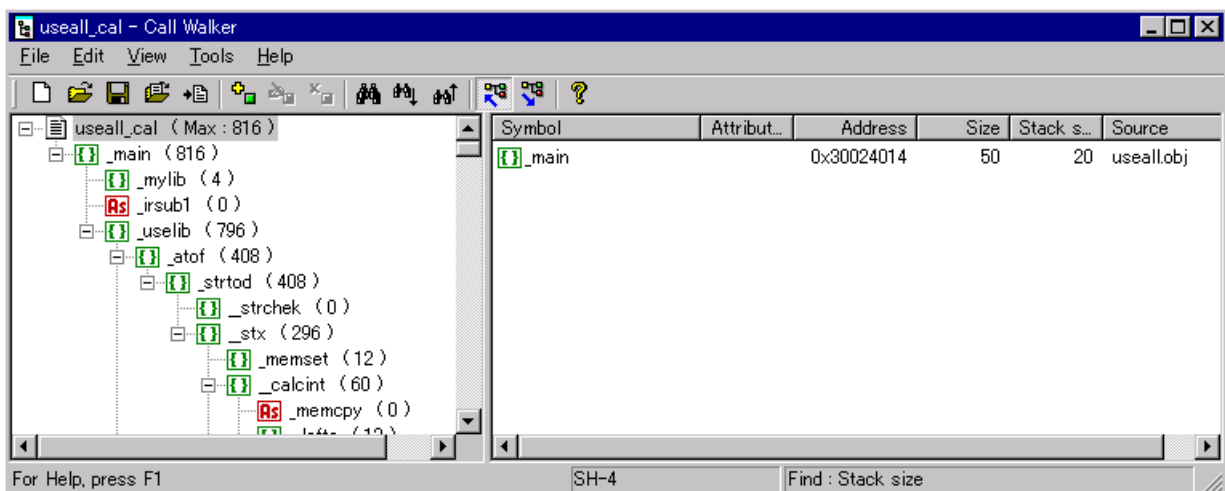
- Using the stack analysis tool

You can start the stack analysis tool to display the amount of stack use by a program/subprogram by performing the following procedure.

For detailed description of the stack analysis tool use, see the documentation supplied with the compiler package and Help on the stack analysis tool.

- ① Click Windows®’s [Start] button, point to [Programs], point to [Hitachi Embedded Workshop 2], and choose [Hitachi Call Walker] to start the stack analysis tool.
- ② From the [File] menu of the stack analysis tool, choose [Import Stack file...]. A dialog box then opens. In the “File” field of the dialog box, specify the stack use amount information file generated by the loader, and then click the button.

- Stack analysis tool display example



3. LOADER

- Precautions for analyzing the amount of stack use by a loaded program/subprogram
When used for stack use amount calculation, the stack analysis tool indicates that the stack size of a program/subprogram written by the assembler is 0 bytes. Therefore, when you use the stack analysis tool for determining the amount of stack use by a program or subprogram loaded by RPDP, you must observe the following precautions:
 - memcpy() stack size
In a loaded program/subprogram, the CPMS library's memcpy() function is linked instead of the C standard library's memcpy() function. The stack analysis tool indicates that the stack size used by the CPMS library's memcpy() function is 0. In reality, however, the CPMS library's memcpy() function uses 28 bytes of stack. From the stack analysis tool's [Edit] menu, use the [Modify] command to change the memcpy() function's stack size to 28 bytes, and then recalculate the amount of stack use.
To determine whether the loaded program/subprogram uses memcpy(), search for "memcpy" using the stack analysis tool's search function.
 - IRSUB stack size
When an IRSUB is called from a loaded program/subprogram, the stack analysis tool calculates that the amount of stack used by the IRSUB is 0 bytes. To calculate the stack use amount including the stack size used by the called IRSUB, use the [Modify] command on the stack analysis tool's [Edit] menu to change the stack size and recalculate the amount of stack use as is the case with memcpy().
The stack size used by an IRSUB can be determined by generating a stack use amount information file at the time of IRSUB loading and making an analysis with the stack analysis tool.
When you determine the stack size used by the local specified for the loader (excluding the called IRSUB), you do not have to recalculate the IRSUB stack size.
 - Stack size of a program/subprogram written by the assembler
To determine the stack size used by a program/subprogram written by the assembler, use the same procedure as described above. More specifically, use the [Modify] command on the stack analysis tool's [Edit] menu to change the stack size and recalculate the amount of stack use.

(3) Specifying the stack size with svload

To specify the stack size with svload, observe the following instructions:

-wnm

n Stack size to be used by the program/subprogram to be loaded.

m Size including the stack size to be used by an IRSUB called from the program/subprogram to be loaded. When this value is specified at the time of program loading, stack allocation takes place in accordance with the specified size. If this value is not specified, on the other hand, stack allocation takes place in accordance with the value n plus the value of the stack size used by the called IRSUB.

If the specified m value is smaller than the value n plus the value of the stack size used by the IRSUB, the loader outputs the following warning message.:

```
Warning: Stack size (name) = xxxx (zzzz) byte [Max refered (sname) size = yyyy byte] Err
```

name	Name of the program or subprogram to be loaded.
xxxx	Size including the stack size to be used by the called IRSUB.
zzzz	Stack size specified for m.
sname	Name of the called IRSUB (having the maximum stack size).
yyyy	Stack size used by sname.

To calculate the stack size used by a program/subprogram, follow the procedure set forth under “(2) Calculating the overall stack size from a function call.”

A typical procedure for specifying the stack size with svload is described below with reference to a call-related program shown in Figure 2-1, “Function Call and Stack Use Amount.”

- When no IRSUB is used

```
main Program's main
f    ISUB
g    ISUB
```

In the above case, specify as indicated below:

```
svload +P -o main ..... -w 80 4096 .....
```

Since the maximum stack size used by the program to be loaded is 80 bytes, specify 80 as the value n.

For m, specify the size of the stack to be allocated. If the value m is not specified, the stack size is set at 80 bytes.

To provide for an increase in the stack size for use in program modification or the necessity for IRSUB calls, it is recommended that the specified stack size be larger than adequate.

The program's stack/BSS is positioned on a page that is separate from the text/data page. Therefore, even if the specified stack size is larger than adequate, the program size does not increase unless the 4096-byte limit is exceeded by the BSS size plus stack size.

3. LOADER

- When an IRSUB is used
main Program's main
f IRSUB
g IRSUB

In the above case, specify as indicated below:

```
svload +I -o g ..... -w 24 .....  
svload +P -o main ..... -w 56 4096 .....
```

The IRSUB and program must be loaded separately. The stack size must be specified for both the IRSUB and program.

- For IRSUB loading
Since the stack size used by the IRSUB (g) is 24 bytes, specify 24 bytes as the stack size. It is recommended that the value m be not specified. If the value m is specified, it is used as the IRSUB stack size that is to be calculated by the IRSUB caller side.
- For program loading
Since the stack size used by the program is 56 bytes, specify 56 as the value n. For m, specify a value that is not smaller than 80 bytes, which is obtained by adding the value 56 to the stack size used by the IRSUB (g), which is 24. As is the case where no IRSUB is used, it is recommended that the specified m value be greater than adequate to provide for program modification.
If the specified m value is smaller than 80 bytes, the loader outputs the following warning message:

```
Warning: Stack size (main) = 80 (m) byte [Max refered (g) size = 24 byte] Err
```

If the value m is not specified, the loader allocates 80 bytes of stack by adding the stack size used by the IRSUB (g), which is 24.

- When an IRSUB is used from another IRSUB

main Program's main

f IRSUB

g IRSUB

In the above case, specify as indicated below:

```
svload +I -o g ..... -w 24 .....
svload +I -o f ..... -w 32 .....
svload +P -o main ..... -w 24 4096 .....
```

- For IRSUB (g) loading

Since the stack size used by the IRSUB (g) is 24 bytes, specify 24 bytes as the stack size. It is recommended that the value m not be specified. If the value m is specified, it is used as the IRSUB stack size that is to be calculated by the IRSUB caller side.

- For IRSUB (f) loading

Since the stack size used by the IRSUB (f) is 32 bytes, specify 32 bytes as the stack size. If the value m is not specified, the IRSUB (f) stack size calculated by the IRSUB caller side is 56 bytes, which is determined by adding the stack size of the IRSUB (g) that is called by the IRSUB (f). If the value m is specified, it is used as the IRSUB (f) stack size that is to be calculated by the IRSUB caller side. It is recommended that the value m not be specified.

If the specified m value is smaller than 56 bytes, the loader outputs the following warning message:

```
Warning: Stack size (f) = 56 (m) byte [Max refered (g) size = 24 byte] Err
```

- For program loading

Since the stack size used by the program is 24 bytes, specify 24 as the value n. For m, specify a value that is not smaller than 80 bytes, which is obtained by adding the value 24 to the stack size used by the called IRSUB (f) having the maximum stack size, which is 56. As is the case where no IRSUB is used, it is recommended that the specified m value be greater than adequate to provide for program modification.

If the specified m value is smaller than 80 bytes, the loader outputs the following warning message:

```
Warning: Stack size (main) = 80 (m) byte [Max refered (f) size = 56 byte] Err
```

If the value m is not specified, the loader allocates 80 bytes of stack by adding the stack size used by the IRSUB (f), which is 56.

3. LOADER

LIBRARY INTEGRITY CHECK

When compilation is performed with the `-denormalization=off` and `-round=zero` options specified, you must specify the library `libsh4nbmzz.lib` (`-lsh4nbmzz`) at the time of loading. If `libsh4nbmzz.lib` is not specified, the loader outputs the following error message:

```
svload : Error   : Undefined symbols
svload :          _use_libsh4nbmzz
```

Similarly, if `libsh4nbmzz.lib` is specified in situations where the `-denormalization=off` and `-round=zero` options are not specified for compilation purposes, the loader outputs the following error message:

```
svload : Error   : Undefined symbols
svload :          _use_libsh4nbmdn
```

If both objects are mixed to specify `-lsh4nbmzz` as well as `-lsh4nbmdn`, the loader outputs the following error message:

```
svload : rpdpload: Inconsistent object was mixed (NO:2004-25)
```

NAME

svdload

SYNOPSIS

svdload pname [option]

DESCRIPTION

The svdload command deletes the program or subprogram registered by the svload command from the development environment. However, the backup file is not zero-cleared.

ARGUMENT

pname Name of the program or subprogram to be deleted.

OPTIONS

- S This option specifies that processing mode is “system.” If this option is not specified, the predefined default access rights (environment variable RSUTYP) are selected.
- u site In site, specify the name of the site to be handled by the loader. If this option is not specified, the predefined default (RSSITE) is selected.
- +P The program (task) is deleted.
- +I The indirect link subroutine (IRSUB) is deleted.
- +U The built-in subroutine (ULSUB) is deleted.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- Other than 0: Abnormal termination.

3. LOADER

NAME

svcomp

SYNOPSIS

svcomp [option] file

DESCRIPTION

The svcomp command compares the contents of the backup file for the program and subprogram registered by the svload with the load module. Then, the command edits and displays the result.

ARGUMENT

file Specify one or more object files or libraries to be combined together compared.

OPTIONS

- S This option specifies that processing mode is “system.” If this option is not specified, the predefined default access rights (environment variable RSUTYP) are selected.
- u site In site, specify the name of the site to be handled by the loader. If this option is not specified, the predefined default (RSSITE) is selected.
- C n Meaningless for svcomp.
- p n Meaningless for svcomp.
- a area Meaningless for svcomp.
- +P Specifies that the item loaded as a program (task) is to be targeted for comparison.
- +I Specifies that the item loaded as an indirect link subroutine (IRSUB) is to be targeted for comparison.
- +U Specifies that the item loaded as a built-in subroutine (ULSUB) is to be targeted for comparison.
- +D Specifies that the item loaded as global data is to be targeted for comparison.
The type of the data conforms to the attribute of a split area to which the data belongs.
- M n Meaningless for svcomp.
- m n [n...] Meaningless for svcomp.
- Z Meaningless for svcomp.
- d Meaningless for svcomp.
- llib Specifies the library (lib) to be linked. Note that libcpms.lib and libsh4nbmdn.lib are automatically linked.
- P [file] Meaningless for svcomp.
- o obj In obj, specify the name of the load module to be created. When the specified load module name represents a subprogram, the specified name is used as the subprogram name. Ensure that the subprogram name agrees with a function in the program.
- E n Meaningless for svcomp.
- r Meaningless for svcomp.
- w n [m] Meaningless for svcomp.

LIBRARY SEARCH PATH

The loader's library search path (library search sequence specified by the -l option) agrees with the input file search sequence for the optimization linkage editor in the shc compiler package.

The input file search sequence for the optimization linkage editor is as indicated below:

- (1) Current directory
- (2) Directory specified by the environment variable `HLNK_DIR`

For the environment variable `HLNK_DIR`, a plurality of paths can be set. To specify two or more paths, separate them with semicolons.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Not different.
- 1: Different.
- 101: A command option was specified.
- Other than above: The compare failed.

svcomp RESULT DISPLAY

The `svcomp` command outputs the following messages depending on whether a difference is found in the compare.

- When no difference is found in the compare
When no difference is found in the compare, the `svcomp` command outputs the following message:

compare OK (type = X name = xxx)

X: Indicates the type of the resource to be compared.
pgm: Program.
irsub: IRSUB
ulsub: Built-in program.
data: Data (GLB).
xxx: Indicates the name of the resource to be compared.
Reads "data" when data is targeted for comparison.
- When any difference is found in the compare
When any difference is found in the compare, the `svcomp` command outputs the following message:

3. LOADER

DISPLAY FORMAT

The display given by the `compnr` command consists of the following fields:

- (1) Header
- (2) Detailed information
- (3) Footnote

Figure 2-2 shows the format of edited data for a program or subprogram, while Figure 2-3 shows that for data. The character strings in italics vary with the execution environment.

```

** compare list **                               date
site name=site-name           (1)
type=type name=name

<text> (2)
text size          new=xxxxxxx    old=xxxxxxx    (a)
loc=xxxxxxx       new=xxxxxxx    old=xxxxxxx    (b)
                  :              :
<data>
text size          new=xxxxxxx    old=xxxxxxx    (c)
loc=xxxxxxx       new=xxxxxxx    old=xxxxxxx    (d)
                  :              :
<bss>
bss size          new=xxxxxxx    old=xxxxxxx

** compare list output end ** (3)

```

Figure 2-2 Format of the Display Shown by `svcomp` for a Program or Subprogram

Each field is explained below.

(1) Header field

<i>date</i>	Time when the <code>svcomp</code> command was started
<i>site-name</i>	Name of the site processed
<i>type</i>	Type of program compared
	<code>pgm</code> : Program used as a task
	<code>irsub</code> : Indirect link subroutine
	<code>ulsub</code> : Built-in subprogram
<i>name</i>	Name of the program compared

(2) Detailed information field

This field displays the results of comparison between the old sizes of the text, data, and bss sections and their new sizes and between the old contents of the text and data sections and their new contents. When the new size is different from the old size, the comparison result is displayed relative to the smaller size.

- (a), (c): The results of comparison between the old sizes of the text, data, and bss sections and their new sizes are displayed in hexadecimal. When there is no difference between them, nothing is displayed.
- (b), (d): The results of comparison between the old contents of the text and data sections and their new contents are displayed in hexadecimal. When there is no difference between them, nothing is displayed. The hexadecimal data displayed immediately after `loc=` is the address where a difference was found. The address is relative to the beginning of the text or data.

(3) Footer field

Indicates that the compare is completed.

```

** compare list **                               date
site name=site-name           (1)
type=data name=data

<data-name>   (2)
data size     new=xxxxxxxx old=xxxxxxxx (a)
loc=xxxxxxxx  new=xxxxxxxx old=xxxxxxxx (b)
              :
              :

** compare list output end ** (3)

```

Figure 2-3 Format of the Display Shown by `svcomp` for GLB

Each field is explained below.

(1) Header field

date Time when the `svcomp` command was started
site-name Name of the site processed

(2) Detailed information field

This field displays the results of comparison between the old size of GLB, and its new size and between the old contents of GLB, and its new contents. When the new size is different from the old size, the comparison result is displayed relative to the smaller size.

- (a): The result of comparison between the old size of GLB, and its new size is displayed in hexadecimal. When there is no difference between them, nothing is displayed.
- (b): The result of comparison between the old contents of GLB, and its new contents is displayed in hexadecimal. When there is no difference between them, nothing is displayed. The hexadecimal data displayed immediately after `loc=` is the address where a difference was found. The address is relative to the beginning of the text or data.

(3) Footer field

Indicates that the compare is completed.

CHAPTER 4 BUILDER

NAME

svctask

SYNOPSIS

```
svctask pname tname tn -s n [option]
```

DESCRIPTION

The svctask creates a task using the load module loaded by the loader as a resource.

ARGUMENTS

pname	Program name of the load module to be used as a resource for task creation.
tname	Name of the task to be created.
tn	Task number from 1 to 224 for a user task or 225 to 255 for a system task. When the specified task number is already in use, an error occurs. The numbers 230 to 255 cannot be registered because they are reserved for system-supplied tasks (stored in the ROM).

OPTIONS

-S	Specifies that the processing mode is to be “system.” If this option is not specified, the predefined default access rights (environment variable RSUTYP) are selected.
-u site	In site, specify the name of the site to be handled by the builder. When this option is omitted, the site set in the RSSITE environment variable is used.
-l lvl	In lvl, specify the execution level at the initial start of the task. The level must be from 4 to 27 for a user task or from 0 to 31 for a system task. When this option is omitted, lvl is assumed to be 27 for a user task or 0 for a system task.
-r n	Specifies the stack area number for use in generating a plurality of tasks from a program that serves as a resource. This value cannot exceed the number of multitasks (-M n), which is specified by a load command. If this option is not specified, the system concludes that the minimum unused stack area number is selected.

NOTES

- User tasks are tasks for which a TN value between 1 and 224 is registered. System tasks are tasks for which a TN value between 225 and 229 is registered.
In registration in which RSUTYP=s, a system task can be generated.
- With RSUTYP=u, specification of -S option allows creation of a system task.
- No system task can be generated if the -S option is not specified when RSUTYP=u.
Combinations of task types and options are shown on the next page.

TERMINATION CODE

Returns one of the following termination codes:

0: Normal termination.

Other than 0: Abnormal termination.

Parameter		Task type	
		Single task	Multi-task
pname		√	√
tname		√	√
tn		√	√
Options	-u site	sp	sp
	-l lvl	sp	sp
	-S	sp	sp
	-r n	ns	√

√: Required sp: Can be specified ns: Cannot be specified

The table below shows the relationships between the user types and the program owner types:

User type	Program owner type	
	System	User
System	√	√
User	nc	√

√: Task can be created nc: Task cannot be created

4. BUILDER

NAME

svdtask

SYNOPSIS

svdtask tname [option]

DESCRIPTION

The svdtask command deletes the task created with svctask.

ARGUMENT

tname Name of the task to be deleted

OPTIONS

- S This option specifies that the task to be deleted is a system task. When this option is omitted, the default (set in the RSUTYP environment variable in advance) is used.
- u site Specifies the name of the site to be processed by the builder. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

NOTE

- The table below shows the relationships between the user types and the owner types for the task to be deleted.

User type	Deletion task owner type	
	System	User
System	√	√
User	nd	√

√: Task can be deleted nd: Task cannot be deleted

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- Other than 0: Abnormal termination.

NAME

svbuild (Registration of an indirect link resident subprogram)

SYNOPSIS

```
svbuild name -ir -e irno [option]
```

DESCRIPTION

The svbuild command registers an indirect link resident subprogram.

ARGUMENTS

name Indirect link resident subprogram name (entry name when multiple entries are loaded)

-ir -e irno Specify the registration number of the indirect link resident subprogram in irno.

OPTION

-u site In site, specify the name of the site to be handled by the builder. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

NOTE

The table below shows the usage of options.

Parameter \ Type		Indirect link resident subprogram
name - ir -e irno		√
Option	-u site	sp

√: Required sp: Can be specified

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- Other than 0: Abnormal termination.

4. BUILDER

NAME

svbuild (Registration of a built-in subroutine)

SYNOPSIS

```
svbuild name point en -ul [option]
```

DESCRIPTION

The svbuild command registers a built-in subroutine.

ARGUMENTS

name Built-in subroutine name
point en -ul Specify the entry point of the built-in subroutine using one of the following character strings. Also specify the entry number in en within the range of 1 to 4.

CPES: CPES built-in subroutine

IES: IES built-in subroutine

EAS: EAS built-in subroutine

INS: INS built-in subroutine

EXS: EXS built-in subroutine

ABS: ABS built-in subroutine

PCKS: PCKS built-in subroutine

MODES: MODES built-in subroutine

WDTES: WDTES built-in subroutine

ADTS: Indicates the subroutine to be linked in the event of an ADT exception.

OPTION

-u site In site, specify the name of the site to be handled by the builder. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

NOTE

Although the same program can be incorporated at a plurality of places (point), it cannot be incorporated for a plurality of entry numbers (en).

The table below shows the usage of options.

Parameter		Type	Built-in subroutine
name point en -ul			√
Option	-u site		sp

√: Required sp: Can be specified

TERMINATION CODE

Returns one of the following termination codes:

0: Normal termination.

Other than 0: Abnormal termination.

4. BUILDER

NAME

svdbuild (Deletion of an indirect link resident subprogram)

SYNOPSIS

svdbuild name -ir [option]

DESCRIPTION

The svdbuild command deletes an indirect link resident subprogram.

ARGUMENTS

name Indirect link resident subprogram name (entry name when multiple entries are loaded)
-ir The specified indirect link resident subprogram is deleted.

OPTION

-u site In site, specify the name of the site to be handled by the builder. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

NOTE

The table below shows a combination of the option.

Parameter		Type	Indirect link resident subprogram
		name - ir	
Option	-u site		sp

√: Required sp: Can be specified

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- Other than 0: Abnormal termination.

NAME

svdbuild (Deletion of a built-in subroutine)

SYNOPSIS

svdbuild name point -ul [option]

DESCRIPTION

The svdbuild command deletes a built-in subroutine.

ARGUMENTS

name Built-in subroutine name

point -ul The specified built-in subroutine is deleted. In this argument, specify one of the following character strings as the entry point of the built-in subroutine:

CPES: CPES built-in subroutine

IES: IES built-in subroutine

EAS: EAS built-in subroutine

INS: INS built-in subroutine

EXS: EXS built-in subroutine

ABS: ABS built-in subroutine

PCKS: PCKS built-in subroutine

MODES: MODES built-in subroutine

WDTES: WDTES built-in subroutine

ADTS: Indicates the subroutine to be linked in the event of an ADT exception.

OPTION

-u site In site, specify the name of the site to be handled by the builder. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

NOTE

The table below shows the usage of options.

Parameter		Type	Built-in subroutine
name point -ul			√
Option	-u site		sp

√: Required sp: Can be specified

TERMINATION CODE

Returns one of the following termination codes:

0: Normal termination.

Other than 0: Abnormal termination.

4. BUILDER

NAME

svirglb

SYNOPSIS

svirglb idxnum name [option]

DESCRIPTION

The svirglb command registers or deletes the secondary partition area allocated with svdfs as the indirect link global area.

ARGUMENTS

idxnum Specify the registration number of the indirect link global table within the range of 1 to 4096.

name Specify the indirect link global area name. (When the -s or -a option is omitted, specify the name of the secondary partition area already allocated by svdfs as the indirect link global area name.)

OPTIONS

-u site In site, specify the name of the site to be handled by the builder. When this option is omitted, the default (set in the RSSITE environment variable in advance) is used.

-s name When using the address of the indirect link global area plus an offset as the address to be stored in the indirect link global table, specify the secondary partition area name in the name argument.

-o offset When using the address of the indirect link global area plus an offset as the address to be stored in the indirect link global table, specify the offset in offset with a decimal or hexadecimal number. A number starting with the "0x" is handled as a hexadecimal number.

-a Specify the address to be stored in the indirect global table with the absolute address in decimal or hexadecimal. A number starting with the "0x" is handled as a hexadecimal number.

-d The specified registration number is deleted from the indirect link global table.

NOTES

- The -s and -o options must be specified together if either option is specified.
- When the -s, -o, and -a options are specified together, an error occurs.
- The table below shows combinations of the options.

Parameter \ Type		Indirect link global	
		Registration	Deletion
idxnum		√	√
name		√	√
Option	-u site	sp	sp
	-s name	sp	sp
	-o offset	sp	sp
	-a	sp	sp
	-d	ns	√

√: Required sp: Can be specified ns: Cannot be specified

- Only the addresses in the GLB, spaces can be specified in the -a option.
- When the specified offset is outside the split area including the secondary partition area specified by the -s option, an error occurs.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- Other than 0: Abnormal termination.

CHAPTER 5 MAP

NAME

svmap

SYNOPSIS

svmap [option]

DESCRIPTION

svmap outputs the information about resources managed by RPDP.

OPTIONS

- u sitename Specifies the name of the site to which the map information is to be output.
- G Outputs the information about a global area.
- a Outputs the information about a split area.
- e Outputs the information about a secondary partition area.
- p Outputs the information about programs. Outputs the information about programs/tasks after sorting them by name.
- s Outputs the information about subprograms.
- t Outputs the information about tasks. Outputs the information about programs/tasks after sorting them by task number.
- g Outputs global (GLB) information.
- v Outputs VAL information.
- irs Outputs IRSUB entry information.
- irg Outputs IRGLB entry information.
- uls Outputs ULSUB entry information.
- fm Outputs the information about a free space in the physical memory.
- CON Outputs the PCs memory map. If this option is not specified, the map information about resources managed by the development machine will be output.
- help Displays a list that explains about a command startup form.
- +a Outputs the results after sorting them by address.
- +n Outputs the results after sorting them by name.
- +e Outputs the results after sorting them by entry number.

+gn { name | num | pnt , typ , ent }

Outputs the information about a specified resource.

If you specify this option in conjunction with the -irs or -irg option, specify the entry number.

If you specify this option in conjunction with the -uls option, specify pnt, typ, and ent (incorporation point and entry number).

If you specify this option in conjunction with an option other than -irs, -irg, or -uls, specify the name.

-f Displays detailed information.

-en Outputs the IRSUB/GLB/VAL entry information only.

When you specify the +gn name option in conjunction with the -G, -a, or -e option, you can output a garea/area hierarchical map. For the relationship between option combinations and output results, see Table 2-5.

If you specify the -u option and leave all the other options unspecified, the system outputs split areas and secondary partition area listings sorted by address, program, subprogram, task, and global data listings sorted by name, IRSUB and IRGLB listings sorted by number, and VAL listings sorted by name.

TERMINATION CODE

Returns one of the following termination codes:

0: Normal termination.

Other than 0: Abnormal termination.

5. MAP

Table 2-5 Relationship between Option Combinations and Outputs

Output description	Option														
	+gn	+a	+n	+e	-G	-a	-e	-p	-s	-t	-g	-v	-irs	-irg	-uls
Hierarchical map of specified gareas/areas (sorted by address)	√	(√)			√	√									
	√	(√)			√	√	√								
	√	(√)				√	√								
Hierarchical map of specified gareas/areas (sorted by name)	√		√		√	√									
	√		√		√	√	√								
	√		√			√	√								
Information about a specified name	√				●	●	●	●	●	●	●	●	●	●	●
Specified information listings sorted by address		√			⊙	⊙	⊙	ns	ns	ns	ns	ns	ns	ns	ns
Specified information listings sorted by name			√		⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙
Specified information listings sorted by number				√	ns	ns	ns	ns	⊙	⊙	⊙	ns	⊙	⊙	⊙
Specified information listings (in default order)(*)					+a	+a	+a	+n	+n	+e	+n	+n	+e	+e	+e

√: Mandatory (√): Mandatory (Can be omitted)

●: Can be specified (Cannot be duplicated)

⊙: Can be specified (Can be duplicated) ns: Cannot be specified

(*) The +a, +n, and +e entries in the above table represent a default in situations where the sorting order option is not specified.

NAME

svadm

SYNOPSIS

svadm [addr] [option]

DESCRIPTION

svadm outputs the name and other information about the resource registered at a specified logical address (see under DISPLAY FORMATS for details). When a logical address is not specified, the command prompts the user to enter it interactively and displays names and other information.

ARGUMENT

addr Specify a logical address in addr within the range of 0x3000000 to 0x7bffffff (for the task space, GLB space or subprogram space). When this argument is omitted, the user is prompted to enter a logical address interactively.

OPTIONS

-u site In site, specify the name of the site for which information on the area indicated by the specified logical address is to be displayed. When this option is omitted, the default site name set in advance is used.

-o file In file, specify the name of the file to which the operation result is to be output.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- Other than 0: Abnormal termination.

5. MAP

OPERATION

- When an address is specified in the argument

```
#svadm addr [Enter]
```

```
Information is displayed.
```

```
#
```

- When an address is not specified in the argument

```
#svadm [Enter]
```

```
++ address information display start --> site (XXXXX) ++
```

```
*addr : addr [Enter]
```

```
Information is displayed.
```

```
*addr : q [Enter]
```

```
++ address information display end ++
```

```
#
```

Explanation

Enter data at the underlined portions.

XXXXX: Site name

addr: Specify the address that indicates the area from which the user wants to get information.

q: The command is terminated.

[Enter]: Press the [Enter] key.

DISPLAY FORMATS

The following four display formats are supported:

- [1] name = XXXXXXXX type = XXXXXXXXXXXX raddr = XXXXXXXX
- [2] lspace = XXXXXXXX external name is not defined
- [3] address error

Explanation

XXX is data.

Display format [1] is used when a resource is registered in a logical space at a specified address.

Display format [2] is used when no resource is registered at a specified address.

Display format [3] is used in the following situations:

- When the specified address is outside the range stated under ARGUMENT.
- When the specified address is within the range stated under ARGUMENT but outside the GAREA limit.

name	External name (program name, subprogram name, or secondary partition area name)
type	Attribute of the external name
task	Program name
sub	Subprogram name
data	Global
raddr	Address relative to the beginning of the area indicated by the external name. One of the following relative addresses is displayed according to the external name.
task	Address relative to the start logical address of the text section of the program
sub	Address relative to the start logical address of the text section of the subprogram
data	Address relative to the start logical address of the secondary partition area in the global space
lspace	Logical space name of the area indicated by the specified address. This name is displayed only when the external name is not defined.

5. MAP

NAME

svsitectl

SYNOPSIS

svsitectl [option]

DESCRIPTION

svsitectl controls and displays the status of a specified site.

OPTIONS

- query Lists the sites registered with the development machine.
- loadall sitename
Ensures that all the resources registered for a specified site are downloaded.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- Other than 0: Abnormal termination.

CHAPTER 6 STARTUP

NAME

svrpl - Perform remote loading.

SYNOPSIS

```
svrpl [{-u site} {-s}] [-r] [{-time|-notime}]
```

DESCRIPTION

svrpl stops a specified site, transfers the contents of a backup file to the PC's main memory, and starts the specified site. The following options can be specified:

OPTIONS

- u site Specifies the site that is targeted for loading. If this option is not specified, svrpl processes the site that is set for the environment variable RSSITE.
- r Does not start a specified site upon completion of loading.
- s Stops a specified site without acknowledging whether or not to stop the specified site.
Specify this option in conjunction with the -u option.
If the -s option is not specified, svrpl prompts for operator intervention in an interactive mode and asks whether or not to stop the site at the time of a download.
- time Sets the time for a specified site.
- notime Does not set the time for a specified site.
Specify this option if you do not want to set the time.

NOTES

- If an error occurs during loading, svrpl terminates while leaving the specified site stopped.
- If the specified site is not to be started after a download (the -r option is specified), svrpl does not set the time because it ignores the -time option even when it is specified.
- If the -time option is not specified, svrpl does not set the time at the time of a download. The time can be set by specifying the -time option when the status control (svcpuctl) command is used to issue a RUN request.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination.
- 1: Abnormal termination.
- 2: A communication error occurred.
- 3: A signal was received.

6. STARTUP

NAME

`svcpuctl` - Control the remote status.

SYNOPSIS

```
svcpuctl [{-u site} {-s {-stop|-run}}] [-time] (status control)
svcpuctl [-u site] -ss (status display)
```

DESCRIPTION

The `svcpuctl` command controls the specified site (PU) or displays the status of the specified site (PU). The following options can be specified:

OPTIONS FOR STATUS CONTROL

- `-u site` Specify the name of the site to be handled. When this option is omitted, the site set in the `RSSITE` environment variable is used.
- `-s` An acknowledgment (response that asks whether the user actually wants to execute this command) is not received. Specify the `-s` option together with the `-u` option, if necessary. When the `-s` option is not specified, operation by the operator is assumed — the user is prompted to specify the status (STOP or RUN) interactively.
- `-stop` The specified site is placed in the STOP status. Specify the `-stop` option together with the `-s` option, if necessary. The `-stop` and `-run` options are mutually exclusive. Do not specify them together.
- `-run` The specified site is placed in the RUN status. Specify the `-run` option together with the `-s` option, if necessary. The `-run` and `-stop` options are mutually exclusive. Do not specify them together.
- `-time` When a run request is issued with the CPU being specified as the site, the time is set.

`svcpuctl` first displays the status of a specified site. When the user instructs the RUN or STOP status, `svcpuctl` asks whether the user actually wants to change the specified site to that status. To change the status of the specified site, respond with `yes`. When `no` is the response, `svcpuctl` does nothing and terminates.

OPTION FOR STATUS DISPLAY

- `-ss` The PU status (RUN or STOP) of the specified site is displayed.

NOTES ON USE

The options for status control and those for status display are mutually exclusive. Do not specify these two types of options together.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination
- 1: Abnormal termination
- 2: Communication error
- 3: Signal received

CHAPTER 7 svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

svdebug - Online debugger

SYNOPSIS

svdebug [option]

DESCRIPTION

The svdebug command offers an on-line debugging function.

This command outputs the following message when it is started and terminated.

When the “site name” prompt appears upon debugger startup, various subcommands are acceptable.

If breakpoint setup and ADT setup are completed prior to svdebug startup, svdebug displays their settings.

<Message at the start of the debugger>

```
++ debugger start ++
address detect trap
addr = 0x*****-0x*****   mode = *****
break point
name = program name raddr = program internal relative address object =
machine language instruction pattern
      .
      .
name = program name raddr = program internal relative address object =
machine language instruction pattern
site name>
```

<Message on the termination of the debugger>

```
++ debugger end ++
```

NOTE: When the debugger is started with the -s option specified, the above messages are not displayed.

OPTIONS

- i fname The entered keystrokes are output to the file specified in fname.
- o fname The date and result of operation are output to the file specified in fname.
- r fname Subcommands in the command file specified in fname are executed. After all subcommands are executed, the svdebug command automatically terminates. The file created with the -i option is used as a command file.
- s subcommand The subcommands specified in this option are directly executed. After execution of the subcommands, the svdebug command automatically terminates.
- u site Specify the name of the site to be handled by the debugger.
- debug Specifies a debug mode. Extended subcommands are usable.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

TERMINATION CODE

When terminated normally, the svdebug command returns the value 0. When terminated abnormally, the command returns the value 1. However, if a subcommand specified by the -s option terminates abnormally, the command returns the value 255.

NOTES

- When multiple options are specified, the options specified after the -s option are regarded as subcommands and ignored.

Example

```
svdebug -i fname -s subcommand:  -i is regarded as an option.
svdebug -s subcommand -i fname:  -i is regarded as part of the subcommands
                                specified in -s.
```

- When the option specified in the svdebug command is not any of the options described in DESCRIPTION, the following usage information is displayed:

```
Usage: svdebug [options]
Options:
  -i fname      specify a "key-input result file"
  -o fname      specify a "operation result file"
  -r fname      specify a "command file"
  -s sub command "sub command" direct run
  -u site       specify a "site name"
```

- When creating a key-input result file as the command file used by the -r option rather than using the file specified by the -i option, note the following points:
 - When data not conforming to the subcommand specifications is set, the line including the data is executed without being checked.
 - For a subcommand having an interactive interface, enter data corresponding to the prompt number for each operation procedure on a per-line basis.
 - A null line in the command file is ignored, assuming that a subcommand line is not encountered.
- When the -r option or -s option is used, breakpoint subcommands (br, rb, rr, rd, and go) cannot be used.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

- List of subcommands

Table 2-6 summarizes the functions of svdebug.

Table 2-6 Functions of svdebug

Classification	Subcommand	Function
Task start or stop	qu	Requests that a task be started.
	ab	Inhibits tasks from being started.
	re	Cancels inhibition of task start.
	ta	Displays task status information.
	su	Suspends or suppresses task execution.
	rs	Cancels suppression of task execution.
	tm	Starts a task cyclically.
	ct	Cancels cyclic task start.
	sht	Displays task cyclic starts.
	si	Effects task initialization.
Memory print or patch	sp	Displays the amount of stack use.
	md	Displays/changes the memory contents in accordance with a specified address.
	sd	Displays/changes the memory contents in accordance with a specified name.
	bs	Sets data in specified bit positions.
	bg	Displays data existing in specified bit positions.
	mcp	Copies the memory contents.
	mmv	Moves the memory contents.
Breakpoint	mF	Sets a pattern value in the memory.
	br	Sets/displays breakpoints.
	rb	Resets breakpoints.
	rd	Displays registers.
	rr	Rewrites the contents of registers.
System error display	g	Resumes execution from a breakpoint.
	el	Displays error log.
Current time setting or display	ss	Displays system status information.
	st	Sets the current time.
Uploading, downloading, or comparison	gt	Displays the current time.
	ld	Downloads resources individually.
	sv	Backs up resources individually.
Enabling or displaying DHP logging	cm	Compares the contents of a backup file and PC's memory.
	dr	Enables DHP logging.
ADT	ds	Disables DHP logging.
	as	Sets/displays the ADT.
Others	ac	Resets the ADT.
	svdhp	Displays the DHP.
	svadm	Displays a resource name for an address.
	ps	Starts displaying a debug statement.
	pe	Finishes displaying a debug statement.
	ver	Displays the version of the CPMS.
	help	Displays a list of subcommands.
	q	Exits the debugger.
!	Executes a command on the development machine at the time of svdebug execution.	

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

- Operation performed upon receipt of a signal
Table 2-7 shows the operations that are performed upon receipt of a signal.

Table 2-7 Operation on Reception of a Signal

Signal	Operation
SIGQUIT SIGINT	The debugger terminates the subcommand being executed and waits for another subcommand to be entered.
SIGHUP SIGTERM	The svdebug command terminates its processing.
Others	SIG_DFL

NAME

qu - Request that a task be started.

SYNOPSIS

```
qu tn[,fact]
qu tname[,fact]
```

DESCRIPTION

The qu subcommand starts the specified task. Specify the following parameters:

tn	Task number (1 to maximum task number)
fact	Start factor (1 to 32)
tname	Task name

RESULT

OK (0)	Normal termination
NG (≠0)	Macro error

The code returned by the macro is displayed.

NOTES

- When the fact parameter is omitted, fact=0 is assumed.
- When this subcommand is started with no parameters specified or with invalid arguments specified, the message shown below is displayed to prompt for parameters. Enter correct parameters after the colon (":"). If you press the [e] or [Enter] key in this state, the subcommand process terminates.

```
input tn[,fact] or tname[,fact]
:
```


7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

ab - Inhibit tasks from being started.

SYNOPSIS

```
ab tn1 [-tn2]
```

```
ab tname
```

DESCRIPTION

The ab subcommand inhibits the specified tasks from being started. Specify the following parameters:

tn1	First task number (1 to maximum task number)
tn2	Last task number (1 to maximum task number)
tname	Task name

RESULT

OK (0) Normal termination

NG (≠0) Macro error

When tn1-tn2 is specified, however, the ab subcommand always terminates normally. When a macro error occurs, the code returned by the macro is displayed.

NOTE

When this subcommand is started with no parameters specified or with invalid arguments specified, the message shown below is displayed to prompt for parameters. Enter correct parameters after the colon (":"). If you press the [e] or [Enter] key in this state, the subcommand process terminates.

```
input tn1[-tn2] or tname
```

```
:
```

NAME

re - Cancel inhibition of task start.

SYNOPSIS

```
re tn1[-tn2]
```

```
re tname
```

DESCRIPTION

The re subcommand releases the specified tasks from the status in which task start is inhibited. Specify the following parameters:

tn1	First task number (1 to maximum task number)
tn2	Last task number (1 to maximum task number)
tname	Task name

RESULT

OK (0) Normal termination

NG (≠0) Macro error

When tn1-tn2 is specified, however, the re subcommand always terminates normally. When a macro error occurs, the code returned by the macro is displayed.

NOTE

When this subcommand is started with no parameters specified or with invalid arguments specified, the message shown below is displayed to prompt for parameters. Enter correct parameters after the colon (“:”). If you press the [e] or [Enter] key in this state, the subcommand process terminates.

```
input tn1[-tn2] or tname
```

```
:
```

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

ta - Display task status information.

SYNOPSIS

```
ta tn1 [-tn2 [-s|-r]]
ta tname
```

DESCRIPTION

The ta subcommand displays status information on the specified tasks. When a specified task is in the SUSPENDED state in which task execution is suppressed, the contents of registers involved are also displayed. Specify the following parameters:

tn1	First task number (1 to maximum task number)
tn2	Last task number (1 to maximum task number)
tname	Task name
-s	Only task numbers, task names, and task status information are displayed.
-r	The status information for tasks that are not in the NON-EXISTENT, DORMANT, or IDLE state is displayed.

The ta subcommand displays status information in the following format:

```
tn=*** (0x**) tname=***** task state=***** (0x*****)
tcb top=0x***** fact=0x***** level=** (**)
task top=0x***** stack=0x*****-0x*****
(Contents of registers)
```

● Explanation of displayed information

tn	Task number (decimal or hexadecimal)
tname	Task name
task state	State of the task (The settings of status bits are displayed in hexadecimal.)
fact	Start factor of the task
level	Current task level (The value in parentheses is the initial task level.)
tcb top	First address of the TCB
task top	First address of the task
stack	Task stack range
Contents of registers	When the task is in the SUSPENDED state, the contents of registers involved are displayed.

The contents of registers are displayed in the following format:

```
IAR =0x***** MSR =0x***** CR =0x***** LR =0x*****
CTR =0x***** XER =0x*****
GR0 =0x***** GR1 =0x***** GR2 =0x***** GR3 =0x*****
GR4 =0x***** GR5 =0x***** GR6 =0x***** GR7 =0x*****
GR8 =0x***** GR9 =0x***** GR10=0x***** GR11=0x*****
GR12=0x***** GR13=0x***** GR14=0x***** GR15=0x*****
GR16=0x***** GR17=0x***** GR18=0x***** GR19=0x*****
GR20=0x***** GR21=0x***** GR22=0x***** GR23=0x*****
GR24=0x***** GR25=0x***** CR26=0x***** GR27=0x*****
GR28=0x***** GR29=0x***** CR30=0x***** GR31=0x*****
```

Table 2-8 lists the task states, one of which is displayed for a task at a time.

Table 2-8 Task States

State	Meaning
NON EXISTENT	The task is not registered.
DORMANT	Start of the task is inhibited.
IDLE	The task is waiting to be started.
READY	The task is being executed or is ready for execution.
WAIT	The task is waiting for an event.
SUSPENDED	Execution of the task is suspended or suppressed.

Table 2-9 shows the status bit strings and their meanings.

Table 2-9 Status Bit Stings

Status bit string	Meaning
0x1	Multiple starts were requested.
0x10	Execution is being suppressed by DELAY.
0x20	Execution is being suppressed by SUSP.
0x40	Unlocking of Resources locked by RSERV or PRSRV is being awaited.
0x1000	Processing by EXIT is in progress.
0x2000	Processing by RLEAS is pending.
0x4000	Processing by ABORT is in progress.
0x8000	Processing by QUEUE is pending.

NOTE: Multiple bits in each status bit string may be turned on at the same time.

NOTES

- When this subcommand is started with no parameters specified or with invalid arguments specified, the message shown below is displayed to prompt for parameters. Enter correct parameters after the colon (":"). If you press the [e] or [Enter] key in this state, the subcommand process terminates.

```
input tn1[-tn2 [-s|-r]] or tname
```

```
:
```

- When the -susp option is specified, the task range cannot be specified.
- If the task is DORMANT or NON-EXISTENT when the -susp option is specified, the system generates an error message and terminates the subcommand.
- When the -susp option is specified, the task is temporarily SUSPENDED in order to acquire register information even if the task is not in the SUSPENDED state.
- Do not use the -susp option during the on-line state.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

su - Suppress task execution.

SYNOPSIS

```
su tn1 [-tn2]
su tname
```

DESCRIPTION

The su subcommand suppresses execution of the specified tasks. Specify the following parameters:

tn1	First task number (1 to maximum task number)
tn2	Last task number (1 to maximum task number)
tname	Task name

RESULT

OK (0) Normal termination

NG (≠0) Macro error

When tn1-tn2 is specified, however, the su subcommand always terminates normally. When a macro error occurs, the code returned by the macro is displayed.

NOTE

When this subcommand is started with no parameters specified or with invalid arguments specified, the message shown below is displayed to prompt for parameters. Enter correct parameters after the colon (":"). If you press the [e] or [Enter] key in this state, the subcommand process terminates.

```
input tn1[-tn2] or tname
:
```

Do not use the -susp option during the on-line state.

NAME

`rs` - Cancel suppression of task execution.

SYNOPSIS

```
rs tn1[-tn2]
```

```
rs tname
```

DESCRIPTION

The `rs` subcommand releases the specified tasks from the condition in which task execution is suppressed. Specify the following parameters:

<code>tn1</code>	First task number (1 to maximum task number)
<code>tn2</code>	Last task number (1 to maximum task number)
<code>tname</code>	Task name

RESULT

`OK (0)` Normal termination

`NG (≠0)` Macro error

When `tn1-tn2` is specified, however, the `rs` subcommand always terminates normally. When a macro error occurs, the code returned by the macro is displayed.

NOTE

When this subcommand is started with no parameters specified or with invalid arguments specified, the message shown below is displayed to prompt for parameters. Enter correct parameters after the colon (“:”). If you press the [e] or [Enter] key in this state, the subcommand process terminates.

```
input tn1[-tn2] or tname
```

```
:
```

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

tm - Start a task cyclically.

SYNOPSIS

```
tm
id:
tn[,fact]:  OR tname[,fact])
t, cyct:
```

or

```
tm
id:
tn[,fact]:  OR tname[,fact]
t:
```

DESCRIPTION

The tm subcommand starts the specified task cyclically. When tm is started, the subprompt is displayed. Specify the following parameters:

id	Type of the timer task to be started (1 to 4)
tn	Task number (1 to maximum task number)
tname	Task name
fact	Start factor (1 to 32)
t	Time of day or length of time relative to the current time when the timer event is started for the first time. The relative time must be specified in milliseconds within the following range: When id is 1 or 3: 1 to 86,400,000 When id is 2 or 4: 0 to 86,399,999
cyct	Interval at which events are generated cyclically. The interval must be specified in milliseconds within the range of 1 to 86,400,000.

For details of id, t, and cyct, see Table 2-10.

RESULT

OK (0)	Normal termination
NG (≠0)	Macro error

When a macro error occurs, the code returned by the macro is displayed.

NOTES

- When the fact parameter is omitted, fact=0 is assumed.
- If you press the [e] or [Enter] key while the system is waiting for parameter input, the subcommand terminates.

Table 2-10 Explanation of the id, t, and cyct Parameters

Timer event	id	t	cyct	Explanation
Length-of-time basis	1	Relative time up to the start time, measured from the current time of day		After the length of time specified by the t parameter elapses, the task specified by the tn or tname parameter is started.
Time-of-day basis	2	Time of day at which the task is started, measured from 00:00		The task specified by the tn or tname parameter is started at the time specified by the t parameter.
Length-of-time and cycle basis	3	Relative time up to the start time, measured from the current time of day (relative time up to the first start)	Interval at which the task is started cyclically after the first start	After the length of time specified by the t parameter elapses, the task specified by the tn or tname parameter is started. Then, the task is started cyclically at the interval specified by the cyct parameter.
Time-of-day and cycle basis	4	Time of day at which the task is started, measured from 00:00 (time of day at the first start)	Interval at which the task is started cyclically after the first start	The task specified by the tn or tname parameter is started at the time specified by the t parameter. Then, the task is started cyclically at the interval specified by the cyct parameter.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

ct - Cancel cyclic task start.

SYNOPSIS

```
ct tn[,fact]
ct tname[,fact]
```

DESCRIPTION

The ct subcommand cancels cyclic start of the specified task. Specify the following parameters:

tn	Task number (1 to maximum task number)
fact	Start factor (1 to 32)
tname	Task name

RESULT

OK (0)	Normal termination
NG (≠0)	Macro error

When a macro error occurs, the code returned by the macro is displayed.

NOTES

- When the fact parameter is omitted, fact=0 is assumed.
- When this subcommand is started with no parameters specified or with invalid arguments specified, the message shown below is displayed to prompt for parameters. Enter correct parameters after the colon (":"). If you press the [e] or [Enter] key in this state, the subcommand process terminates.

```
input tn[,fact] or tname[,fact]
:
```

NAME

sht - Displays task cyclic starts.

SYNOPSIS

sht

DESCRIPTION

sht displays the cyclic starts of all the currently selected tasks.

RESULT

Displays task cyclic starts in the following form:

ID	TN	FACT	TIME	CYT
*	***	**	****/**/** **:*:*:*.***	*****
			.	
			.	
			.	

ID Timer type (1: period-of-time-specific start (timer); 2: point-in-time-specific start (timer); 3: period-of-time-specific cyclic start (timer); 4: point-in-time-specific cyclic start (timer); 10: point-in-time-specific start (wake); 11: point-in-time-specific cyclic start (wake)).

TN Task number.

FACT Start factor.

TIME Start time (year/month/day hour:minute:second millisecond)

CYT Cycle (milliseconds)

NOTE

Multiple instances of the sht subcommand cannot be simultaneously started for the same site.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

md - Display or modify addressed memory.

SYNOPSIS

```
md
1 storage (s,m,*):      {s}
                       {m}
                       {*}
                       {e}
                       {return}
2 addr:  {addr1[-addr2] [-h|-d|-f] [-l|-w|-b] [-all|-omit]}
         {addr1[-addr2] [-fd] [-all|-omit]}
         {addr1[,len] [-h|-d|-f] [-l|-w|-b] [-all|-omit]}
         {addr1[,len] [-fd] [-all|-omit]}
         {*n}
         {e}
0xaaaaaaaa 0xdddddddd: {data}
                       {return}
                       {e}
```

NOTE: The underlined portions are data to be entered by the user.

DESCRIPTION

The md subcommand displays or modifies the contents of logical addressed memory. When this subcommand is started, subprompts are displayed. Respond to each subprompt as follows:

```
1 storage (s,m,*)
s      The backup file is modified or displayed.
m      Memory in the controller is modified or displayed.
return Memory in the controller is modified or displayed.
*      Both the backup file and memory in the real machine are modified.
e      The subcommand is instructed to terminate.
```

NOTE: When storage=* is specified, data in the backup file is displayed.

```
2 addr
addr1-addr2  Data from the first display address (addr1) to the last display address
              (addr2) is displayed.
addr1,len    Data is displayed for the number of bytes specified by len, starting from
              the first display address (addr1).
-h           Data is displayed in hexadecimal.
-d           Data is displayed in decimal.
-f           Data is displayed in the single-precision floating-point format.
-fd          Data is displayed in the double-precision floating-point format.
-l           The data length is set to four bytes.
-w           The data length is set to two bytes.
-b           The data length is set to one byte.
-all        Specifies that an abbreviated display mode is not to be used even when
              there are two or more consecutive lines of the same data.
-omit        Specifies that an abbreviated display mode is to be used when there are
              two or more consecutive lines of the same data.
*n           Returns to an earlier process having the prompt number n (n is
              acceptable only when it is 1).
e           The subcommand is instructed to terminate.

0xaaaaaaaa 0xdddddddd
data        Specify the new value as the data to be changed.
return      No data is changed.
e           Control is returned to "2 addr."
```

NOTES

- When the `addr2` or `len` parameter is not specified in “2 addr,” data change (patch) mode is enabled. When either parameter is specified, data display (print) mode is enabled. Upon completion of data display, the user is prompted for an input again in “2 addr.”
- When neither a data output format option nor a data length option is specified, the last options specified in this subcommand are used. By default, the `-h` and `-l` options are used, that is, four-byte data is displayed in hexadecimal.
- The values specifiable in data change (patch) mode depend on the combination of the specified data output format option and data length option. Table 2-11 shows the relationships between specifiable values and options.

Table 2-11 Relationships between Specifiable Values and Options

fmt \ size	-l	-w	-b	No specification
-h	○	○	○	—
-d	○	○	○	—
-f	⊙	○	○	—
-fd	×	×	×	⊙
No specification	—	—	—	—

○: Octal, decimal, or hexadecimal numbers can be specified.

⊙: Real numbers can be specified.

—: Specifiable values depend on the values of the previously specified `fmt` and `size`.

×: Invalid combination

`fmt`: Data output format `size`: Data size

- After data is displayed or changed by `-fd` (double-precision floating-point format) specified as the data output format, data may be displayed or changed, with either the data output format or data size option omitted. When the data output format option is omitted, `-h` (hexadecimal format) is assumed. When the data size option is omitted, `-l` (four bytes) is assumed.
- If “s” is specified for “1 storage” and the address at which no backup file exists is specified for “2 addr,” the system displays an error message and waits for “2 addr” input.
- If “s” is specified for “1 storage” and the address at which the package’s shared resource is contained is specified for “2 addr,” the system displays an error message and waits for “2 addr” input.
- In data display (print) mode, the display format depends on the combination of the specified data output format option and data length option, as shown in Table 2-12.

Table 2-12 Display Formats Depending on the Combination of Options

fmt \ size	-l	-w	-b	No specification
-h	h/4	h/2	h/1	—
-d	d/4	d/2	d/1	—
-f	f/4	h/2	h/1	—
-fd	×	×	×	f/8
No specification	—	—	—	—

`fmt/size`: `fmt`: h (hexadecimal)

d (decimal)

f (floating-point)

`size`: byte size

—: Specifiable values depend on the values of the previously specified `fmt` and `size`.

×: Invalid combination

`fmt`: Data output format `size`: Data size

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

- Figure 2-4 shows the memory access range allowed for the md subcommand that changes (reads) or modifies (writes) the contents of memory. Spaces are inaccessible if they are not mapped in the physical memory.

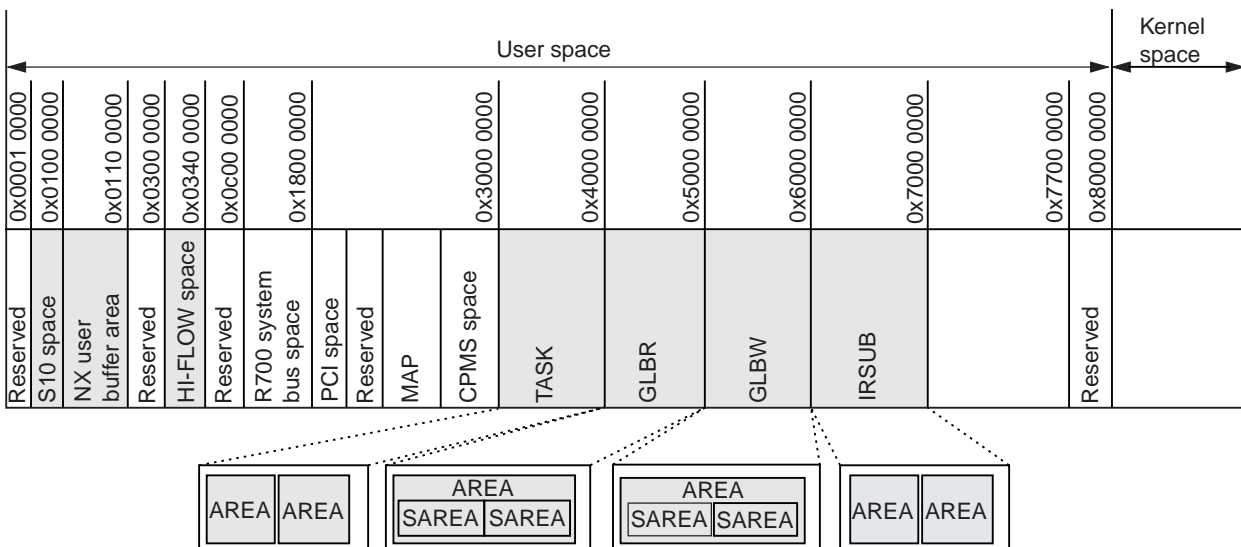


Figure 2-4 Memory Access Range

- md supports the dynamic display function in the data display (print) mode. Figure 2-5 shows the operation procedure for dynamic display.

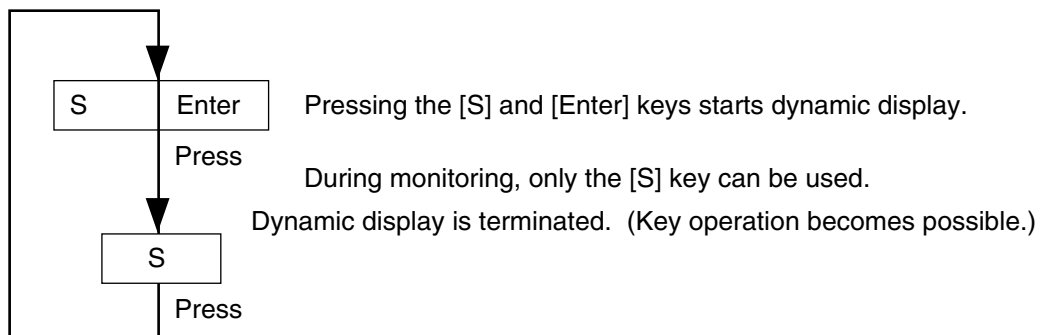


Figure 2-5 Operation Procedure for Dynamic Display

NAME

sd - Display or modify memory specified with a task name, subprogram name, program name, or global name.

SYNOPSIS

```
sd
1 name : {name [-t|-s|-g|-b|-w]}
         {e}
2 storage(s,m,*) : {s}
                   {m}
                   {*}
                   {*n}
                   {e}
                   {return}
3 baddr : {addr [-h|-d|-f] [-l|-w|-b] [-all|-omit]}
          {addr [-fd] [-allv|-omit]}
          {*n}
          {e}
4 raddr : {addr1[-addr2|-*]}
          {addr1 [, len | ,*]}
          {*}
          {*n}
          {e}
0xaaaaaaaa (0x111111) 0xdddddddd : {data}
                                   {return}
                                   {e}
```

NOTE: The underlined portions are data to be entered by the user.

DESCRIPTION

The **sd** subcommand displays or modifies the contents of memory specified by a task name, subprogram name, program name, or global name. When this subcommand is started, subprompts are displayed. Respond to each subprompt as follows:

```
1 name
  name      Name for which memory is displayed or modified
  -t        The name specified in name is handled as a program name. The relative
            address is handled as the address counted from the beginning of the text.
  -s        The name specified in name is handled as a subprogram name.
  -g        Specifies that the name specified in the "name" position is to be used as the
            global data's secondary partition area (SAREA) name.
  -b        Specifies that the BSS area of a program corresponding to the program name
            specified in the "name" position is to be handled. The relative address is
            handled as the address counted from the beginning of the BSS area for a
            specified program.
  -w        The stack area of the task identified by the name specified by name is handled.
            The relative address is handled as the address counted from the beginning of
            the stack area for the specified task.
  e         The subcommand is instructed to terminate.
```

NOTE: When none of the -t, -s, -g, -b, and -w options is specified, -g is assumed.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

- 2 storage (s, m, *)
- s The backup file is modified or displayed.
 - m Memory in the real machine is modified or displayed.
 - return Memory in the real machine is modified or displayed.
 - * Both the backup file and memory in the real machine are modified.
 - *n Control is returned to preprocessing indicated by the prompt number specified in n. (Only the number 1 can be specified in n.)
 - e The subcommand is instructed to terminate.

NOTE: When storage=* is specified, data in the backup file is displayed.

- 3 baddr
- addr Specify an address relative to the beginning of the area to be modified or displayed.
 - *n Control is returned to preprocessing indicated by the prompt number specified in n (n = 1 or 2).
 - h Data is displayed in hexadecimal.
 - d Data is displayed in decimal.
 - f Data is displayed in the floating-point format.
 - l The data length is set to four bytes.
 - w The data length is set to two bytes.
 - b The data length is set to one byte.
 - all Specifies that an abbreviated display mode is not to be used even when there are two or more consecutive lines of the same data.
 - omit Specifies that an abbreviated display mode is to be used when there are two or more consecutive lines of the same data.
 - e The subcommand is instructed to terminate.

- 4 raddr
- addr1-addr2 Data from the first display address (addr1) to the last display address (addr2) is displayed. (These addresses are counted, starting from addr of baddr.)
 - addr1, len Data is displayed for the number of bytes specified by len, starting from the first display address (addr1). (The address is counted, starting from addr of baddr.)
 - addr1, * The area indicated by the specified symbol is displayed, starting from the first display address (addr1) and continuing up to the end of the area. (The address is counted, starting from addr of baddr.)
 - addr1-* The area indicated by the specified symbol is displayed, starting from the first display address (addr1) and continuing up to the end of the area. (The address is counted, starting from addr of baddr.)
 - * The entire area indicated by the specified symbol is displayed.
 - *n Control is returned to preprocessing indicated by the prompt number specified in n (n = 1, 2, or 3).
 - e The subcommand is instructed to terminate.

- 0xaaaaaaaa (0x111111) 0xdddddddd
- data Specify the new value as the data to be changed.
 - return No data is changed.
 - e Control is returned to "4 raddr."

NOTES

- When only the `addr1` parameter is specified in “4 raddr,” data change (patch) mode is enabled. Other combinations of specified parameters enable data display (print) mode. Upon completion of data display, the user is prompted for an input again in “4 raddr.”
- When neither a data output format option nor a data length option is specified, the last options specified in this subcommand are used. By default, the `-h` and `-l` options are used, that is, four-byte data is displayed in hexadecimal.
- The areas that can be displayed or modified by `sd` are secondary partition areas (SAREAs) allocated in the global area when a global name is specified. When a program or subprogram name is specified, the displayable or modifiable area is the text and data sections, the `bss` section, or the stack section.
- The values specifiable in data change (patch) mode depend on the combination of the specified data output format option and data length option. Table 2-11 shows the relationships between specifiable values and options.
- As regards the display format for data display (print), the combination of the data output form and data length options is the same as for the `md` subcommand.
- As is the case with the `md` subcommand, the `sd` subcommand supports the dynamic display function in the data display (print) mode. For the dynamic display operation, see the section on the `md` subcommand.
- If an area containing no backup file is specified for “1 name” and “s” is specified for “2 storage,” the system displays an error message and waits for “2 storage” input.
- If the package’s shared resource area is specified for “1 name” and “s” is specified for “2 storage,” the system displays an error message and waits for “2 storage” input.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

bs - Set data in specified bit positions.

SYNOPSIS

```
bs
1 storage (s,m,*): {s}
                  {m}
                  {*}
                  {e}
                  {return}

2 addr: {addr}
        {*n}
        {e}

3 bit:  {bit1, len}
        {bit1-bit2}
        {*n}
        {e}

4 data: {data}
        {*n}
        {e}
```

NOTE: The underlined portions are data to be entered by the user.

DESCRIPTION

The bs subcommand sets data at the bit positions having the specified addresses. When this subcommand is started, subprompts are displayed. Respond to each subprompt as follows:

```
1 storage (s,m,*)
s      Data is set in bits in the backup file.
m      Data is set in bits in memory in the real machine.
return Data is set in bits in memory in the real machine.
*      Data is set in bits in both the backup file and memory in the real machine.
e      The subcommand is instructed to terminate.

2 addr
addr   Specify the address of memory in which to set data.
*n     Control is returned to preprocessing indicated by the prompt number n
       (where n = 1).
e      The subcommand is instructed to terminate.

3 bit
bit1, len Data is set in bits whose length is specified by len, starting from the first
          bit number (bit1).
bit1-bit2 Data is set in bits between the first bit number (bit1) and last bit number
          (bit2).
*n      Control is returned to preprocessing indicated by the prompt number n (a
          value between 1 and 3 can be specified as n).
e      The subcommand is instructed to terminate.
```

4	data	
	data	Specify data to be set. When the data starts with 0x or 0X, it is handled as hexadecimal data. Otherwise, the data is handled as binary data. Specify as many patterns as the number of specified bits.
	*n	Control is returned to preprocessing indicated by the prompt number n (a value between 1 and 3 can be specified as n).
	e	The subcommand is instructed to terminate.

NOTES

- The user can modify the contents of memory within the same range as for the md subcommand.
- If “s” is specified for “1 storage” and the address at which no backup file exists is specified for “2 addr,” the system displays an error message and waits for “2 addr” input.
- If “s” is specified for “1 storage” and the address at which the package’s shared resource is contained is specified for “2 addr,” the system displays an error message and waits for “2 addr” input.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

bg - Display data existing in specified bit positions.

SYNOPSIS

```
bg
1 storage (s,m,*): {s}
                  {m}
                  {*}
                  {e}
                  {return}

2 addr: {addr}
        {*n}
        {e}

3 bit:  {bit1, len}
        {bit1-bit2}
        {*n}
        {e}
```

NOTE: The underlined portions are data to be entered by the user.

DESCRIPTION

The bg subcommand displays data at the specified bit positions. When this subcommand is started, subprompts are displayed. Respond to each subprompt as follows:

```
1 storage (s,m,*)
s      Bit data is read from the backup file.
m      Bit data is read from memory in the real machine.
return Bit data is read from memory in the real machine.
*      Bit data is read from both the backup file and memory in the real machine.
e      The subcommand is instructed to terminate.

2 addr
addr   Specify the address of memory from which to read bit data.
*n     Control is returned to preprocessing indicated by the prompt number n
       (where n = 1).
e      The subcommand is instructed to terminate.

3 bit
bit1, len Data in bits whose length is specified by len is displayed, starting from the
          first bit number (bit1).
bit1-bit2 Data in bits between the first bit number (bit1) and last bit number (bit2) is
          displayed.
*n      Control is returned to preprocessing indicated by the prompt number n (a
          value between 1 and 2 can be specified as n).
e      The subcommand is instructed to terminate.
```

RESULT

The contents of memory are displayed in the following format:

a d d r	0 1 2 3 4 5 6 7 8 9 a b c d e f
0xNNNNNNNNN	c c c c c c c c c c c c c c c c

0xNNNNNNNNN: Address

c: "1," "0," or "*." When the specified bit position is out of range, an "*" is displayed.

After the contents of memory are displayed, the user is prompted for an input again in "2 addr."

NOTES

- The user can display the contents of memory within the same range as for the md subcommand.
- If "s" is specified for "1 storage" and the address at which no backup file exists is specified for "2 addr," the system displays an error message and waits for "2 addr" input.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

mcp - Copy contents of memory.

SYNOPSIS

```
mcp
1 storage (s,m,*): {s}
                  {m}
                  {*}
                  {e}
                  {return}

2 s_addr : {addr1,len}
           {addr1-addr2}
           {*n}
           {e}

3 d_addr : {addr}
           {*n}
           {e}
```

NOTE: The underlined portions are data to be entered by the user.

DESCRIPTION

mcp copies the contents of memory or backup file to a specified address.

```
1 storage (s,m,*)
s      The backup file is copied.
m      Memory in the object machine is copied.
return Memory in the object machine is copied.
*      Both the backup file and memory in the object machine are copied.
e      The end of subcommand is declared.

2 s_addr
addr1,len  A copy is made, starting from address addr1. len specifies the number
of bytes to be copied.
addr1-addr2 A copy is made, starting from address addr1 and ending at address
addr2.
*n        Control is returned to preprocessing indicated by the prompt number n
(where n = 1).
e        The end of subcommand is declared.

3 d_addr
addr      Specify the first address of the destination area to which a copy is sent.
*n        Control is returned to preprocessing indicated by the prompt number n (a value
between 1 and 2 can be specified as n).
e        The end of subcommand is declared.
```

NOTES

- The mcp subcommand displays a confirmation message as shown below after the destination address is entered in "3 d_addr" but before a copy starts.

```

Range of source addresses   Range of destination addresses
copy : 0x*****-0x***** -> 0x*****-0x*****
memory data copy ok ? (y/n) :
    
```

Message corresponding to the storage type

Storage type=s	:backup file
m	:memory
*	:memory and backup file

Enter a "y" or "Y" to start a copy. The messages below appear when a copy starts and terminates. If another character is entered, control is returned to the prompt in "2 s_addr" without a copy being made.

```

At the start of copying      *****
At the end of copying        *****
                             *****
                             memory copy start *****
                             memory copy end   *****
                             *****
                             ↑
                             Message corresponding to the storage type
    
```

- When an odd-numbered address is entered, one is subtracted to make it an even-numbered address.
- At the end of memory copying, control is returned to the prompt in "2 s_addr."
- When "s" or "*" is specified for "1 storage" and the address at which no backup file exists is specified, the system displays an error message and waits for "2 s_addr" input.
- When "s" or "*" is specified for "1 storage" and the address at which the package's shared resource is contained is specified, the system displays an error message and waits for "2 s_addr" input.
- The memory access range allowed for mcp is indicated by the shaded areas in Figure 2-6. Unlike md, bs, and bg, mcp does not access the R700 system bus space or S10 space. Further, it cannot access spaces that are not mapped in the physical memory.

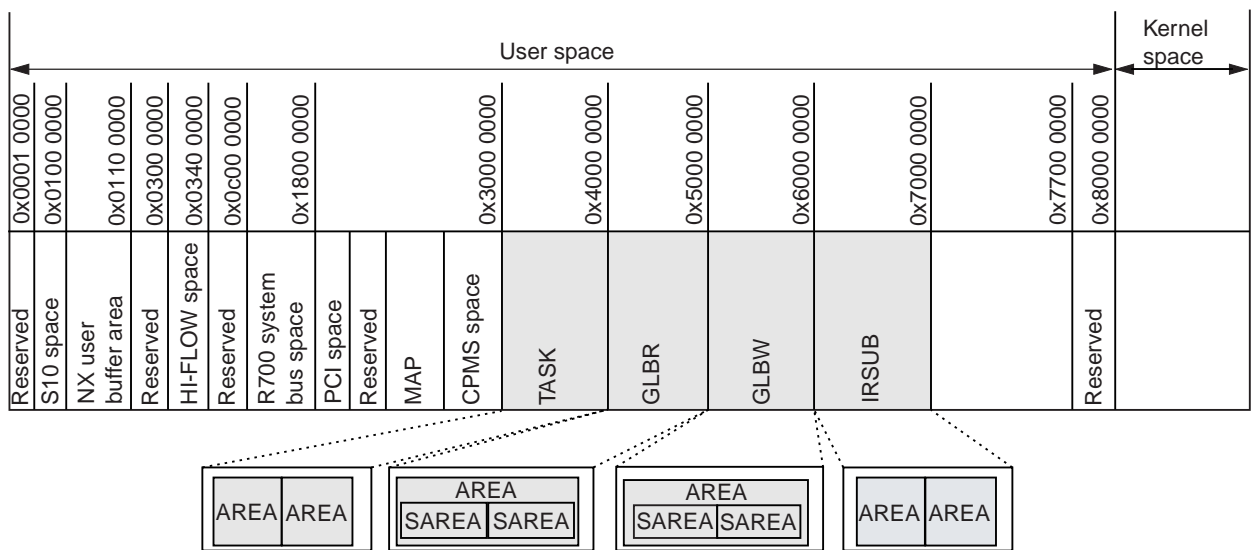


Figure 2-6 Memory Access Range

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

mmv - Move contents of memory.

SYNOPSIS

```
mmv
1 storage (s,m,*): {s}
                  {m}
                  {*}
                  {e}
                  {return}
2 s_addr : {addr1,len}
           {addr1-addr2}
           {*n}
           {e}
3 d_addr : {addr}
           {*n}
           {e}
```

NOTE: The underlined portions are data to be entered by the user.

DESCRIPTION

mmv moves the contents of memory and clears the source to zero.

```
1 storage (s,m,*)
s      The contents of the backup file are moved.
m      Contents of memory in the object machine are moved.
return Contents of memory in the object machine are moved.
*      The contents of both the backup file and memory in the object machine are
moved.
e      The end of subcommand is declared.

2 s_addr
addr1,len  A move is made, starting from address addr1. len specifies the
number of bytes to be moved.
addr1-addr2 A move is made, starting from address addr1 and ending at address
addr2.
*n        Control is returned to preprocessing indicated by the prompt number n
(where n = 1).
e        The end of subcommand is declared.

3 d_addr
addr      Specify the first address of the destination area to which to move contents.
*n        Control is returned to preprocessing indicated by the prompt number n
(a value between 1 and 2 can be specified as n).
e        The end of subcommand is declared.
```

NOTES

- The mmv subcommand displays a confirmation message as shown below after the destination address is entered in “3 d_addr” but before a move starts.

```

Range of source addresses   Range of destination addresses
-----
move : 0x*****-0x***** -> 0x*****-0x*****
memory data move ok ? (y/n) :
    
```

Message corresponding to the storage type

Storage type=s :backup file
 m :memory
 * :memory and backup file

Enter a “y” or “Y” to start a move. The messages below appear when a move starts and terminates. If another character is entered, control is returned to the prompt in “2 s_addr” without a move being made.

At the start of a move
 At the end of a move

```

*****
*****
memory copy start *****
memory copy end *****
    
```

↑
 Message corresponding to the storage type

- When an odd-numbered address is entered, one is subtracted to make it an even-numbered address.
- The memory access range allowed for mmv is the same as that for mcp.
- At the end of moving contents of memory, control is returned to the prompt in “2 s_addr.”
- If “s” or “*” is specified for “1 storage” and the address at which no backup file exists is specified, the system displays an error message and waits for “2 s_addr” input.
- If “s” or “*” is specified for “1 storage” and the address at which the package's shared resource is contained is specified, the system displays an error message and waits for “2 s_addr” input.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

mf - Set pattern data in the specified address range.

SYNOPSIS

```
mf
1 storage (s,m,*): {s}
                  {m}
                  {*}
                  {e}
                  {return}
2 addr: {addr1,len [-l/-w/-b]}
        {addr1-addr2 [-l/-w/-b]}
        {*n}
        {e}
3 data: {data}
        {*n}
        {e}
```

NOTE: The underlined portions are data to be entered by the user.

DESCRIPTION

mf sets pattern data within a given address range.

```
1 storage (s,m,*)
s      Pattern data is set in the backup file.
m      Pattern data is set in memory in the object machine.
return Pattern data is set in memory in the object machine.
*      Pattern data is set in both the backup file and memory in the object machine.
e      The end of subcommand is declared.

2 addr
addr1,len  Pattern data is set for the number of cases specified by len, starting
           from address addr1.
addr1-addr2 Pattern data is set, starting from address addr1 and ending at address
           addr2.
*n        Control is returned to preprocessing indicated by the prompt number n
           (where n = 1).
e        The end of subcommand is declared.
-l       The data length is set to 4 (bytes).
-w       The data length is set to 2 (bytes).
-b       The data length is set to 1 (byte).

3 data
data      Specify data to be set.
*n        Control is returned to preprocessing indicated by the prompt number n (a value
           between 1 and 2 can be specified as n).
e        The end of subcommand is declared.
```

NOTES

- If no data length is specified, the data length in the mf command specified last is used. The default is -1 (four bytes).
- The mf subcommand displays a confirmation message as shown below after pattern data is set in “3 data” but before the setting of the pattern data starts.

Range of addresses to set pattern data Pattern data (hexadecimal)

```
write : 0x*****-0x***** pattern data = 0x*****
memory data write ok ? (y/n) :
```

Message corresponding Storage types=s :backup file
to the storage type m :memory
 * :memory and backup file

Enter a “y” or “Y” to start the setting of pattern data. The messages below appear when the setting of pattern starts and terminates. If another character is entered, control is returned to the prompt in “2 addr” without pattern data being set.

- If the specified address is not on a data-length boundary, the value of the address is decremented accordingly.
- The specifiable address range allowed for mf is the same as the memory access range for mcp.
- At the end of pattern data setting in memory, control is returned to the prompt in “2 addr.”
- If “s” or “*” is specified for “1 storage” and the address at which no backup file exists is specified for “2 addr,” the system displays an error message and waits for “2 addr” input.
- If “s” or “*” is specified for “1 storage” and the address at which the package's shared resource is contained is specified for “2 addr,” the system displays an error message and waits for “2 addr” input.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

`e1` - Display error log.

SYNOPSIS

`e1 [-u site] [-f {s|m|l}] [-logno] [+count] [-o fname]`

DESCRIPTION

The `e1` subcommand starts the `svelog` command to display error logs. For details of the `e1` subcommand, see the command specifications of the `svelog` command.

NAME

`ss` - Display system status information.

SYNOPSIS

`ss`

DESCRIPTION

The `ss` subcommand starts the `svcpuctl` command to display system status information. For details of the `ss` subcommand, see the command specifications of the `svcpuctl` command.

NAME

st - Set the current time.

SYNOPSIS

st [yyyy.mm.dd.hh:mt:ss]

DESCRIPTION

The st sets the time that is managed by the PCs as the current time. Specify the following parameters:

yyyy	Year (four digits)
mm	Month
dd	Day
hh	Hours
mt	Minutes
ss	Seconds

NOTE: Enter the each time data in decimal.

RESULT

OK (0) Normal termination

NOTES

- When the time setting parameters (yyyy.mm.dd.hh:mt:ss) are not specified in this subcommand, the following message is displayed to prompt the user for them. If you press the [e] or [Enter] key in this state, the subcommand process terminates.

YYYY.MM.DD.HH:MT:SS :

- When this subcommand is executed directly with the -s option and an error occurs, no error message is displayed.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

br - Set/display breakpoints.

SYNOPSIS

```
br [pname break1 ... break5 [-t|-s]]
```

DESCRIPTION

br sets breakpoints or displays breakpoints that are set.

Breakpoints can only be set in a task program (TEXT space) and indirect link subprogram (TEXT space).

When break occurs in a task, the task goes into the WAIT state. In this instance, the task cyclic start timer continues running so that timer startup takes place. However, the task stays in the WAIT state unless execution is resumed from a breakpoint.

While a task is stopped due to a breakpoint, no break occurs at the other breakpoints. The other breakpoints become valid when the task stopped due to a break is resumed.

The following parameters are to be specified:

pname	Specifies the name of the program for which breakpoints are to be set.
break1-break5	Specifies breakpoints (relative addresses in a program).
-t	Indicates that the specified program name is a task's program name.
-s	Indicates that the specified program name is a subprogram's name.

NOTE: When the -t and -s options are not specified, a search will be conducted in the task and subprogram order.

RESULT

The following message appears when breakpoint setup is completed normally:

```
break point set
name = program name raddr = program internal relative address
object = machine language instruction pattern
```

When "pname" and "break" are not specified, br displays the currently set breakpoints as shown below:

```
break point
name = program name raddr = program internal relative address
object = machine language instruction pattern
name = program name raddr = program internal relative address
object = machine language instruction pattern *BREAK*
      .
      .
```

↑
This mark appears for a breakpoint at which execution is currently halted.

An error message appears if breakpoint setup is being performed by another terminal.

NOTES

- Up to five breakpoints can be set for each site.
- When a breakpoint is reached, the following message appears:

```
break!!
```

```
tn = task number  name = program name  raddr = program internal relative address
```

- If rb, rd, rr, go, or like subcommand terminates abnormally, you must issue only br to check the breakpoint status. This makes it possible to adjust the development machine's breakpoint information for the controller's if they are not in agreement.
- After a controller restart, the breakpoint information set before a controller stop will not be stored. In this instance, you must issue only br to adjust the development machine's breakpoint information for the controller's.
- Breakpoints cannot be set for the tasks and built-in subroutines that are dedicated to the system.
- This subcommand can be used when the -debug option is specified to start svdebug. When the subcommand is used in the other situations, an error occurs.
- If the execution machine differs from the development machine in breakpoint setup due to a controller restart or abnormal debugger termination, proceed as directed below:
Execute the br subcommand with no parameter specified. When any breakpoint setup remains after execution, clear it with the rb subcommand.

USAGE EXAMPLE

The breakpoint setup procedure is described below:

- The following example is used to explain about the procedure for setting a breakpoint at point 1) in situations where a task program named “prog” is generated from prog1.c and prog2.c.

prog1.c

```
main()
{
    int    a, b, c;
    int    ret;

    a = 10;
    b = 20;

    c = add( a , b );
    ans_print( c );

    exit();
}

int add( int a , int b )
{
    int    ans;

    ans = a + b; ←──────────────────────────────────①

    return(ans);
}
```

prog2.c

```
extern char    glb01_g[1024];          /* 1024byte */

void ans_print( int ans )
{
    int    ret;

    ret = rs_printf( &glb01_g[0] , "anser = %d\n" , ans );

    return;
}
```

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

- A C-language source file can be inserted into an assembler source when source compilation is performed with the “-listfile -show=source,object” option specified. prog1.lst is referenced in order to set a breakpoint in the prog1.c source. Since the C-language source for breakpoint setup is (1), the assembler instruction (2), which immediately follows (1), corresponds to the C-language source. The offset in the corresponding instruction source (prog1.c) is 0x00000014.

prog1.lst

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

FILE NAME: prog1.c

SCT OFFSET  CODE      C LABEL      INSTRUCTION OPERAND  COMMENT
-----
          prog1.c      1  main()
P 00000000          _main:                ; function: main
                                ; frame size=4
00000000 4F22                STS.L      PR,@-R15
          prog1.c      2  {
          prog1.c      3      int    a, b, c;
          prog1.c      4      int    ret;
          prog1.c      5
          prog1.c      6      a = 10;
          prog1.c      7      b = 20;
          prog1.c      8
          prog1.c      9      c = add( a , b );
00000002 E40A                MOV        #10,R4      ; H'0000000A
00000004 B006                BSR        _add
00000006 E514                MOV        #20,R5      ; H'00000014
          prog1.c     10      ans_print( c );
00000008 D204                MOV.L     L12+2,R2     ; _ans_print
0000000A 420B                JSR        @R2
0000000C 6403                MOV        R0,R4
          prog1.c     11
          prog1.c     12                exit();
0000000E D204                MOV.L     L12+6,R2     ; _exit
00000010 422B                JMP        @R2
00000012 4F26                LDS.L     @R15+,PR
          prog1.c     13      }
          prog1.c     14
          prog1.c     15      int add( int a , int b )
00000014          _add:                ; function: add
                                ; frame size=0
          prog1.c     16      {
          prog1.c     17                int    ans;
          prog1.c     18
          prog1.c     19      ans = a + b; (1)
          prog1.c     20
          prog1.c     21      return(ans);
          00000014 345C                ADD        R5,R4      (2)
          00000016 000B                RTS
          00000018 6043                MOV        R4,R0
          0000001A                L12:
          0000001A 00000002                .RES.W    1
          0000001C <00000000>                .DATA.L   _ans_print
          00000020 <00000000>                .DATA.L   _exit
          prog1.c     22      }
```


7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

- When the svload command is executed with the -P option specified for the purpose of generating an executable program, a map file will be generated.

When you note the map's SECTION=P, you will see that prog1.obj begins with 0x3003c014.

Since the source's internal relative address for breakpoint setup is 0x00000014, the breakpoint setup address is 0x3003c028 (= 0x3003c014 + 0x00000014).

The program's internal relative address can be determined from the breakpoint setup address and the program's start address and is therefore 0x00000028 (= 0x3003c028 - 0x3003c000).

prog.map

```
SECTION=P
FILE=C:\WINNT40\renix\tmp\str2545.obj
    3003c000  3003c013      14
__start
    3003c000      0 none ,g      *
L237
    3003c00c      0 none ,l      *
FILE=prog1.obj
    3003c014  3003c037      24
_main
    3003c014     14 func ,g      *
_add
    3003c028     10 func ,g      *
FILE=prog2.obj
    3003c038  3003c04f      18
_ans_print
    3003c038     18 func ,g      *
FILE=cpms_exit
    3003c050  3003c09f      50
_exit
    3003c098      0 none ,g      *
FILE=rpdp_rsprintf
    3003c0a0  3003c113      74
_rs_printf
    3003c0a0     74 func ,g      *
FILE=rpdp_rssetmsg
    3003c114  3003c16b      58
_rssetmsg
    3003c160      0 none ,g      *
FILE=sprintf
    3003c16c  3003c19b      30
_sprintf
    3003c16c     30 func ,g      *
```

NAME

rb - Reset breakpoints.

SYNOPSIS

```
rb [pname break1 ... break5 [-t|-s]]
```

DESCRIPTION

rb resets the currently set breakpoints. If no parameter is specified, rb resets all the currently set breakpoints. The following parameters can be specified:

pname	Specifies the name of the program for which breakpoints are to be set.
break1-break5	Specifies breakpoints (relative address in a program).
-t	Indicates that the specified program name is a task's program name.
-s	Indicates that the specified program name is a subprogram's name.

NOTE: When the -t and -s options are not specified, a search will be conducted in the task and subprogram order.

RESULT

The following message appears when a breakpoint reset is completed normally:

```
break point reset
name = program name  raddr = program internal relative address
object = machine language instruction pattern
```

An error message appears if breakpoint setup is being performed by another terminal.

NOTES

- This subcommand can be used when the debug option is specified to start svdebug. If the subcommand is used in the other situations, an error occurs.
- If any task is halted at a breakpoint, that breakpoint cannot be reset.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

rd - Display registers.

SYNOPSIS

rd [-f|-h]

DESCRIPTION

rd displays the contents of registers prevailing at a breakpoint. If no parameter is specified, the floating-point registers are excluded from the resulting on-screen information. The following parameters can be specified:

- f Displays the contents of floating-point registers in the form of a real number.
- h Displays the contents of floating-point registers in hexadecimal notation.

RESULT

OK(0) Normal termination.

rd displays the contents of registers as shown below.

When abnormally terminated, rd displays an error message.

```
SR =0x***** PC =0x***** GBR =0x***** PR =0x*****
MACH=0x***** MACL=0x***** FPUL=0x***** FPSCR=0x*****
R0 =0x***** R1 =0x***** R2 =0x***** R3 =0x*****
R4 =0x***** R5 =0x***** R6 =0x***** R7 =0x*****
R8 =0x***** R9 =0x***** R10 =0x***** R11 =0x*****
R12 =0x***** R13 =0x***** R14 =0x***** R15 =0x*****
```

When the -f option is specified, rd displays the contents of floating-point registers in the form of a real number as indicated below:

```
FR0 =**.*E*** FR1 =**.*E*** FR2 =**.*E*** FR3 =**.*E***
FR4 =**.*E*** FR5 =**.*E*** FR6 =**.*E*** FR7 =**.*E***
FR8 =**.*E*** FR9 =**.*E*** FR10=**.*E*** FR11=**.*E***
FR12=**.*E*** FR13=**.*E*** FR14=**.*E*** FR15=**.*E***
XF0 =**.*E*** XF1 =**.*E*** XF2 =**.*E*** XF5 =**.*E***
XF4 =**.*E*** XF5 =**.*E*** XF6 =**.*E*** XF7 =**.*E***
XF8 =**.*E*** XF9 =**.*E*** XF10=**.*E*** XF11=**.*E***
XF12=**.*E*** XF13=**.*E*** XF14=**.*E*** XF15=**.*E***
DR0 =**.*E*** DR2 =**.*E***
DR4 =**.*E*** DR6 =**.*E***
DR8 =**.*E*** DR10=**.*E***
DR12=**.*E*** DR14=**.*E***
XD0 =**.*E*** XD2 =**.*E***
XD4 =**.*E*** XD6 =**.*E***
XD8 =**.*E*** XD10=**.*E***
XD12=**.*E*** XD14=**.*E***
```

When the `-h` option is specified, `rd` displays the contents of floating-point registers in hexadecimal notation.

```

FR0 =0x*****      FR1 =0x*****      FR2 =0x*****      FR3 =0x*****
FR4 =0x*****      FR5 =0x*****      FR6 =0x*****      FR7 =0x*****
FR8 =0x*****      FR9 =0x*****      FR10=0x*****     FR11=0x*****
FR12=0x*****     FR13=0x*****     FR14=0x*****     FR15=0x*****
XF0 =0x*****      XF1 =0x*****      XF2 =0x*****      XF5 =0x*****
XF4 =0x*****      XF5 =0x*****      XF6 =0x*****      XF7 =0x*****
XF8 =0x*****      XF9 =0x*****      XF10=0x*****     XF11=0x*****
XF12=0x*****     XF13=0x*****     XF14=0x*****     XF15=0x*****
DR0 =0x*****      0x*****          DR2 =0x*****      0x*****
DR4 =0x*****      0x*****          DR6 =0x*****      0x*****
DR8 =0x*****      0x*****          DR10=0x*****     0x*****
DR12=0x*****     0x*****          DR14=0x*****     0x*****
XD0 =0x*****      0x*****          XD2 =0x*****      0x*****
XD4 =0x*****      0x*****          XD6 =0x*****      0x*****
XD8 =0x*****      0x*****          XD10=0x*****     0x*****
XD12=0x*****     0x*****          XD14=0x*****     0x*****

```

REGISTER DESCRIPTIONS

SR	Status register.
PC	Program counter.
GBR	Global base register. A register for storing the base address of a GBR indirect with displacement/GBR indirect with index for use in addressing.
PR	Procedure register used to call a subroutine. If the executed program is at the end of subroutine calling, this register stores a return address.
MACH	System register (sum-of-products upper-level register) used to store the addition value of the MAC instruction (sum-of-products operation) and the results of MAC and MUL instruction executions. When the operation result is a 64-bit value, this register stores the most significant 32 bits. When the operation result is a 32-bit value, this register stores 32 bits.
MACL	System register (sum-of-products lower-level register). When the operation result is a 64-bit value, this register stores the least significant 32 bits.
FPUL	Floating-point communication register.
FPSCR	Floating-point status/control register.
R0 - R15	General-purpose registers (the R15 is used as a stack pointer).
FR0 - FR15	Single-precision floating-point registers. Represent the <code>FPRxx_BANK0</code> value when <code>FPSCR.PR</code> (19th bit value of bits 31-0) = 0 or the <code>FPRxx_BANK1</code> value when <code>FPSCR.PR</code> = 1.
XF0 - XF15	Single-precision floating-point extended registers. Represent the <code>FPRxx_BANK1</code> value when <code>FPSCR.PR</code> (19th bit value of bits 31-0) = 0 or the <code>FPRxx_BANK0</code> value when <code>FPSCR.PR</code> = 1.
DR0 - DR15	Double-precision floating-point registers. Represent the <code>FPRxx_BANK0</code> value when <code>FPSCR.PR</code> (19th bit value of bits 31-0) = 0 or the <code>FPRxx_BANK1</code> value when <code>FPSCR.PR</code> = 1.
XD0 - XD15	Double-precision floating-point extended registers. Represent the <code>FPRxx_BANK1</code> value when <code>FPSCR.PR</code> (19th bit value of bits 31-0) = 0 or the <code>FPRxx_BANK0</code> value when <code>FPSCR.PR</code> = 1.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NOTES

- This subcommand can be used when the -debug option is specified to start svdebug. If the subcommand is used in the other situations, an error occurs.
- Even when a real-number display mode is selected, this subcommand displays the contents of registers in hexadecimal notation if the floating-point data is as indicated below. The subcommand also displays the associated character string after the hexadecimal number display.

Floating-point data	Character string	Display example	
		Single precision	Double precision
Nonnumeric	Na	FR0 =0x7fffffff:Na	DR0 =0xff00000 0x00000001:Na
Infinite	In	FR0 =0x7f800000:In	DR0 =0xff00000 0x00000000:In
Maximum expressible value	Ma	FR0 =0x7f7fffff:Ma	DR0 =0x7feffff 0xfffffff:Ma
Minimum expressible value	Mi	FR0 =0xff7fffff:Mi	DR0 =0xffefffff 0xfffffff:Mi

NAME

rr - Change the contents of registers.

SYNOPSIS

```
rr  
register : rx  
data : datax
```

NOTE: The user must enter the underlined portions.

DESCRIPTION

rr changes the contents of a register when it is halted at a breakpoint.

rx Specifies the abbreviated name of the register whose contents are to be changed. Use the abbreviated register name that appears upon rd subcommand execution.

datax Specifies the data to be changed. (Octal, decimal, hexadecimal, and real-number data can be specified.) For double-precision floating-point registers, only real-number data input is acceptable.

RESULT

OK (0) Normal termination.
When abnormally terminated, rr displays an error message.

NOTES

- The rr subcommand is valid only when a task is halted at a breakpoint.
- This subcommand can be used when the -debug option is specified to start svdebug. If the subcommand is used in the other situations, an error occurs.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

go - Resume execution from a breakpoint.

SYNOPSIS

go

DESCRIPTION

The go subcommand resumes a program at an address at which the program was halted due to a breakpoint. The breakpoint resets after the program is resumed.

RESULT

OK (0) Normal termination.

When abnormally terminated, the go subcommand displays an error message.

NOTES

- The go subcommand is valid only when a task is halted at a breakpoint.
- This subcommand can be used when the -debug option is specified to start svdebug. If the subcommand is used in the other situations, an error occurs.

NAME

as - Set/display the ADT.

SYNOPSIS

```
as
1 addr : {addr[,mask] }
          {return}
          {e}
2 mode(r,w,a) : {r}
                 {w}
                 {a}
                 {*n}
                 {return}
                 {e}
```

NOTE: The user must enter the underlined portions.

DESCRIPTION

The “as” subcommand sets the ADT (address detect trap*). When started, this subcommand displays subprompts. The input for each subprompt is explained below:

- 1 addr
 addr Specifies a logical address (in bytes) at which a trap is to be set.
 mask Specifies the mask value for logical address masking.
 10: Masks the least significant 10 bits of the address (range: 1KB).
 12: Masks the least significant 12 bits of the address (range: 4KB).
 16: Masks the least significant 16 bits of the address (range: 64KB).
 20: Masks the least significant 20 bits of the address (range: 1MB).
 e Specifies the termination of the subcommand.
 return Displays the current ADT settings in the following form:

```
address detect trap
addr = 0x*****-0x*****   mode = *****
```

- 2 mode(r,w,a)
 r Specifies a read trap.
 w Specifies a write trap.
 a Specifies a read/write trap.
 return Specifies a read/write trap.
 *n Returns to an earlier process having the prompt number n (n is acceptable only when it is 1).
 e Specifies the termination of the subcommand.

* The term “ADT” is an acronym for “Address Detect Trap.” It represents a function that links with built-in subroutine ADTS to output an error log when a specified address is accessed.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

Byte access, word access, and longword access to a preselected logical address is targeted for the ADT.

When the logical address set by the ADT is accessed in a specified mode, built-in subroutine ADTS is linked for error logging.

RESULT

When the process terminates normally, the following message appears on the display:

```
address detect trap set
addr = 0x*****-0x***** mode = *****
```

Displays the address range over that the ADT is set.

Displays the mode that is set for the ADT.

Mode = read: read trap
 write: write trap
 access: read/write trap

NOTES

- The ADT can be set at only one place per site.
- The ADT does not reset when the CPU restarts.
- Only the access via the processor’s MMU can be detected by the ADT. Therefore, the ADT cannot detect the access via the RPL/RRB or the debugger’s memory access subcommand that uses a physical address for accessing purposes.
- The ADT set up by the “as” subcommand can be reset by executing the svcputl command with the -time option specified.
- The ADT can be set for shaded areas shown below. Note, however, that the ADT cannot be set for unmapped areas.
- Never access an address set up by the ADT within built-in subroutine ADTS. Failure to observe this precaution will cause the system to go down.

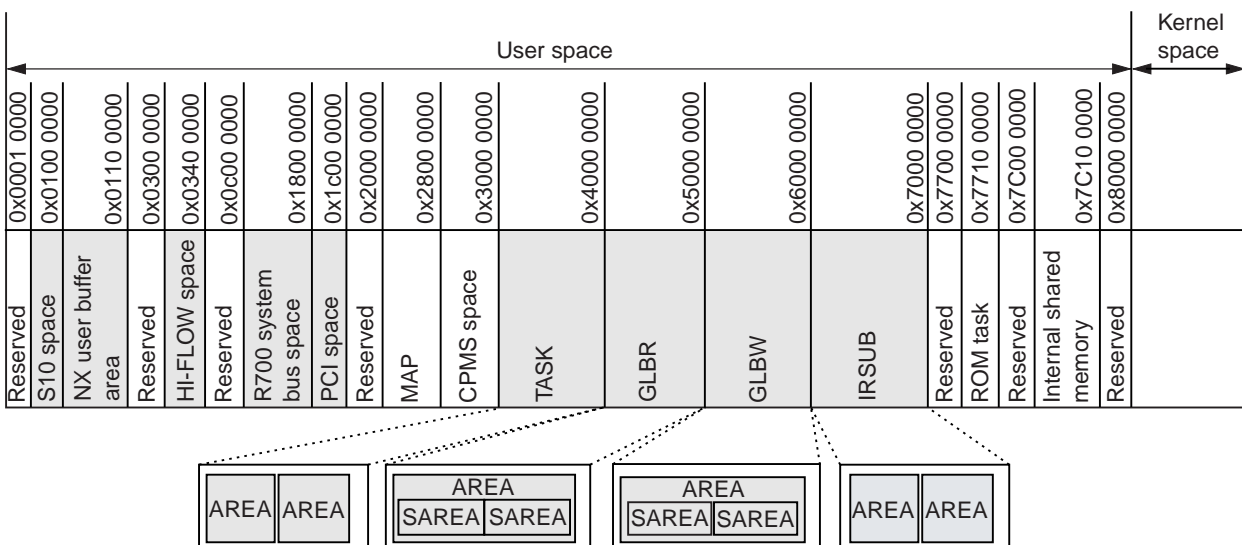


Figure 2-7 ADT Setup Areas

NAME

ac - Reset the ADT.

SYNOPSIS

ac

DESCRIPTION

ac clears the ADT (Address Detect Trap) setup.

RESULT

When the ADT is reset normally, the following message appears on the display:

```
address detect trap reset
addr = 0x*****-0x*****   mode = *****
```

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

gt - Display the current time.

SYNOPSIS

gt

DESCRIPTION

The st displays current time managed by the PCs.

RESULT

The current time is displayed in the following format:

```
yyyy.mm.dd.hh:mt:ss
```

yyyy Year (four digits)

mm Month

dd Day

hh Hours

mt Minutes

ss Seconds

NAME

ld - Transfer the contents of a backup file to memory in the controller.

SYNOPSIS

```
ld {-t tname}
ld {-s sname}
ld {-g gname}
ld {-a aname}
ld {-m addr, len}
ld {-T [tno]}
ld {-U [point, ent]}
ld {-S [sno]}
ld {-G [gno]}
ld {-f fname}
```

DESCRIPTION

The ld subcommand transfers the contents of a backup file to memory in the PCs, the file that is specified by one of the options listed below.

- | | |
|--------------|---|
| -t tname | The program specified by tname is downloaded. A task is created for tname on the PCs. When the program specified by tname is a multitask program, as many tasks as required are also created on the PCs. |
| -s sname | The subprogram specified by sname is downloaded together with the associated entry in the indirect link resident subprogram address table or the associated entry in the built-in subroutine management table. When the subprogram specified by sname is registered as a multi-entry IRSUB, all the entries in the indirect link resident subprogram address table corresponding to the IRSUB entry points are downloaded. |
| -g gname | The global data specified by gname is downloaded. When the global data is indirect global data, the associated entry for the indirect link global address table is also downloaded. |
| -a aname | The contents of the split area specified by aname are downloaded. The split area specified by aname must have been defined in the GLB area. |
| -m addr, len | Data of len bytes is downloaded, starting from the first address specified by addr. When the specified address range includes a non-allocate space, downloading does not take place. |
| -T [tno] | The task having the task number specified by tno is deleted from or created on the PCs. If tno identifies a task that has been created on the development machine, this option creates a task for tno. If tno identifies a task that has not been created on the development machine, the option deletes the task. When tno is omitted, a list is displayed that shows the tasks for which creation or deletion on the development machine is not reflected on the PCs. |

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

-U [point, ent] The entry for the built-in subroutine management table specified by point and ent is downloaded. Specify point and ent as follows:

point: Specifies the character string that indicates a point at which a built-in subroutine is to be incorporated.

ent: Entry number from 1 to 4

When both point and ent are omitted, a list is displayed that shows the built-in subroutine management table entries for which registration or deletion on the development machine is not reflected on the PCs.

-S [sno] The indirect link resident subprogram address table entry specified by sno is downloaded. When sno is omitted, a list is displayed that shows the indirect link resident subprogram address tables for which registration or deletion on the development machine is not reflected on the controller.

-G [gno] The indirect link global address table specified by gno is downloaded. When gno is omitted, a list is displayed that shows the indirect link global address tables for which registration or deletion on the development machine is not reflected on the PCs.

-f fname The contents of the file specified by fname are downloaded. Only the name of the file output by the sv subcommand can be specified in fname.

RESULT

When resources are downloaded by ld, the result is displayed upon completion of downloading. The following paragraphs explain the results of ld with regard to each option.

(1) -t tname (downloading an individual program)

The first line shows the main unit address. The second and subsequent lines show the TCB addresses at which a download was performed due to task generation/deletion. For a multitask, the display shows all the TCB addresses for downloading.

```
address : 0x*****-0x*****
address : 0x*****-0x*****
```

When a split area or the BSS area of the program was allocated or deallocated or a task was created or deleted, the following message is displayed:

```
allocate : ANAME (0x*****-0x*****)
free : ANAME (0x*****-0x*****)
allocate : bss (0x*****-0x*****)
free : bss (0x*****-0x*****)
ctask : TN (STACK=0x*****-0x*****)
dtask : TN (STACK=0x*****-0x*****)
```

ANAME is the name of the allocated or deallocated split area, while TN is the number of the created or deleted task.

(2) -s sname (downloading an individual subprogram)

The result is displayed in a different way according to the type of the downloaded subprogram.

● IRSUB

The range of addresses where the IRSUB was downloaded is displayed in the format below. The first line shows the address of the subprogram, while the second line shows the address of the indirect link resident subprogram address table. When the IRSUB has multiple entries, the addresses of all indirect link resident subprogram address tables to be downloaded are displayed.

```
address : 0x*****-0x*****
address : 0x*****-0x*****
```

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

- Built-in subroutine

The range of addresses where the subroutine was downloaded is displayed together with the entry for the built-in subroutine management table, in the format below. “address” is the address of the downloaded built-in subroutine, while “point, ent” is the include point representing the downloaded entry for the built-in subroutine management table and the entry number.

```
address : 0x*****-0x*****
point,ent : POINT,N
```

At POINT, one of the following character strings, each of which represents an include point, is displayed: CPES, IES, EAS, INS, EXS, ABS, PCKS, PIOS, MODES, WDTES, NXS, XEAS, BRKS and ADTS. At N, an entry number from 1 to 4 is displayed.

(3) -g gname (downloading individual global data)

The range of addresses where global data was downloaded is displayed in the format below. When the global data is indirect link global data, the address of the indirect link global address table is also displayed. When the global data specified by gname is registered as multiple indirect link global data items, the addresses of all indirect link global address tables are displayed.

```
address : 0x*****-0x*****
```

(4) -a aname (downloading an individual split area)

The range of addresses where the split area was downloaded is displayed in the following format:

```
address : 0x*****-0x*****
```

(5) -m addr, len (address-based downloading)

The range of addresses where downloading was performed is displayed in the following format:

```
address : 0x*****-0x*****
```

(6) -T tno (reflecting task creation or deletion in memory in the PCs)

The result is displayed in a different way according to what was reflected.

- When task generation/deletion is reflected in the PCs memory

The display shows the TCB addresses at which a download was performed due to task generation/deletion.

```
address : 0x*****-0x*****
```

- When a list of the tasks requiring reflection of their memory images in memory in the PCs whenever necessary is requested to be displayed:
When only the -T option is specified, a list is displayed in the format below, the list that shows the tasks which need to be reflected in memory in the PCs.

```
TN  TNAME  PNAME  STATUS
tn  tname  pname  stat
```

tn, tname, pname, and stat in the above list represent task number, task name, program name, and management state of task. Table 2-13 explains the management states. When different TNs are used for a task in the unmatched management state on the development machine and that on the controller, the TN on the development machine is displayed.

Table 2-13 Management States of Resources

Management state	Description
non-exist	Registered in neither the development machine nor the PCs.
not-build not-build2	Loaded by a subprogram but not built. If “dbuild” is performed after downloading into the PCs, “not-build2” results.
defined-POC	Registered in the development machine only.
defined	Registered in both the development machine and PCs.
defined-CON	Registered in the PCs only. Deleted from the development machine after downloading into the PCs.
unmatch	Registered in both the development machine and PCs but mismatched. Deleted or reregistered at the development machine but not downloaded.

- (7) -U point, ent (downloading a built-in subroutine management table)
- When a built-in subroutine management table was downloaded:
The entry of the updated built-in subroutine management table is displayed in the following format:

```
address : 0x*****-0x*****
```


7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

- When a list of the built-in subroutine management tables requiring reflection in memory in the PCs is requested to be displayed:
When only the -U option is specified, a list is displayed in the format below, the list that shows the built-in subroutine management tables which need to be reflected in memory in the controller.

POINT	ENT	SUBNAME	STATUS
pnt	eno	subname	stat

pnt, eno, subname, and stat in the above list represent an include point, entry number, subprogram name, and management state of the built-in subroutine management table entry. Table 2-13 explains the management states.

As the management state of the built-in subroutine management table entry, defined-POC, defined-CON, not-build, not-build2, or unmatched is displayed.

When different include points are used for a built-in subroutine management table entry in the unmatched state on the development machine and that on the PCs, the include point on the development machine is displayed.

(8) -S sno (downloading an indirect link resident subprogram address table)

- When an indirect link resident subprogram address table was downloaded:
The address of the downloaded indirect link resident subprogram address table is displayed in the following format:

```
address : 0x*****-0x*****
```

- When a list of the indirect link resident subprogram address tables requiring reflection in memory in the PCs is requested to be displayed:
When only the -S option is specified, a list is displayed in the format below, the list that shows the indirect link resident subprogram address tables which need to be reflected in memory in the PCs.

IRSBNO	ENTNAME	SUBNAME	STATUS
sno	entname	subname	stat

sno, entname, subname, and stat in the above list represent an IRSUB number, entry name, subprogram name, and management state of the IRSUBT entry. Table 2-13 explains the management states.

When different IRSUB numbers are used for an IRSUBT entry in the unmatched state on the development machine and that on the PCs, the IRSUB number on the development machine is displayed. When different subprogram names are used, the subprogram name on the development machine is also used.

(9) -G gno (downloading an indirect link global address table)

- When an indirect link global address table was downloaded:

The address of the downloaded indirect link global address table (IRGLBT) is displayed in the following format:

```
address : 0x*****-0x*****
```

- When a list of the indirect link global address tables requiring reflection in memory in the PCs is requested to be displayed:

When only the -G option is specified, a list is displayed in the format below, the list that shows the indirect link global address tables which need to be reflected in memory in the controller.

IRGLBNO	ENTNAME	SNAME	STATUS
gno	entname	sname	stat

gno, entname, sname, and stat in the above list represent an indirect link global number, entry name, secondary partition area, and management state of the indirect link global address table entry. Table 2-13 explains the management states.

When different indirect link global numbers are used for an indirect link global address table entry in the unmatched state on the development machine and that on the PCs, the indirect link global number on the development machine is displayed.

When different secondary partition area names are used, the secondary partition area name on the secondary partition area is also used.

(10) -f fname (downloading from the specified file)

The range of addresses where downloading was performed is displayed in the following format:

```
address : 0x*****-0x*****
```

NOTES

- Before replacing a resource, abort all tasks that reference the resource, or take a similar action, so that the replacement will not cause a problem.
- When a resource is downloaded with the -a or -m option, be sure to download the management table with the -T, -U, -S, or -G option and also create or delete tasks.
- If a resource or area containing no backup file is specified without specifying the file name with the -f option, the system displays an error message and terminates the subcommand.
- If the failure recovery process is skipped, the ld subcommand's individual loading function cannot be exercised until batch loading is performed by the svrpl command.
- Programs and subprograms for which a breakpoint is set cannot be downloaded.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

sv - Back up resources individually.

SYNOPSIS

```
sv {-t tname [-f fname]}
sv {-s sname [-f fname]}
sv {-g gname [-f fname]}
sv {-a aname [-f fname]}
sv {-m addr, len [-f fname]}
```

DESCRIPTION

The sv subcommand transfers the contents of memory in the PCs to the backup file as specified by the options listed below.

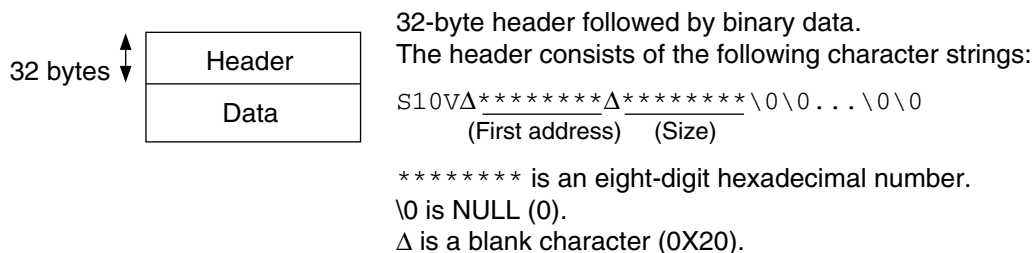
- tname The text and data sections of the program specified by tname are transferred.
- s sname The text and data sections of the subprogram specified by sname are transferred.
- g gname The contents of the secondary partition area specified by gname are transferred.
- a aname The contents of the split area specified by aname are transferred. The split area specified by aname must have been defined in the GLB area.
- m addr, len Data is transferred for the number of bytes specified by len, starting from the first address specified by addr. When the specified address range includes a non-mapped space, a transfer does not take place.
- f fname Data is transferred to the file specified by fname. When this option is omitted, data is transferred to the backup file. When an error is detected during saving, the specified file is deleted and this subcommand is terminated.

RESULT

The range of addresses where data was transferred is displayed in the following format:

```
address : 0x*****-0x*****
```

When transferring to a file other than the backup file, make sure that the file is in the following format:



NOTES

- If an area containing no backup file is specified without specifying the file name with the -f option, the system displays an error message and terminates the subcommand.
- If the package's shared source is specified without specifying the file name with the -f option, the system displays an error message and terminates the subcommand.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

cm - Compare the contents of a backup file with those of memory in the PCs.

SYNOPSIS

```
cm {-t tname}
cm {-s sname}
cm {-g gname}
cm {-a aname}
cm {-m addr, len}
cm {-f fname}
```

DESCRIPTION

The cm subcommand compares the contents of a backup file with those of memory in the PCs. Use the following options to specify the comparison target.

-t tname	The text and data sections of the program specified by tname are compared.
-s sname	The text and data sections of the subprogram specified by sname are compared.
-g gname	The contents of the secondary partition area specified by gname are compared.
-a aname	The contents of the split area specified by aname are compared. The split area specified by aname must have been defined in the GLB area.
-m addr, len	Data is compared for the number of bytes specified by len, starting from the first address specified by addr. When the specified address range includes a non-allocate space, a comparison does not take place.
-f fname	The file specified by fname is compared with the contents of memory. (Only the file to which an output is made by the sv subcommand can be specified.)

RESULT

When the comparison result is normal, the address range is displayed in the following format:

```
address : 0x*****-0x*****
++ compare OK ++
```

When a mismatch is found during comparison, the result is displayed one word (two bytes) at a time, as shown below.

```
address : 0x*****-0x*****
address = 0x***** memory data = 0x**** backup data = 0x****
```

NOTE

If an area containing no backup file is specified without specifying the file name with the -f option, the system displays an error message and terminates the subcommand.

NAME

dr - Enable DHP logging.

SYNOPSIS

dr

DESCRIPTION

The dr subcommand starts the dhphr command to enable DHP logging. For details on the dr subcommand, see the specifications for the svdhp command's -on option.

NAME

ds - Disable DHP logging.

SYNOPSIS

ds

DESCRIPTION

The ds subcommand starts the dhphr command to disable DHP logging. For details on the ds subcommand, see the specifications for the svdhp command's -off option.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

svdhp - Display the DHP.

SYNOPSIS

```
svdhp [-u site] [+count] [-on|-off] [-o fname] [-f fname] [-d fname]
```

DESCRIPTION

The svdhp subcommand starts the svdhp command to display the DHP.

For details on the svdhp subcommand, see the section on the svdhp command.

NAME

svadm - Display a resource name for an address.

SYNOPSIS

```
svadm [addr] [-u site]
```

DESCRIPTION

The svadm subcommand starts the svadm command to display the name and other information about a specified logical address.

For details on the svadm subcommand, see the section on the svadm command.

NAME

si - Initialize the stack.

SYNOPSIS

```
si tn1[-tn2] [,data]
si tname [,data]
```

DESCRIPTION

The si subcommand initializes the stack of the specified task with a fixed pattern. Specify the following parameters:

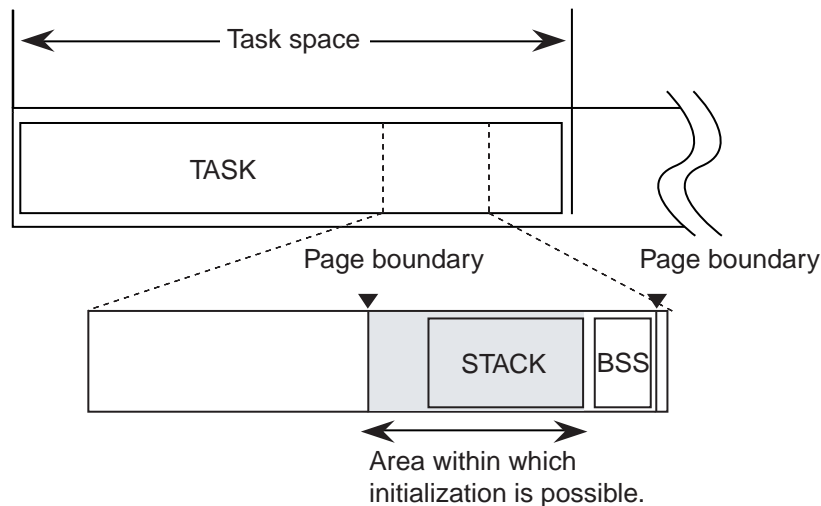
tn1	First task number (1 to maximum task number)
tn2	Last task number (1 to maximum task number)
tname	Task name
data	Initialization data (0 to 0xf)

RESULT

OK (0) Normal termination

NOTES

- When nothing is specified in data, the stack is initialized with all fs.
- The task for which initialization is to be performed must be in the DORMANT state.
- The stack of the specified task can be initialized only within the range of the first address of the page on which the stack is present to the last address of the stack (see figure below).



- When this subcommand is started with no parameters specified or with invalid arguments specified, the message shown below is displayed to prompt for parameters. Enter correct parameters after the colon (":"). If you press the [e] or [Enter] key in this state, the subcommand process terminates.

```
input tn1[-tn2] [,data] or tname [,data]
:
```


7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

`sp` - Display the amount of stack use.

SYNOPSIS

```
sp tn1[-tn2] [,data]
sp tname [,data]
```

DESCRIPTION

The `sp` subcommand displays the size of the stack used by the specified task. Specify the following parameters:

`tn1` First task number (1 to maximum task number)
`tn2` Last task number (1 to maximum task number)
`tname` Task name
`data` Check pattern (0 to 0xf)

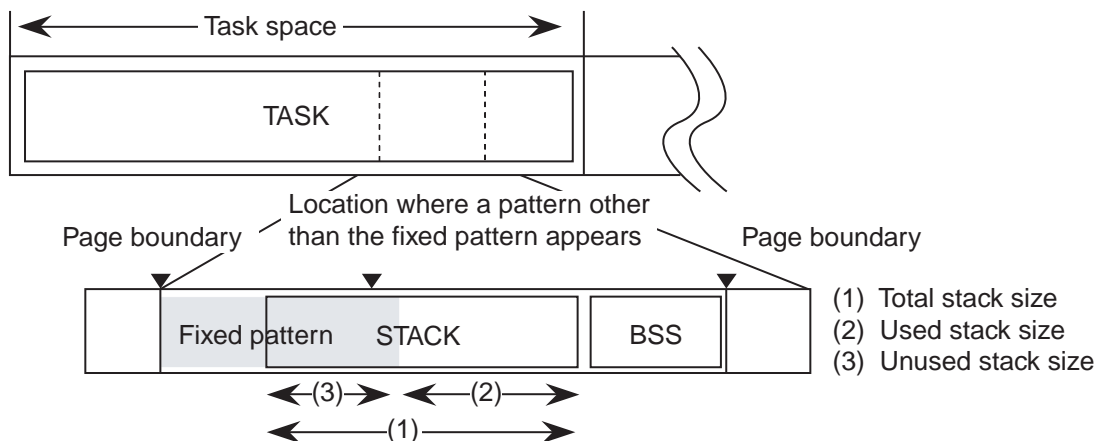
RESULT

The size of the stack used by the task is displayed in the following format:

<u>tn=***</u>	<u>total:*****bytes</u>	<u>use:*****bytes</u>	<u>rest:*****bytes</u>
Task number	Total stack size	Used stack size	Unused stack size

NOTES

- The check pattern specified in `data` must be the same as the initialization data specified in the `si` subcommand. When check pattern parameter (`data`) is omitted, an “0xf” is assumed.
- The used stack size is calculated from the address at which a pattern, other than the check pattern specified in the `data` parameter, appears in the page area in the stack used by the specified task. For this reason, if the pattern at the beginning of the stack is the same as the check pattern, the used stack size cannot be displayed correctly.
- The figure below shows the relationship between the information displayed by `sp` and the task operation space.



7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

- When this subcommand is started with no parameters specified or with invalid arguments specified, the message shown below is displayed to prompt for parameters. Enter correct parameters after the colon (":"). If you press the [e] or [Enter] key in this state, the subcommand process terminates.

```
input tn[-tn2] [-data] or tname [-data]
:
```

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

`ps` - Start displaying a debug statement.

SYNOPSIS

`ps`

DESCRIPTION

`ps` instructs that the messages output by `rs_printf()` within a program should start appearing on the terminal's display.

Debug statements output prior to the execution of the `ps` subcommand do not appear on the display.

(For `rs_printf()`, see Appendix B, "Libraries.")

RESULT

The debug statement display sequence starts after normal process termination.

NOTE

If the buffer for debug statement storage is not sufficient, the subcommand may fail to output debug statements.

NAME

`pe` - Finish displaying a debug statement.

SYNOPSIS

`pe`

DESCRIPTION

`pe` instructs that the messages output by `rs_printf()` within a program should finish appearing on the terminal's display. (For `rs_printf()`, see Appendix B, "Libraries.")

RESULT

The debug statement display sequence finishes after normal process termination.

NAME

ver - Display the version of the CMU.

SYNOPSIS

ver

DESCRIPTION

ver displays the version of the CMU.

RESULT

The version of the CMU appears in the following format upon normal process termination.

CMU Ver.** Rev.**

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

help - Display a list of subcommands.

SYNOPSIS

help

DESCRIPTION

help displays a list of svdebug subcommands.

The help subcommands list the subcommand names together with an outline of their functions, in the following format:

<Sub Command>	<Function>
qutask queue
abtask abort
retask release
tatask status
sutask suspend
rstask resume
tmtask timer request
cttask cancel timer
mdmemory print/patch appointed address
sdmemory print/patch appointed name
mcpmemory copy
mmvmemory move
mfmemory fill
bsbit data set
bgbit data get
elsystem error display
sssystem status display
stset timer
gtget timer
brbreak point set
rbbreak point reset
rdregister print
rrregister set
gobreak point restart
asadress detect trap set
acadress detect trap reset
ldmemory load
svmemory save
cmmemory compare
drDHP regist start
dsDHP regist stop
svdhpDHP data display
svadmaddress -> sarea name
sistack initial
spstack print
psdebug message print start
pedebug message print end
verCMU version print
qdebughr end
!execute external command
helpcommand menu display

NAME

q - Terminate the debugger.

SYNOPSIS

q

DESCRIPTION

The q subcommand terminates the debugger.

However, if a breakpoint is set, the subcommand displays it and waits for key input.

NOTE

If the message “A breakpoint is set” appears, reset it using the go subcommand if execution is halted at a breakpoint or using the rb subcommand if execution is not halted at a breakpoint, and then reissue the q subcommand.

NAME

! - Execute a command on the development machine at the time of svdebug execution.

SYNOPSIS

! commands_on_development_machine

DESCRIPTION

The ! subcommand executes the character strings after the “!” as commands on the development machine.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

svelog - Display error log.

SYNOPSIS

```
svelog [-u site] [-f format] [-logno] [+case] [-d fname] [-o fname]
```

```
    )  
    Error log for one screen is displayed.  
    )  
    {p}  
    {-}  
    {±nl}  
    {n}  
    {Blank}  
    {q}
```

DESCRIPTION

The svelog command reads error log from the error log buffer in the controller, and displays the error log. The following options can be specified:

- u site Specify the name of the site to be handled. When this option is omitted, the site set in the RSSITE environment variable is used.
- f format Specify one of the following output formats for error log display. The following formats are available. The default format is m.
 - s: Error log is displayed in a simplified format.
 - m: Complete error log is displayed.
 - l: Error log is displayed together with DHP traces.
- logno The error log having the error log number specified by logno is displayed.
- +case Specify the number of log cases to be displayed. When this option is omitted, the complete error log is displayed, starting from the most recent case.
- d fname Specify the name of the file in which to store the screen operation history (operation results). When a file name already in use is specified, the screen operation history is added to the file.
- o fname Specify the name of the file in which to store the error log. When a file name already in use is specified, the file is deleted and a new file is created.

The following display commands are supported to control error log display:

- p, blank The next page is displayed.
- The previous page is displayed.
- ±nl Error log display starts from the line specified by nl before (when a “-” is specified) or after (when a “+” is specified) the current line.
- n Error log display starts from the *n*th line.
- q Error log display is terminated.

NOTES ON USE

- svelog is operable when the user task is in the RUN or STOP state.
- When the log number specified by -logno is smaller than the log number of the most recent log, the most recent log is displayed.
- When both -logno and +case are specified, error log is displayed by the number of cases specified by +case, starting from the case having the log number specified by logno.
- If the -f format is not specified, the “m” format is used.

TERMINATION CODE

Returns one of the following termination codes:

- 0: Normal termination
- 1: Abnormal termination
- 2: Communication error
- 3: Signal received

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

svdhp - Display DHP traces.

SYNOPSIS

```
svdhp [-u site] [+count] [-on|-off] [-d fname] [-o fname]
```

One screen of DHP trace information is displayed.

```
{
  {p}
  {-}
  {±nl}
  {n}
  {Blank}
  {q}
```

DESCRIPTION

The svdhp command displays DHP traces stored in the DHP trace buffer in the PCs sequentially from the most recent DHP trace. The following options can be specified:

- u site Specify the name of the site to be handled. When this option is omitted, the site set in the RSSITE environment variable is used.
- +count The trace specified by count is displayed. When this option is omitted, all traces are displayed.
- on DHP logging is enabled.
- off DHP logging is disabled.
- d fname Specify the name of the file in which to store the screen operation history (operation results). When a file name already in use is specified, the screen display is added to the file.
- o fname Specify the name of the file in which to store the displayed DHP trace information. When a file name already in use is specified, the file is deleted and a new file is created.
- f fname Specifies the DHP log input file name.
As the site name, specify the S10V site name.
The input file is acceptable only when it stores a DHP log in a GLB with the dhpread macro and stores the GLB's DHP log with the debugger's sv subcommand.
For details on the dhpread macro, refer to the S10V Software Function Specification (CPMS) (Drawing No. 331SR26517).

The following display commands are supported to control DHP trace display.

- p, blank The next page is displayed.
- The previous page is displayed.
- ±nl DHP trace display starts from the line specified by nl before (when a “-” is specified) or after (when a “+” is specified) the current line.
- n DHP trace display starts from the *n*th collection of line.
- q DHP trace display is terminated.

7. svdebug (ONLINE DEBUGGER) AND DEBUG SUPPORT COMMANDS

NAME

svcpunow - Display the PU load ratio.

SYNOPSIS

```
svcpunow [-u site] [-t second]
```

DESCRIPTION

The svcpunow command fetches the accumulated idle time and the time of the day from the specified site (PU), and displays the PU load ratio.

Calculation expression

$$\text{PU load ratio} = (\text{Measurement time} - \text{Idle time}) / \text{Measurement time}$$

The following options can be specified:

- u site Specify the name of the site to be handled. When this option is omitted, the site set in the RSSITE environment variable is used.
- t second Specify the length of time in seconds during which the PU load ratio is measured, within the range of 1 to 3,600. The default is 1 (second).

NOTE ON USE

An attempt to execute a svcpunow command fails if another cpunowhr command is already in service.

TERMINATION CODE

Retruns one of the following termination codes:

- 0: Normal termination
- 1: Abnormal termination
- 2: Communication error
- 3: Signal received

OUTPUT FORMAT

The following output results:

```
2002/04/24 17:57:33 SITE=pcs01b_cp ** 1 second wait **  
CPU(pcs01b_cp) load ratio = 0.06%
```

NAME

svtimex - Display the task activity ratio.

SYNOPSIS

```
svtimex [-u site] [tn/tname] [-t second]
```

DESCRIPTION

The svtimex command fetches the number of task executions and the accumulated execution time in the measurement time as well as the time of day. According to them, timexhr displays the task activity ratio.

The following options can be specified:

- u site Specify the name of the site to be handled. When this option is omitted, the site set in the RSSITE environment variable is used.
- tn Specify the task number from 1 to 255 in decimal or hexadecimal. (To specify a hexadecimal number, prefix it with "0x.")
- tname Specify the task name.
When neither tn nor tname is specified, the user is prompted to enter a measurement time interactively. Enter a measurement time within the range of 1 to 86,400. Then the user is prompted to enter task names or task numbers. Settings for up to 10 tasks are possible. To execute the svtimex command after entry of a task name or number, just press the [Enter] key without entering task names or task numbers.
- t second Specify the length of time in seconds during which the task activity ratio is measured, within the range of 1 to 86,400. The default is 1 (second).

NOTES ON USE

- When the -t option is used to specify a measurement time, be sure to specify the task number (tn) or task name (tname) together.
- An attempt to execute a svtimex command fails if another timexhr is already in service.
- The tn and tname options are mutually exclusive. They cannot be specified together. Interactively, up to 10 task names or task numbers can be specified.

TERMINATION CODE

Retruns one of the following termination codes:

- 0: Normal termination
- 1: Abnormal termination
- 2: Communication error
- 3: Signal received

OUTPUT FORMAT

The following output results:

```
2002/04/24 18:02:18 SITE=pcs01b_cp ** 1 second wait **
sist(255) load ratio=0.00% execute count=0 total time=0.000sec average time=0.000sec
```

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIXES

APPENDIX A NAMES USABLE IN PROGRAMS

When using a program that has the same name as a subroutine provided in the system in advance, be careful. All subroutines provided in the system are contained in the library files. These subroutines can be linked simply by executing svload with the -l option specified. When linking a program that has the same name as a system subroutine, be sure to specify the object file in which the subroutine is defined, in an argument of svload. Otherwise, the subroutine that has the same name will be linked from the library file.

Library files provided for each system and names defined there are listed below. In programming, take care so that the same name may not be duplicated. If it is unavoidable to use the same name, specify the library file after the object file to be linked. The subroutine having the same name will not be linked from the library file.

In the listings below, the subroutines provided in the system are grouped for each library. The names starting with an underscore (“_”) are reserved for the system. Do not use them.

The following table shows the supported libraries.

Library name	Library description	Remarks
libsh4nbmzz.lib	Subroutines for C language Denormalized numbers: Denormalized numbers Value rounding: Disregarding	For details, refer to the documentation supplied with the shc compiler.
libsh4nbmdn.lib	Subroutines for C language Denormalized numbers: 0 Value rounding: Disregarding	
libfirad.lib	Indirect link address reference subroutines	Refer to the respective documentation.
libcpms.lib	CPMS macro linkage subroutines	
libsysctl.lib	Subroutines for system control	
libcpr_hr.lib	Remote CPU control subroutines	
libcyem.lib	Cyclic communication subroutines	
libnet.lib	Socket communication subroutines	

A. NAMES USABLE IN PROGRAMS

<libsh4nbmzz.lib>

<libsh4nbmdn.lib>

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

<libfirad.lib>

irglbad	irsubad
---------	---------

<libcpms.lib>

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

A. NAMES USABLE IN PROGRAMS

<libsysctl.lib>

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

<libcpr_hr.lib>

cpr_grstart	cpr_stop	cpr_rpl	cpr_rrb	cpr_chk
cpr_setipa	setipa_mkdata	setipa_parchk	setipa_sndrcv	setipa_dupchk
setipa_inet_addr	setipa_send	setipa_headchk	setipa_rcv	

<libcycm.lib>

getcycm_hr	restcycm_hr	wcycm_hr	cycinf_hr	stcycm_hr
rcycm_hr				

<libnet.lib>

accept	bind	connect	getsockopt	listen
recv	recvfrom	send	sendto	setsockopt
shutdown	socket			

APPENDIX B LIBRARIES

(1) Conditions for specifying library files

When specifying library files with `svload`, specify library names as shown in Table B-1.

Table B-1 Conditions for Specifying Library Names

Condition	Library name	Specification in <code>loadhr</code>	Remarks
Programs are coded in C. (Functions in the pertinent library shown in Appendix A are used.)	<code>libcvt.lib</code> <code>libmrs.lib</code> <code>libcrs.lib</code>	<code>-lcvt</code> <code>-lmrs</code> <code>-lcrs</code>	Refer to the “CPMS General Description and Macro Specifications” (Manual number SVE-3-201).
Created programs use CPMS macros.	<code>libcpms.lib</code>	<code>-lcpms</code>	
Indirect link addresses or indirect link subroutines are referenced.	<code>libfirad.lib</code>	<code>-lfirad</code>	See (3), “Subroutines that reference indirect link addresses.”
User-specific libraries are used.	<code>user_library</code>	<code>-l character_string</code> or <code>library_name</code>	

(2) Library specification order

When specifying libraries in `svload`, note the following points:

- Specify libraries containing common subroutines later as far as possible.
- When the same name is duplicated in more than one specified library, place the library holding the object file to be linked at the top of them.
- Specify system libraries in the following order:
 - When the `libcvt.a` library is used
`libcvt.lib`
`-lcvt -lmrs -lcrs`
 - When the `libmrs.a` library is used
`libmrs.lib`
`-lmrs -lcrs`
 - When the `libcrs.a`, `libcpms.a`, or `libfirad.a` library is used
`libcrs.lib`
 Specify only `-lcrs`, `-lcpms`, or `-lfirad`.

B. LIBRARIES

(3) Subroutines that reference indirect link addresses

When using the following module, link with libfirad.lib.

NAME

irglbad

SYNOPSIS

```
int *irglbad(no)
int no;
```

DESCRIPTION

When an indirect link global number (1 to maximum global number) is specified in no, the irglbad subroutine returns the corresponding global address.

RETURN CODES

When a registered indirect link global number is specified in no, irglbad returns the corresponding global address. When an unregistered indirect link global number is specified in no, irglbad returns a value of 0. (The first address of the indirect link global address management table is 0xafd00004 when a 1 is specified in no. Each time the value of n is increased by one, a 4 is added to the first address as the indirect link global management table address corresponding to no.)

NAME

irsubad

SYNOPSIS

```
int *irsubad(no)
int no;
```

DESCRIPTION

When an IRSUB number (1 to the maximum IRSUB number) is specified in no, the irsubad subroutine returns the corresponding IRSUB address.

RETURN CODES

When a registered IRSUB number is specified in no, irsubad returns the corresponding IRSUB address. When an unregistered IRSUB number is specified in no, irsubad returns a value of 0.

(4) Message output routine

When using the following module, link with `libcpms.lib`.

NAME

`rs_printf`

SYNOPSIS

```
int rs_printf (buf, fmt, p1, p2, ...,p10)
char *buff;
char *fmt;
long p1, p2, ...,p10;
```

DESCRIPTION

`rs_printf` converts data to a message in a specified format and writes the message into the message buffer area within the OS.

The message can be read using the `ps` subcommand of the `debughr` command.

When specifying, ensure that the converted message character string size is between 1 and 1024 bytes. If the converted message character string size is outside the range of 1 to 1024 bytes, a parameter error occurs so that `rs_printf` cannot perform a write.

PARAMETERS

<code>buff</code>	Specifies the start address of a memory area into which the message to be written is to be temporarily stored.
<code>fmt</code>	Specifies the start address of a memory area that stores a format-indicating character string.
<code>p1-p10</code>	Used to specify data.

RETURN CODES

When terminated normally, `rs_printf` returns the character string size (number of bytes) of the converted message.

When terminated abnormally, `rs_printf` returns one of the following returns codes:

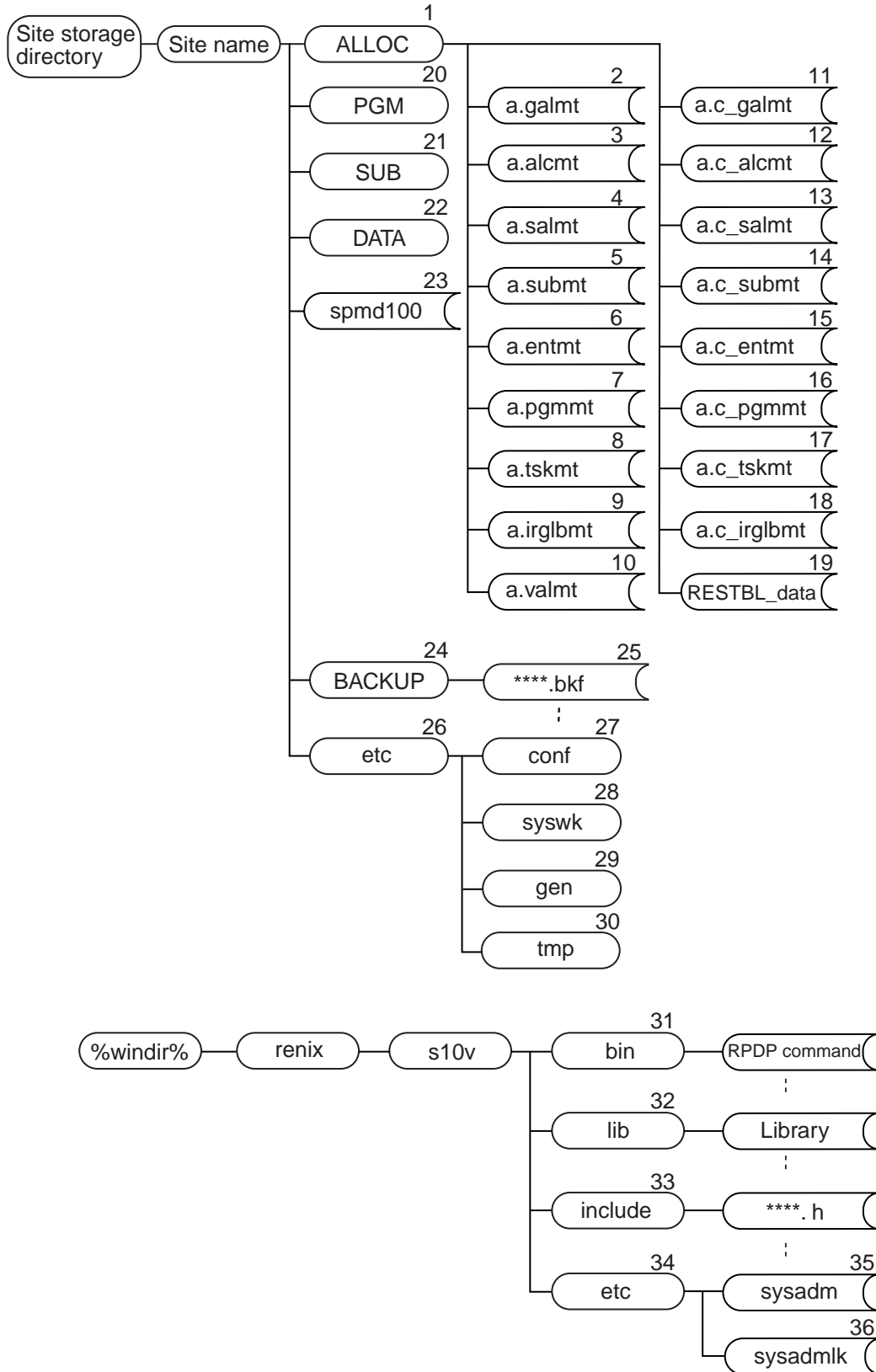
- 0: The message buffer area within the OS was full and could not be written into.
- 1: A write operation could not be performed due to a parameter error.
- 2: A write operation could not be performed due to a message write error.

NOTES

- To output floating-point data, convert it to a character string with the `sprintf` library (C/C++) and then perform a message write with this routine.
- Use this routine for debugging purposes only.

APPENDIX C SITE MANAGEMENT FILES

This appendix shows the configuration of the directory containing files used for site management, and also explains each file.



(1/3)

No.	File/directory name	Descriptive name	Description	Remarks
1	ALLOC	Allocator management table storage directory	This directory stores allocator management tables.	
2	a.galmt	garea management file	Manages the names, sizes, and other attributes of areas within a logical space.	
3	a.alcmt	area management file	Manages task text and data, subprogram text and data, and AREAs (split areas) allocated within a global area.	Includes entries the number of which is defined by MAXAREA in the system.u file.
4	a.salmt	sarea management file	Manages the SAREAs (secondary partition areas) allocated within an AREA (split area) in a global area.	Includes entries to which the number of entries defined by MAXSAREA in the system.u file are added.
5	a.submt	Subprogram management file	Manages subprograms (IRSUBs and built-in subroutines).	Includes entries the number of which is equivalent to the sum of the maximum number of built-in subroutines and the number defined by ENTMT_MAXENT in the system.u file.
6	a.entmt	IRSUB management file	Manages indirect link subprograms (IRSUBs).	Includes entries the number of which is defined by ENTMT_MAXENT in the system.u file.
7	a.pgmmt	Program management file	Manages programs registered as tasks.	Includes entries the number of which is defined by PGM_MAXNUM in the system.u file.
8	a.tskmt	Task management file	Manages tasks	
9	a.irglbmt	Indirect global management file	Manages the registration of indirect link global data.	Includes entries the number of which is defined by IRG_MAXENT in the system.u file.
10	a.valmt	Value management file	Manages the registration of values.	Includes entries to which the number of entries defined by VAL_MAXNUM in the system.u file are added.
11	a.c_galmt	PCs garea management file	Manages the names, sizes, and other attributes of areas within a logical space on the PCs side.	Downloaded into the PCs with the svrpl command or the svdebug command's ld subcommand.
12	a.c_alcmt	PCs area management file	Manages task text and data, subprogram text and data, and AREAs (split areas) allocated within a global area.	
13	a.c_salmt	PCs sarea management file	Manages the SAREAs (secondary partition areas) allocated within an AREA (split area) in a global area on the PCs side.	

C. SITE MANAGEMENT FILES

(2/3)

No.	File/directory name	Descriptive name	Description	Remarks
14	a.c_submt	PCs subprogram management file	Manages the subprograms (IRSUBs and built-in subroutines) on the PCs side.	Downloaded into the PCs with the svrpl command or the svdebug command's ld subcommand.
15	a.c_entmt	PCs IRSUB management file	Manages indirect link subprograms (IRSUBs) on the PCs.	
16	a.c_pgmmt	PCs program management file	Manages programs registered as PCs tasks.	
17	a.c_tskmt	PCs task management file	Manages PCs tasks.	
18	a.c_irglbmt	PCs indirect link global management file	Manages indirect link global registration on the PCs side.	
19	RESTBL_data	Resource management table data file	This table data file manages a site's resources.	
20	PGM	Program storage directory	This directory stores program load modules.	Stored when svload is executed with -d option specified.
21	SUB	Subprogram storage directory	This directory stores subprogram load modules.	
22	DATA	Global initial value data storage directory	This directory stores a global area's initial value data.	
23	spmd100	SPM file	This is a data file for the OS and drivers in the PCs main memory (SPM area).	This file is provided for future use and not targeted for loading at the time of svrpl execution.
24	BACKUP	Backup file storage directory	This directory stores backup files.	Backup files created by svdfa are stored.
25	****.bkf	Backup file	PCs memory initial value file for each split area.	Loaded into the PCs main memory upon svrpl command execution.
26	etc	System generation file storage directory	This directory stores a system generation file.	
27	conf	Site definition information file storage directory for user	This directory contains a file that is user-defined at the time of system generation.	Only the adapter.u file and memory.u file can be user-defined. Do not edit the contents of the other files.
28	syswk	Site definition information file storage directory for system	This directory contains files that the system references at the time of system generation.	Do not edit the contents of the files in this directory.
29	gen	Configuration definition file storage directory	This directory stores a configuration definition information file that is output by the svconf command.	

C. SITE MANAGEMENT FILES

(3/3)

No.	File/directory name	Descriptive name	Description	Remarks
30	tmp	Configuration definition information output file storage directory	This directory stores a definition information output file that is created in accordance with the information under conf.	
31	bin	RPDP command storage directory	This directory stores RPDP commands.	
32	lib	PCs library storage directory	This directory stores PCs libraries.	
33	include	Include file storage directory for PCs	This directory stores PCs include files.	
34	etc	RPDP command management file	This directory stores RPDP command management files.	
35	sysadm	Site management file	This file provides site management for RPDP commands.	
36	sysadmlk	Site lock file	This file provides a site lock for RPDP commands.	

APPENDIX D ERROR MESSAGES

(1) Commands concerning system generation

<svgen>

(1/2)

No.	Error message	Explanation	User's action
1	Can't Find %s. (EC0102)	The specified file was not found.	Review the specified file.
2	Can't Find SUBSYS_DIR. (EC0103)	The SUBSYS_DIR syntax was not found.	
3	Can't Find SITE_NAME. (EC0102)	The SITE_NAME syntax was not found.	
4	Invalid Site Directory Name. (%s) (EC0102)	The site directory name was invalid.	
5	Invalid Machine Type. (%s) (EC0103)	The machine type was invalid.	
6	Can't Open %s. (EC0105)	The file could not be opened.	Confirm the system status.
7	Can't Write %s. (EC0106)	Data could not be written into the file.	
8	Can't Read %s. (EC0106)	Data could not be read from the file.	
9	Can't Link Site Directory. (EC0107)	The site could not be linked.	Review the specified file.
10	Invalid data. (EC0108)	Invalid data was detected.	
11	Invalid Slot No. (%s) (EC0109)	The slot number was invalid.	
12	Duplicate Slot No. (%s) (EC0110)	A duplicate slot number was found.	
13	Can't make directory. (%s) (EC0111)	The directory could not be generated.	Confirm the system status.
14	Can't make Sub-directory. (EC0112)	The subdirectory could not be generated.	
15	Invalid Name. (%s) (EC0113)	An invalid name was encountered.	Review the specified file.
16	Invalid File. (%s) (EC0115)	The file was invalid.	Review the specified file (*1).
17	Generation Data Illegal. (%s) (EC0116)	The data for generation was illegal.	Check the data for generation.
18	Can't Creat Lock File. (%s) (EC0117)	The lock file could not be generated.	Confirm the system status.
19	sysadm Busy. (EC0118)	There was another user.	Try again.
20	Not Super User. (EC0119)	You were not a superuser.	Try again as a superuser
21	Over Max Site Count. (EC0120)	The maximum number of sites was exceeded.	Adjust the number of sites (*2).
22	Can't Copy Define File. (EC0121)	The definition file was not successfully copied.	Confirm the system status.

(2/2)

No.	Error message	Explanation	User's action
23	RPDP Initialize Not Finish. (EC0122)	RPDP was not successfully initialized.	Restart and then try again.
24	Invalid CPU Type. (%s). (EC0124)	An invalid CPU type was specified.	Review the specified file.
25	Usage:svgen fname	—	—

(*1) When the tmpsysadm file exists in the same directory in situations where the sysadm file is found invalid, check for a site generated in the tmpsysadm file. When such a site exists, rename the tmpsysadm file to sysadm. If, on the other hand, no site name is found in the tmpsysadm file and the sysadm file exists, delete the tmpsysadm file.

(*2) By default, the maximum number of sites is 512.

D. ERROR MESSAGES

<svconf>

(1/4)

No.	Error message	Explanation	User's action
1	Can't open file. (%s) (EC0203)	The file could not be opened.	Confirm the system status. Confirm the site name. Confirm the generation information.
2	Invalid Site. (%s) (EC0205)	An invalid site was detected.	
3	Invalid Slot No. (%d) (EC0207)	The slot number was invalid.	
4	Over MAX ARP Entry. (EC0208)	The entry limit was exceeded by the ARP information.	
5	Invalid IP Address. (%s) (EC0209)	The IP address was invalid.	
6	Invalid MAC Address. (%s) (EC0210)	The MAC address was invalid.	
7	Over MAX Route Entry. (EC0211)	The entry limit was exceeded by the route information.	
8	Invalid Gate Way Address. (%s) (EC0212)	The gateway address was invalid.	
9	Over MAX Adapter Entry. (EC0213)	The entry limit was exceeded by the adapter information.	
10	Invalid Station No. (%d) (EC0214)	The station number was invalid.	
11	Invalid Interface Count. (%d) (EC0215)	The number of interfaces was invalid.	
12	Invalid UNO. (%s) (EC0216)	The UNO was invalid.	
13	Invalid Interface No. (%d) (EC0217)	An invalid interface number was found.	
14	Invalid Subnet Mask. (%s) (EC0218)	The subnet mask was invalid.	
15	Invalid BroadCast Address. (%s) (EC0219)	The broadcast address was invalid.	
16	Over MAX Adapter Entry. (System) (EC0220)	The adapter entry limit was exceeded.	
17	Invalid Trace Mode. (%d) (EC0221)	An invalid trace mode was used.	
18	Invalid TCP Port No. (%d) (EC0223)	The TCP port number was invalid.	
19	Invalid UDP Port No. (%d) (EC0224)	The UDP port number was invalid.	
20	Invalid IP Port No. (%d) (EC0225)	The IP port number was invalid.	
21	Invalid ARP Flag. (%d) (EC0226)	The ARP flag was invalid.	
22	Invalid ARP Send Count. (%d) (EC0227)	The number of ARP transfers was invalid.	
23	Invalid TCP Window Size. (%d) (EC0228)	The TCP window size was invalid.	
24	Invalid TCP TIMEOUT. (%d) (EC0230)	The TCP timeout time was invalid.	

D. ERROR MESSAGES

25	Invalid UDP Check SUM Flag. (%d) (EC0231)	The UDP SUM flag was invalid.	
----	--	----------------------------------	--

D. ERROR MESSAGES

(2/4)

No.	Error message	Explanation	User's action
26	Invalid HOP No. (%d) (EC0232)	The HOP number was invalid.	Confirm the generation information.
27	Invalid Remote CPU Control Port No. (%d) (EC0233)	The remote CPU's port number was invalid.	
28	Invalid Cyclic Port No. (%d) (EC0234)	The cyclic port number was invalid.	
29	Invalid AI Report Time. (%d) (EC0235)	The AI report time was invalid.	
30	Invalid AI Report Count. (%d) (EC0236)	The number of AI reports was invalid.	
31	Invalid file. (%s) (EC0237)	An invalid file was detected.	
32	Invalid Slot No. (%s) (EC0238)	The slot number was invalid.	
33	Duplicate Slot No. (%s) (EC0239)	A duplicate slot number was found.	
34	Invalid Adapter Name. (%s) (EC0240)	The adapter name was invalid.	
35	Invalid Main Memory Size. (%s) (EC0241)	The main memory size was invalid.	
36	Invalid Common Memory Size. (%s) (EC0242)	The shared memory size was invalid.	
37	Invalid Duplex Common Memory Size. (%s) (EC0243)	The duplex shared memory size was invalid.	
38	Can't Copy Data File. (%s) (EC0244)	The data file could not be copied.	
39	Can't Rename Data File. (%s) (EC0245)	The data file could not be renamed.	
40	RPDP Initialize Not Finish. (EC0246)	RPDP initialization did not finish.	
41	Invalid UCB No. (%s) (EC0256)	The UCB number was invalid.	
42	Invalid Channel No. (%s) (EC0257)	The channel number was invalid.	
43	Invalid Timeout. (%s) (EC0258)	The timeout time was invalid.	
44	Over MAX Host Entry. (EC0259)	The maximum number of hosts was exceeded.	
45	Invalid Host Name. (%s) (EC0260)	The host name was invalid.	
46	Over MAX Network Entry. (EC0261)	The maximum number of networks was exceeded.	
47	Invalid Network Name. (%s) (EC0262)	The network name was invalid.	
48	Over MAX Service Entry. (EC0263)	The maximum number of services was exceeded.	
49	Invalid Service Name. (%s) (EC0264)	The service name was invalid.	
50	Invalid Port No. (%s) (EC0265)	The port name was invalid.	
51	Invalid Protocol No. (%s) (EC0266)	The protocol number was invalid.	

(3/4)

No.	Error message	Explanation	User's action
52	Over MAX IP_route Entry. (EC0267)	The maximum number of IP routes was exceeded.	Confirm the generation information.
53	Invalid UNO. (%d) (EC0216)	The UNO was invalid.	
54	Can't Write file. (%s) (EC0269)	The file was not successfully written into.	
55	Can't Find UNO in Network Information. (%d) (EC0270)	The UNO was not found in the network information.	
56	Can't Get Memory Size. (EC0271)	The memory size could not be acquired.	
57	Over MAX Dual Network Entry. (EC0275)	The maximum number of duplicate networks was exceeded.	
58	Invalid PATROL. (%d) (EC0278)	The specified PATROL was invalid.	
59	Invalid SOFT_TO. (%d) (EC0279)	The SOFT_TO time was invalid.	
60	Invalid RT_BUFSIZE. (%d) (EC0280)	The RT_BUFSIZE was invalid.	
61	Over MAX UCB Entry. (EC0273)	The maximum UCB count was exceeded.	
62	Over MAX Interface Entry. (EC0276)	The maximum number of interfaces was exceeded.	
63	Invalid HI_WATER. (%d) (EC0281)	The HI_WATER value was invalid.	
64	Invalid LO_WATER. (%d) (EC0282)	The LO_WATER value was invalid.	
65	Invalid CYC_SEND. (%s) (EC0283)	The CYC_SEND value was invalid.	
66	Invalid CYC_WATCH. (%s) (EC0284)	The CYC_WATCH value was invalid.	
67	Invalid CYC_WATCH2. (%s) (EC0285)	The CYC_WATCH2 value was invalid.	
68	Invalid RT_NETADDR. (%s) (EC0286)	The RT_NETADDR was invalid.	
69	Invalid NET_SRBCNT. (%d) (EC0287)	The NET_SRBCNT was invalid.	
70	Not Network Adapter. (Slot:%d) (EC0290)	The slot did not have a network adapter.	
71	Not Network Adapter. (UNO:%d) (EC0291)	The UCB was not a network adapter.	
72	Can't Find UNO in ucb.u. (UNO:%d) (%s) (EC0292)	The UNO number was not found in ucb.u.	
73	Generation Data Illegal. (%s) (EC0295)	The overall information was illegal.	

D. ERROR MESSAGES

(4/4)

No.	Error message	Explanation	User's action
74	Invalid MBUF_CNT. (%d) (EC0296)	The MBUF_CNT value was invalid.	Confirm the generation information.
75	Can't Creat Lock File. (%s) (EC0297)	The lock file could not be generated.	Confirm the system status.
76	Site Busy. (EC0298)	The site was busy.	Try again.
77	Not Super User. (EC0299)	You were not a superuser.	Try again as a superuser.
78	Cannot omit parameter(%s). (EC0701)	Every parameter needs to be specified.	Confirm the generation information.
79	Invalid parameter specified(%s). (EC0702)	An improperly specified parameter was found.	
80	Cannot create file(rms_data.c). (EC0703)	The file could not be generated.	
81	Systemcall error(%s). (EC0704)	A system call error occurred.	
82	Parameter unmatch between definition file and server configuration(%s). (EC0705)	A parameter mismatch was found between the definition file and server information.	
83	GAREA size must multiple of 4 (%s).	The GAREA size needs to be a multiple of 4.	
84	GAREA size too large (%s).	The GAREA size was too large.	
85	Total GAREA size too large.	The total GAREA size was too large.	
86	Old map range over on new map(%s). (EC0320)	The old map usage range was exceeded by the new map.	
87	Can not change oswork top in TCB(%s). (EC0321)	The TCB OS work could not be changed.	
88	Illegal site-information found(%s). (EC0323)	The site information was invalid.	
89	LANCP count is over. (EC0325)	The LANCP setting information was excessive.	
90	Except CMU adapter not supported(%s). (EC0326)	Adapters other than the CMU adapter are not supported.	
91	Usage:svconf site	—	—

<svshconf>

No.	Error message	Explanation	User's action
1	Invalid Site Directory Name. (%s)	The site directory name was invalid.	Confirm the name of the directory containing the site.
2	Can't Open %s. (EC0405)	The file could not be opened.	Confirm the file (*).
3	Can't Read %s. (EC0406)	The file could not be read.	Confirm the file.
4	Illegal sitename. (%s)	The site name was invalid.	Review the site name.
5	Can't Find specified sitename. (%s)	The specified site name was not found.	Confirm the site name.
6	Usage:svshconf sitename [-f ofile]	—	—

(*) Check whether the svconf command was executed for the memory.e file.

D. ERROR MESSAGES

<svsitecp>

(1/2)

No.	Error message	Explanation	User's action
1	Usage:svsitecp site1 site2	An improper option was used.	Start with a proper option.
2	svsitecp:Can't Define Copy Site Name. (site name) (EC0502)	The specified copy source site name was not defined.	Review the site name.
3	svsitecp:Invalid Copy Site Number. (EC0504)	The number of specified copy destination sites did not coincide with that of copy source sites.	Specify the same number of copy destination sites as that of copy source sites, and then restart.
4	svsitecp:Can't Updata sysadm. (EC0505)	The PCs system management file could not be updated.	This error occurs if no site is constructed or the sysadm file is deleted. If a site is constructed, contact the system administrator (*).
5	svsitecp:Invalid Copy Site Name. (site name) (EC0506)	The specified copy destination site name was incorrect.	Review the site name.
6	svsitecp:Can't Copy. (EC0508)	The copy process could not be performed.	It is conceivable that the disk containing the copy destination site directory may not have a sufficient free space. Increase the free space and then restart.
7	svsitecp:Can't Site Lock. (EC0509)	The target site could not be locked.	An RPDP command was being executed for the target site. Terminate the RPDP command and then restart.
8	svsitecp:File Access Error. (file name) (EC0511)	The file could not be accessed.	It is conceivable that the disk may not have a sufficient free space. Increase the free space and then restart.
9	svsitecp:Can't Open file name. (EC0510)	The file could not be opened.	Confirm the security and other conditions of files and directories.
10	svsitecp:Can't Create Lock File. (file name) (EC0512)	The lock file could not be created.	
11	svsitecp:sysadm Busy. (EC0513)	The sysadm file was being used by another generation command.	Restart after termination of the other generation command.
12	svsitecp:Not Super User. (EC0514)	You were a non-administrator.	Restart as an administrator.

(2/2)

No.	Error message	Explanation	User's action
13	svsitecp:Over Max Site Count. (EC0515)	The maximum number of sites that can be constructed on the PC was exceeded.	Increase the maximum site count, which is registered in the %windir%\renix\usr\s10\etc\max_sites file, or delete unnecessary sites, and then restart.
14	svsitecp:Invalid File. (file name) (EC0516)	The file was invalid.	Contact the system administrator (*).
15	svsitecp:Specified Name is Already Used (site name). (EC0521)	The specified site name was already used.	Review the site name.
16	svsitecp:Can't Read System Management File. (EC0522)	The PC's system management file could not be read.	This error occurs if no site is constructed or the sysadm file is deleted. If a site is constructed, contact the system administrator (*).

(*) For errors requiring you to "Contact the system administrator" as a remedial action, collect the following data for investigation:

- %windir%\renix\etc\log\s10v\1_RPDP_CTL, Rpdpcntl

D. ERROR MESSAGES

<svsitedel>

No.	Error message	Explanation	User's action
1	Usage:svsitedel site	An improper option was used.	Start with a proper option.
2	svsitedel:Can't Define Delete Directory Name. (site name) (EC0602)	The specified site name was not defined in sysadm.	Review the site name.
3	svsitedel:Can't Write file name. (EC0605)	The file could not be written into.	It is conceivable that the disk may not have a sufficient free space. Increase the free space and then restart.
4	svsitedel:Can't Delete Data in sysadm. (EC0608)	The data in the sysadm file could not be deleted.	
5	svsitedel:Invalid File file name. (EC0606)	The file structure was invalid.	Correct the file structure and then restart.
6	svsitedel:Not Super User. (EC0607)	You were a non-administrator.	Restart as an administrator.
7	svsitedel:Can't Open file name. (EC0603)	The file could not be opened.	Confirm the security and other conditions of files and directories.
8	svsitedel:Can't Creat Lock File. (file name) (EC0611)	The lock file could not be created.	
9	svsitedel:sysadm Busy. (EC0612)	The sysadm file was being used by another generation command.	Restart after termination of the other generation command.
10	svsitedel:Can't Delete Directory. (directory name) (EC0613)	The directory was not successfully deleted.	Terminate access to the directory and then restart (*).
11	svsitedel:Can't Site Lock. (EC0615)	The target site could not be locked.	An RPDP command was being executed for the target site. Terminate the RPDP command and then restart.

(*) Stop referencing with Windows Explorer or the like and then reexecute the svsitedel command with the -d option specified.

(2) Error messages displayed by the allocator, loader, and builder commands

- Operation in reception of a signal

When a signal is received during the execution of an allocator, loader, builder, or map display command, the command operates as shown in Table D-1.

Table D-1 Operation in Reception of a Signal

Signal	Operation in reception of a signal
SIGINT	The command suspends the processing.
SIGQUIT	

- Error messages

Table D-2 lists the error messages displayed by the allocator, loader, builder, and map display commands. When detecting an error, these commands display the pertinent error message and terminate immediately.

Table D-2 Error Messages (1/8)

Error code	Message	Explanation	User's action
Internal errors requiring recovery by the system administrator			
1001-1	Abnormal allocator management table (%s)	An allocator management table was in error.	Contact the system administrator. (*)
Errors involved in insufficient resources			
1002-4	Not enough physical memory allocated	Insufficient physical memory	Perform system generation again for the insufficient resource.
5	Default gareaa is not defined	The default GAREA was not defined.	
6	Not enough area allocated (%s)	Insufficient GLB area	
7	Not enough secondary storage area allocated (%s)	Insufficient auxiliary memory	
8	No task number available	There were no more available task numbers.	
10	No free table to make new entry (%s)	There were no more available allocator management tables.	
11	Too small system constants to resize management table (%s)	The system constant was too small.	
13	Cannot get RSSITE (%s)	The RSSITE environment variable could not be fetched.	
15	Please set environment variable(%s)	The environment variable was not set.	
Errors involved in insufficient system resources			
1003-1	Memory allocation error (malloc, %s)	Memory could not be allocated with malloc.	Contact the system administrator. (*)
3	Cannot create temporaries: %s	A temporary file could not be created.	
Errors unrecoverable by retry			
1004-1	Allocator management table is busy	The allocator management table was being used by another command.	Reexecute the command.

D. ERROR MESSAGES

Table D-2 Error Messages (2/8)

Error code	Message	Explanation	User's action
System call errors due to identifiable causes			
1005-1	Cannot open %s (%s)	The file could not be opened.	Contact the system administrator. (*)
2	Cannot read %s (%s)	The file could not be read from.	
3	Cannot write % (%s)	The file could not be written to.	
4	Can't exec %s	An internal command could not be executed.	
5	Lost %s - No child process!	A child process was deleted.	
6	Macro error (dtaskhr, %d)	dtaskhr could not be executed.	
7	Not dormant status	The task was not in the DORMANT state. (There was an internal inconsistency.)	
8	Cannot unlink %s (Text file busy)	A program could not be deleted.	
9	Cannot stat %s (%s)	Status information could not be read from the file.	
10	Cannot lseek %s (%s)	A file pointer could not sought.	
System call errors due to unidentifiable causes			
1006-1	systemcall error (%s, errno=%d)	System call error	Contact the system administrator. (*)
2	Command name: WIN32API error (API name, EC = error code)	A function having an API name caused an error indicated by the error code.	
Parameter insufficiency errors			
1007-3	Not enough parameter	Arguments were not sufficient.	Review the arguments.

Table D-2 Error Messages (3/8)

Error code	Message	Explanation	User's action
Out-of-range input parameters			
2001-2	Align number is out of range	The alignment number was invalid.	Check the data that can be entered, then reexecute the command.
3	Task number is out of range (1 to %d)	The task number was invalid.	
5	Priority level is out of range (%d to %d)	The user task execution level was invalid.	
6	Priority level for system is out of range (%d to %d)	The system task execution level was invalid.	
9	Bad align type	The alignment type was invalid.	
10	Illegal point number (%d)	The point number for a built-in subroutine was invalid.	
11	Entry number is out of range (1 to %d)	The entry number for a built-in subroutine was invalid.	
12	Specified index number is out of range (1 to %d)	The specified index number is was invalid.	
13	Invalid name (%s)	The specified name was in error.	
18	Illegal point name (%s)	The point name specified for a built-in subroutine was in error.	
19	Numeric value is out of range	The specified numeric value was invalid.	
23	Limit size for stack is out of range (0 to 2097152)	The specified stack size was invalid.	
24	User task number is out of range (1 to 224)	The specified user task number was invalid.	
25	Specified system index number is out of range (1 to %d)	The specified system index number was invalid.	
26	Specified number with -r is out of range	The number specified by the -r option was invalid.	
27	Loading data is empty	There was no loading data.	
28	Number of user task is over (number_of_tasks)	The number of user tasks registered was greater than the value of the system constant.	Check the value of the system constant MAX_USERTN, then execute svgen or svconf.
33	ULSUB stack size (%d) is out of range (0 to 512)	The stack size of a built-in subprogram was larger than 512 bytes.	
35	Program/IRSUB stack size (%d) is out of range (0 to 8388608)	The stack size of a task was larger than 8MB.	Review the stack size.

D. ERROR MESSAGES

Table D-2 Error Messages (4/8)

Error code	Message	Explanation	User's action
Undefined input parameters			
2002-1	Specified name is undefined (%s)	An undefined name was given.	Check the data that can be entered, then reexecute the command.
4	Specified point number in the entry number is empty	An undefined point number was given for a built-in subroutine.	
5	Specified IRSUB is not built (%s)	An IRSUB not yet built was specified.	
7	Specified IRSUB is already built (%s)	An already-built IRSUB was specified.	
8	Specified number is undefined	An undefined number was given.	
10	%s is undefined	An undefined name was detected in an object.	
11	Loading data is empty	There was no data to be loaded.	
13	Area (%s) kind is wrong	The area type was wrong.	Review the specified area.
14	Can not load data in GLBW (%s)	No initial value can be loaded into a GLB area without an initial value.	
Multiply defined input parameters			
2003-1	Specified name is already defined (%s)	An already-registered name was given.	Check the data that can be entered, then reexecute the command.
5	Task number is already defined	An already-registered task number was given.	
7	Point number is already defined	An already-registered point number was given for a built-in subroutine.	
8	Specified IRSUB number is already defined	An already-registered IRSUB number was given.	Change the IRSUB number, then reexecute the command.
13	Unmatched reserved index number	The specified index number did not match.	Check the data that can be entered, then reexecute the command.
15	PN=%s is already defined	A duplicate program management number was given.	
19	Specified number is already defined	An already-registered number was given.	
21	Can not specify -s or -a with SAREA (%s)	The -s or -a option given for a secondary partition area was illegal.	
22	Specified pgmname is already defined as TASK (%s)	An already-registered task name was given as a program name.	
23	PN=%d is already loaded for single task	An already-registered program management number was given.	Change the program management number, then reexecute the command.

Table D-2 Error Messages (5/8)

Error code	Message	Explanation	User's action
Unmatching input parameter attributes			
2004-1	Unmatched owner type	Unmatched owner type	Check the data that can be entered, then reexecute the command.
2	Illegal user type (%s)	Unmatched user type	
3	Specified area is not global (%s)	The area type was not GLB.	
5	Illegal program type	Unmatched program type	
8	Unmatched entry number	An unmatched entry number was given for a built-in subroutine.	
10	Area type is not GLBI	The area type was not GLB having initial data.	
11	Multi task attribute error	Unmatched multitask attribute	
16	Unmatched entry type	An unmatched entry set number was given for a built-in subroutine.	
17	Specified name is defined as GLB (%s)	An already-defined GLB name was given.	
19	Specified name is defined as VAL (%s)	An already-defined VAL name was given.	
21	4096 aline error (%s)	The specified address was not on a 4096-byte boundary.	
22	Physical address error (%s)	An invalid logical address was given.	
23	Area (%s) kind is wrong	An invalid area attribute was given.	
24	Loading data is too large (sname=%s)	The specified data size was greater than the area size.	
25	Inconsistent object was mixed	Both -lsh4nbmzz and -lsh4nbmdn were specified for library selection purposes.	
Incorrect operations			
2005-1	Cannot delete area which is already used	The area could not be deleted because tasks or subprograms were registered.	Execute svdload before deleting the area.
2	Cannot delete program which is registered as task	The program could not be deleted because it was registered as a task.	Execute svdtask before deleting the program.
3	Cannot delete built subprogram (%s)	The subprogram could not be deleted because it was an already built subprogram.	Execute svdbuild before deleting the already built subprogram.
6	Cannot delete defined %s (%s)	The area could not be deleted because it was registered as GLB or VAL.	Execute svdls or svdlv before deleting the area.
7	Specified name (%s) is referenced by PROG or SUB	The specified resource could not be deleted because it was referenced by a program or subprogram.	Delete the referencing program or subprogram before deleting the specified resource.

D. ERROR MESSAGES

Table D-2 Error Messages (6/8)

Error code	Message	Explanation	User's action
Invalid parameters detected in the svload command			
2006-1	Invalid subargument: -W%s	An unusable sub-argument was given.	Check the data that can be entered, then reexecute the command.
5	Too few arguments	The number of specified arguments was insufficient.	
8	Missing operand (%s)	Some operands were missing.	
9	Bad option (%s)	An unusable option was given.	
10	Invalid name (%s)	A name was specified incorrectly.	
Invalid parameters detected in the svload command			
2008-1	Error in %s; Status %d	An internal command (%s) caused an error.	Check whether the specified library is correct, review the object file and its source, then reexecute the command.
2	Fatal error in %s; Status %d	An internal command (%S) caused a fatal error.	Contact the system administrator. (*)
Allocator management table errors			
200-1	RMphase, (0x%02x)	The processing phase was invalid.	Contact the system administrator. (*)
Invalid input data			
-	Argument list too long	Too many arguments were given.	Check the data that can be entered, then reexecute the command.
-	Argument data too large	Too much data was given in an argument.	
-	File open error	The specified file could not be opened.	
-	Illegal format in file_name, line	line in the specified file was in an invalid format.	
-	Illegal format of name	A name was specified in an invalid format.	

Table D-2 Error Messages (7/8)

Error code	Message	Explanation	User's action
Invalid input data (continued from the previous page)			
—	Illegal format of numeric value	Numerical data was specified incorrectly.	Check the data that can be entered, then reexecute the command.
—	Illegal format of task name	A task name was specified in an invalid format.	
—	Illegal operand	An operand was specified incorrectly.	
—	Program text is empty	The specified program text size was 0.	
—	%s is different from subprogram top name	A subprogram name was specified incorrectly.	
—	%s is referred from system type	A system task was referencing the user task.	
—	%s is referred from user type	A user task was referencing the system task.	
—	Illegal option	An invalid option was given.	
—	Illegal option combination	Options were combined incorrectly.	
—	Missing option parameter	An option parameter was specified incorrectly.	
—	Numeric value is out of range	The specified numeric value was invalid.	
—	Parameter error	A parameter was specified incorrectly.	
—	Specified number is undefined (specified_name)	The name and -n options were given in the svmap command, but the entry number specified by name was not defined.	
—	Specified number is illegal (specified_name)	The name and -n options were given in the svmap command, but the entry number specified by name was invalid.	
—	Specified name is undefined (specified_name)	The name option was given in the svmap command, but name was undefined.	
—	Task number error	A task number was specified incorrectly.	
—	Bad file name	The file name, specified as the operation result output destination in the svadm command, was longer than 255 characters.	
—	Bad site name	The site name given in the svadm command was longer than 14 characters.	
Execution environment errors			
—	Please set RSSITE	The RSSITE environment variable was not set.	Set the RSSITE environment variable, then reexecute the command.
—	Unknown RSUTYP	An invalid parameter was set in the RSUTYP environment variable.	Set an “s” or “u” in the RSUTYP environment variable.

D. ERROR MESSAGES

Table D-2 Error Messages (8/8)

Error code	Message	Explanation	User's action
Command errors (continued from previous page)			
—	abnormal error (error=error_number)	An internal error was detected.	Contact the system administrator. (*)
—	cannot perform malloc	A work area could not be allocated by malloc or realloc.	
—	Internal error (timeout detected)	A communication time-out was detected.	
—	Internal error (no valid data)	An error was found in communication data.	
—	Internal error (select failed)	A select system call error was detected.	
—	select error (errno=error_number)	A select system call error was detected.	
—	cannot get filesize	The svmap command could not fetch the size of the file that manages the free areas in the backup file.	
—	cannot open %s	The svmap command could not open the allocator management table file.	
—	bad user name %s	A user name was not registered in the password file entry.	
—	Specified site is undefined	The specified site could not be found.	

(*) For errors requiring you to “Contact the system administrator” as a remedial action, collect the following data for investigation:

- %windir%\renix\etc\log\s10v\1_RPDP_CTL,Rpdctl

(3) Error messages displayed by the svdebug command

(1/6)

No.	Error message	Explanation	User's action
1	File "file_name" already exists	The specified file was already existent.	Specify the correct file name.
2	Site "site_name" not found	The specified site could not be found.	Check the site name.
3	Cannot open "file_name"	The specified file could not be opened.	Specify the correct file name.
4	No filename given for -i/-o/-r option	A file name was missing.	Specify a file name.
5	No sitename given for -u option	A site name was missing.	Specify a site name.
6	Task no error	A task number was specified incorrectly.	Check the task number.
7	Task name error	A task name was specified incorrectly.	Check the task name.
8	Factor error	A start factor was specified incorrectly.	Check the start factor.
9	Cannot Specify RPC-server task	Giving an RPC server task was illegal.	Check the task number and task name.
10	Unknown sub command	An uninterpretable subcommand name was given.	Check the subcommand name.
11	Name error	An uninterpretable name was given.	Check the name.
12	Option error	An uninterpretable option was given.	Check the option.
13	Storage error	An uninterpretable storage device was specified.	Check the storage device.
14	Invalid address set	An inaccessible address was given.	Check the address.
15	Misformed patch data	New data used for modification was set incorrectly.	Enter a real number in octal, decimal, or hexadecimal.
16	Uknown RSSITE	The RSSITE environment variable was not yet set.	Set the RSSITE environment variable.
17	Unknown RSUTYP	An attempt was made to access a system resource in user mode.	Set an "s" or "u" in the RSUTYP environment variable.
18	Break point already used by another process	The break point was being used by another debugger process.	Use after the other user finishes.
19	Input error	Incorrect input format.	Check the input format.
20	Type or length error	The md or sd option was specified incorrectly.	Check the option.
21	RPDP library error (library_name: error_code)	An RPDP library caused an error.	Contact the system administrator.
22	Allocator management table busy	The allocator management table was busy.	Reexecute the command.

D. ERROR MESSAGES

(2/6)

No.	Error message	Explanation	User's action
23	Unmatch resource status	There was a disagreement in the resource state between the development machine and controller.	Check the specified name.
24	Id no error	id was specified incorrectly in the tm subcommand.	Check id.
25	Time error	The "t" was specified incorrectly in the tm subcommand.	Check the "t."
26	Cycle time error	The "cyct" was specified incorrectly in the tm subcommand.	Check the "cyct."
27	Initial/check data error	An initialization check pattern was specified incorrectly in the si or sp subcommand.	Check the specified initialization check pattern.
28	Addr error	An address was specified incorrectly in the br, rb, or as subcommand.	Check the specified address.
29	Break point is used	A breakpoint was being used.	Reset the breakpoint and then exit the debugger.
30	Bit data error	Bit data was specified incorrectly in the bs or bg subcommand.	Check the specified bit data.
31	Sht sub command is already executed by another process	The sht subcommand was being executed by another debugger process.	Use after the other user finishes.
32	Task no error (NO.2100-01)	The task number was incorrectly specified.	Review the task number.
33	Task name error (NO.2100-02)	The task name was incorrectly specified.	Review the task name.
34	Factor error (NO.2100-03)	The start factor was incorrectly specified.	Review the start factor.
35	Cannot Specify RPC-server task (NO.2100-04)	No RPC server task can be specified.	Review the task number and task name.
36	Specified task is dormant (NO.2100-05)	The specified task was dormant.	Confirm the task status and then try again.
37	Specified task is not dormant (NO.2100-06)	The specified task was not dormant.	
38	Specified task is already suspend (NO.2100-07)	The specified task was already suspended.	
39	Specified task is not suspend (NO.2100-08)	The specified task was no longer suspended.	
40	Specified task is not registered (NO.2100-09)	The specified task was not registered.	Review the task number and task name.
41	Backup file access error (NO.2100-10)	The backup file was not successfully accessed.	Contact the system administrator (*1).

(3/6)

No.	Error message	Explanation	User's action
42	Unmatched RSUTYP (NO.2100-11)	The system resources cannot be accessed in the user mode.	Gain access in the system mode.
43	Unmatch resource status (NO.2100-12)	A resource status mismatch was found between the development machine and controller.	Review the specified name.
44	Specified task is undefined (NO.2100-13)	The specified task name was not defined.	Review the specified task.
45	User Task not running (NO.2100-14)	The user task was not running.	Ensure that the user task is running.
46	Task is not dormant (tn=%d) (NO.2100-15)	The task was not dormant.	Render the task dormant and then try again.
47	Invalid address error (NO.2100-16)	An attempt was made to access an inaccessible address.	Review the address.
48	Invalid address set (NO.2100-17)	An inaccessible address was specified.	
49	Cannot open "file name" (NO.2100-18)	The file could not be opened.	Specify a correct file name.
50	Processor connection table is full (NO.2100-19)	The inter-process connection table was full.	Wait until another user finishes and then try again.
51	Specified task is not idle (NO.2100-20)	The specified task was not idle.	Confirm the task status and then try again.
52	Timer event is not registered (NO.2100-21)	No timer event was registered.	Review the specified task.
53	Time error (NO.2100-22)	The value t was incorrectly specified for the tm subcommand.	Review the value t.
54	Cycle time error (NO.2100-23)	The value cyct was incorrectly specified for the tm subcommand.	Review the value cyct.
55	Cannot get system constant (NO.2100-24)	The system constant was not successfully acquired.	Contact the system administrator (*1).
56	Ent error (NO.2100-25)	The value ent was incorrectly specified for the ld subcommand.	Review the specified ent value.
57	Irsub No error (NO.2100-26)	The indirect link subroutine number was wrong.	Review the indirect link subroutine number.
58	Irglobal No error (NO.2100-27)	The indirect link global number was incorrectly specified.	Review the indirect link global number.
59	Task suspend failed (NO.2100-28)	The task was not successfully suspended by the ta subcommand.	Confirm the task status and then try again.

D. ERROR MESSAGES

(4/6)

No.	Error message	Explanation	User's action
60	Point error (NO.2100-29)	An incorrect point was specified for the ld subcommand.	Review the specified point.
61	Cannot get register information (NO.2100-30)	The contents of a register were not successfully acquired by the ta/rr subcommand.	Contact the system administrator (*1).
62	Specified name "name" is undefined (NO.2100-31)	The specified name was not defined.	Review the specified name.
63	Cannot register timer event in TRB (NO.2100-32)	Timer event registration failed.	Contact the system administrator (*1).
64	File "file_name" already exists (NO.2100-33)	An existing file was specified.	Review the specified file name.
65	File "file_name" creat error (NO.2100-34)	A file could not be created.	Check the specified file name.
66	Cannot save "file_name" (NO.2100-35)	A file could not be saved.	
67	File "file_name" read error (NO.2100-36)	A file could not be read.	
68	File "file_name" format error (NO.2100-37)	A file was specified in an invalid format in the ld or cm subcommand.	
69	Pname "file_name" not found (NO.2100-38)	The program name was not found.	Review the program name.
70	Must specify address in text space (NO.2100-39)	Specify an address within a text space.	Review the specified address.
71	Specified address is already set (NO.2100-40)	The specified address was already set.	
72	Must specify break point address (NO.2100-41)	Specify a breakpoint address.	
73	Cannot get TCB (NO.2100-42)	The ta subcommand could not fetch a TCB.	Contact the system administrator. (*1)
74	Cannot set break point beyond the max (NO.2100-43)	The maximum selectable number of breakpoints (5 breakpoints) were already specified.	Reset the breakpoints and then try again.
75	Cannot use ld sub command after RSSRCV set	An attempt was made to use the ld subcommand after RSSRCV was set.	Perform batch loading with the rplhr command.
76	Inconsistency detected %s (NO.2100-46)	An inconsistency was found in the allocator management table.	Contact the system administrator (*1).
77	Specified area is not defined for glb (NO.2100-50)	The specified split area was not a global area.	Review the specified split area name.
78	Specified pgm number is out of range (1 to 255) (NO.2100-52)	The specified program number was out of range.	Review the specified parameters.
79	ADT channel is already set (NO.2100-56)	The ADT was already set.	Reset the ADT and then execute.

(5/6)

No.	Error message	Explanation	User's action
80	Illegal break point address (laddr=logical address) (NO.2100-58)	The logical address set for a breakpoint was not registered as a program address.	Restart the CPU and then reset the breakpoint setting.
81	Specified raddr is not break point address (raddr=relative address) (NO.2100-59)	No breakpoint was set at the specified relative address (raddr).	Review the specified address.
82	Break task is not found (NO.2100-60)	No task was found to be halted at a breakpoint.	Confirm the breakpoint setup.
83	Specified name (program name) is used break point (NO.2100-61)	A breakpoint was set for the program name that was specified by the ld subcommand.	Reset the breakpoint and then try again.
84	Specified area is not initialize data area (NO.2100-62)	The specified address pointed to an area containing no backup file.	Review the specified address.
85	Specified address is not initialize data area (NO.2100-63)	No backup file was found in the specified area.	Review the specified area.
86	Specified name (%s) is not GLB area (NO.2100-66)	The specified name did not represent a GLB area.	Specify a GLB area.
87	Communication error (catch signal) (NO.2101-01)	A signal was received.	Contact the system administrator. (*1)
88	Communication error (connection timeout) (NO.2101-02)	A time-out was generated.	
89	Communication error (connection refused) (NO.2101-03)	No RPC server was existent.	
90	Communication error (connection cut) (NO.2101-04)	The RPC server was disconnected.	
91	Communication error (connection reset) (NO.2101-05)	A connection was reset.	
92	Communication error (server closed) (NO.2101-06)	The RPC server was closed.	
93	Communication error (port busy) (NO.2101-07)	The line port was busy.	Wait until another user terminates communication, then reexecute the command.
94	Communication error (bad socket specified) (NO.2101-08)	The specified socket was invalid.	Contact the system administrator. (*1)
95	Communication error (socket creat error) (NO.2101-08)	A socket could not be created.	
96	Communication error (no buffer) (NO.2101-10)	Memory could not be allocated.	
97	Communication error (network not reached) (NO.2101-11)	The network was not connected.	
98	Communication error (network down) (NO.2101-12)	The interface connected to the network was down.	

D. ERROR MESSAGES

(6/6)

No.	Error message	Explanation	User's action
99	Communication error (port No error) (NO.2101-13)	A port number could not be fetched.	Contact the system administrator. (*1)
100	Communication error (IP address error) (NO.2101-14)	An IP address could not be fetched.	
101	Communication error (memory attach failed) (NO.2101-15)	Shared memory could not be attached.	
102	Communication error (trace file cannot open) (NO.2101-16)	A trace file could not be opened.	
103	Communication error (trace file cannot copy) (NO.2101-17)	A trace file could not be copied.	
104	Communication error (fatal error) (NO.2101-18)	A fatal error was detected.	
105	Communication error (library_name: error_number) (NO.2101-19)	An RPL or RRB library caused an error.	
106	Communication error (inter PU communication timeout) (NO.2101-20)	A time-out occurred during inter-PU communication.	Contact the system administrator. (*1) (*2)
107	Communication error (rc=error_number) (NO.2101-21)	RPC library error	

(*1) For errors requiring you to “Contact the system administrator” as a remedial action, collect the following data for investigation:

- %windir%\renix\etc\log\s10vc\RPDP_CTL
- %windir%\renix\etc\log\s10v\1_RPDP_CTL

(*2) In the event of a communication error, see the following:

<Meanings of communication error codes>

- | | |
|--|---|
| 0x11: The socket was invalid. | 0x04: The frame creation memory was not successfully allocated. |
| 0x12: The IP address was invalid. | 0x05: Data transmission failed. |
| 0x14: The storage area address was invalid (0 specified, dta). | 0x06: An error occurred during a wait for response reception. |
| 0x15: The storage area address was invalid (0 specified, wka). | 0x07: The maximum retry count was exceeded because the response was not received. |
| 0x16: The size was invalid (smaller than 0 KB or larger than 16 KB). | 0x08: Data reception failed. |
| 0x17: The size was invalid (non-longword size). | 0x18: The storage area address was invalid (0 specified, dmaia). |
| 0x03: The remote adapter type was invalid. | 0x19: The storage area address was invalid (0 specified, reta). |

(4) List of svrpl command error messages

No.	Error message	Explanation	User's action
1	No sitename given for -u option	The site name was not specified.	Specify the site name.
2	No unitname given for -U option	The unit name was not specified.	Specify the unit name.
3	unknown RSSITE	The specified site name was not found.	Confirm the specified site.
4	Site=%s not found		
5	Unit=%s not found	The specified unit name was not found.	Confirm the specified unit.
6	%s cannot open	The file could not be opened.	Check whether the file is normal.
7	%s file access error	The file could not be accessed.	
8	Internal error (%s)	An internal error occurred.	Try again.
9	download file (%s) not found	The backup file to be downloaded was not found.	Confirm the environment.
10	site (%s) lock busy	The site was being used by another process.	Try again.
11	site (%s) lock error		
12	communication error (%s, RC=0x%x, 0x%x)	A communication error occurred.	Identify the cause of the error in accordance with the RC code (*).
13	communication error (%s, RC=0x%x)		
14	site (%s) allocator management tables modify error	An error occurred during an allocator management table update.	With the svmkrestbl command, repair the allocator management table.
15	IP ADDRESS SET ERROR (RC=0x%x)	An error occurred during IP address setup.	Identify the cause of the error in accordance with the RC code.
16	%s (slot=%d) NON EXIST	The slot was not found.	Confirm the system generation information.
17	File mapping error (%s)	The RPD resource table could not be set up.	Try again.
18	site (%s) unlock error	An error occurred when an attempt was made to unlock the site.	
19	Can not specified NP site(%s)	The specified site was an NP site.	Specify a CP site.
20	Usage:svrplhr [-u site {-s}] [-r] [-time -notime]	—	—

(*) In the event of a communication error, see the following:

<Meanings of communication error codes>

0x11: The socket was invalid.

0x12: The IP address was invalid.

0x14: The storage area address was invalid (0 specified, dta).

0x15: The storage area address was invalid (0 specified, wka).

0x16: The size was invalid (smaller than 0 KB or larger than 16 KB).

0x17: The size was invalid (non-longword size).

0x04: The frame creation memory was not successfully allocated.

0x05: Data transmission failed.

0x06: An error occurred during a wait for response reception.

0x07: The maximum retry count was exceeded because the response was not received.

0x08: Data reception failed.

0x18: The storage area address was invalid (0 specified, dmaia).

0x19: The storage area address was invalid (0 specified, reta).

D. ERROR MESSAGES

0x03: The remote adapter type was invalid.

(5) List of svcpuctl command error messages

No.	Error message	Explanation	User's action
1	No sitename given for -u option	The unit name was not specified.	Specify the unit name.
2	unknown RSSITE	The specified site name was not found.	Confirm the specified site.
3	Site=%s not found		
4	Internal error (%s)	An internal error occurred.	Try again.
5	site (%s) lock busy	The site was being used by another process.	
6	site (%s) lock error		
7	communication error (%s, RC=0x%x, 0x%x)	A communication error occurred.	Identify the cause of the error in accordance with the RC code.
8	communication error (%s, RC=0x%x)		
9	%s (slot=%d) NON EXIST	The slot was not found.	Confirm the system generation information.
10	site (%s) unlock error	An error occurred when an attempt was made to unlock the site.	Try again.
11	Usage:svcpuctl [{-u site} {-s {-stop -run}}] [-time] Usage:svcpuctl [-u site] -ss	—	—

D. ERROR MESSAGES

(6) Error messages displayed by the svelog command

No.	Error message	Explanation	User's action
1	usage: svelog [-u site] [-f format] [-logno] [+case] [-d fname] [-o fname]	An option was specified incorrectly.	Specify a correct option.
2	Unknown RSSITE	The RSSITE environment variable was not set.	Set the RSSITE environment variable.
3	logno error. logno is 1-999	The specified log number was invalid.	Check the log number.
4	unknown site (sitename)	The specified site could not be found.	Check the site name.
5	communication error (error_code)	Transmission or reception failed between the development machine and controller.	Contact the system administrator. (*)
6	memory allocate error	Memory could not be allocated.	
7	logno "log_number": not found	The error log having the specified log number could not be found.	Check the log number.
8	no error log.	There was no error log.	There was no error.
9	cannot open "file_name"	A file could not be opened.	Contact the system administrator. (*)
10	cannot read "file_name"	An error was detected while a file was being read.	
11	specified logno is not found	The error log having the specified log number could not be found.	Check the log number.
12	eloghr: Invalid file name (XXXX)	File name XXXX was invalid.	Specify the correct file name, then re-execute the command.

(*) For errors requiring you to “Contact the system administrator” as a remedial action, collect the following data for investigation:

- %windir%\renix\etc\log\s10v\RPDP_CTL
- %windir%\renix\etc\log\s10v\1_RPDP_CTL

(7) Error messages displayed by the svdhp command

No.	Error message	Explanation	User's action
1	usage: svdhp [-u site] [+count] [-on -off] [-d fname] [-o fname]	An option was specified incorrectly.	Specify a correct option.
2	Unknown RSSITE	The RSSITE environment variable was not set.	Set the RSSITE environment variable.
3	No such site (site name)	The specified site was not found.	Review the site name.
4	Some system constants are not defined	An undefined system constant was encountered.	Review the system constant definitions.
5	Bad realtime environment	The system environment was improper.	Contact the system administrator. (*)
6	memory allocate error	Memory could not be allocated.	It is conceivable that the error occurred because the available memory was temporarily insufficient. Verify that the available memory is adequate, and then reexecute the command.
7	cannot open "file_name"	A file could not be opened.	Confirm the security and other conditions of files and directories.
8	cannot read "file_name"	An error was detected while a file was being read.	
9	cannot write "file_name"	An error was detected while a file was being written.	
10	No such AREA (DHP_RD) in salmt	The DHP read area (DHP_RD) could not be found in the global area.	Contact the system administrator. (*)
11	Memory access error	The controller memory was not successfully accessed.	
12	Memory allocation error (malloc, dhp read area)	A DHP read area could not be allocated.	It is conceivable that the error occurred because the available memory was temporarily insufficient. Verify that the available memory is adequate, and then reexecute the command.
13	Communication error (catch signal)	A signal was received.	Contact the system administrator. (*)
14	Communication error (connection timeout)	A connection time-out was generated.	
15	Communication error (connection refused)	No RPC server was existent.	
16	Communication error (connection cut)	A connection was discontinued.	
17	Communication error (connection reset)	A connection was reset.	
18	Communication error (server closed)	The RPC server was closed.	

D. ERROR MESSAGES

No.	Error message	Explanation	User's action
19	Communication error (port busy)	The line port was busy.	Wait until another user terminates communication, then reexecute the command.
20	Communication error (socket create error)	A socket could not be created.	Contact the system administrator. (*)
21	Communication error (no buffer)	Memory could not be allocated.	
22	Communication error (network not reached)	The network was not connected.	
23	Communication error (network down)	The interface connected to the network was down.	
24	Communication error (port No error)	A port number could not be fetched.	
25	Communication error (IP address error)	An IP address could not be fetched.	
26	Communication error (memory attach failed)	Shared memory could not be attached.	
27	Communication error (trace file cannot open)	A trace file could not be opened.	
28	Communication error (trace file cannot copy)	A trace file could not be copied.	
29	Communication error (fatal error)	A fatal error was detected.	
30	dhp data read error	An error was detected when an attempt was made to read dhp trace data.	
31	Cannot dhp trace ON/OFF	An error was detected when an attempt was made to exercise dhp trace control.	
32	svdhp: Invalid file name (XXXX)	File name XXXX was invalid.	Specify the correct file name, then re-execute the command.

(*) For errors requiring you to “Contact the system administrator” as a remedial action, collect the following data for investigation:

- %windir%\renix\etc\log\s10v\1_RPDP_CTL, Rpdpc_t1

(8) Error messages displayed by the svcpunow command

No.	Error message	Explanation	User's action
1	Usage:svcpunow [-u site] [-t second]	An option was specified incorrectly.	Specify a correct option.
2	Unknown RSSITE	The RSSITE environment variable was not set.	Set the RSSITE environment variable and then try again.
3	memory allocation error (malloc, puloadinfo area)	An area from which to read PU load ratio information could not be allocated.	It is conceivable that the error occurred because the available memory was temporarily insufficient. Verify that the available memory is adequate, and then reexecute the command.
4	cannot get PU load information	A PU load ratio could not be fetched.	Contact the system administrator. (*)
5	Not available parameter	An unusable parameter was detected.	
6	No such AREA (puloading) in salmt	An area from which to read PU load ratio information could not be searched.	
7	Communication error (catch signal)	A signal was received.	
8	Communication error (connection timeout)	A time-out was generated.	
9	Communication error (connection refused)	No RPC server was existent.	
10	Communication error (connection cut)	The RPC server was disconnected.	
11	Communication error (connection reset)	A connection was reset.	
12	Communication error (server closed)	The RPC server was closed.	
13	Communication error (port busy)	The line port was busy.	
14	Communication error (bad socket specified)	A socket descriptor was specified incorrectly.	
15	Communication error (socket create error)	A socket could not be created.	
16	Communication error (no buffer)	Memory could not be allocated.	It is conceivable that the error occurred because the available memory was temporarily insufficient. Verify that the available memory is adequate, and then reexecute the command.

D. ERROR MESSAGES

No.	Error message	Explanation	User's action
17	Communication error (network not reached)	The network was not connected.	Contact the system administrator. (*)
18	Communication error (network down)	The interface connected to the network was down.	
19	Communication error (port No error)	A port number could not be fetched.	
20	Communication error (IP address error)	An IP address could not be fetched.	
21	Communication error (memory attach failed)	Shared memory could not be attached.	
22	Communication error (trace file cannot open)	A trace file could not be opened.	
23	Communication error (trace file cannot copy)	A trace file could not be copied.	
24	Communication error (fatal error)	A fatal error was detected.	
25	Communication error (cannot connection errno = %x)	A communication line could not be connected.	
26	Communication error (RRB errno = %x)	Memory could not be read.	
27	Memory access error	Memory in the S10V could not be read or written.	
28	target status error	An S10V command support task could not be started.	
30	Cannot get TCB	TCB information could not be read.	
31	command is already execution	The command could not be executed because another user was measuring a PU load ratio.	
32	No sitename given for -u option	A site name was missing.	Check the data that can be entered, then reexecute the command.
33	Site=%s not found	No such site could be found.	
34	PU load measuring period error [second = 1-3600]	A measurement time was specified incorrectly.	

(*) For errors requiring you to “Contact the system administrator” as a remedial action, collect the following data for investigation:

- %windir%\renix\etc\log\s10v\1_RPDP_CTL, Rpdpc_t1

(9) Error messages displayed by the svtimex command

No.	Error message	Explanation	User's action
1	Usage:svtimex [-u site] [tname/tn] [-t second]	An option was specified incorrectly.	Specify a correct option.
2	Unknown RSSITE	The RSSITE environment variable was not set.	Set the RSSITE environment variable and then try again.
3	memory allocation error (malloc, taskloadinfo area)	An area from which to read PU load ratio information could not be allocated.	It is conceivable that the error occurred because the available memory was temporarily insufficient. Verify that the available memory is adequate, and then reexecute the command.
4	cannot get task load information	A task load ratio could not be fetched.	Contact the system administrator. (*)
5	cannot get task load information Taskname=%s(%s)	A task load ratio could not be fetched. (The task was identifiable.)	
6	Not available parameter	An unusable parameter was detected.	
7	No such AREA (puloading) in salmt	An area from which to read task load ratios could not be allocated.	
8	Communication error (catch signal)	A signal was received.	
9	Communication error (connection timeout)	A time-out was generated.	
10	Communication error (connection refused)	No RPC server was existent.	
11	Communication error (connection cut)	The RPC server was disconnected.	
12	Communication error (connection reset)	A connection was reset.	
13	Communication error (server closed)	The RPC server was closed.	
14	Communication error (port busy)	The line port was busy.	
15	Communication error (bad socket specified)	A socket descriptor was specified incorrectly.	

D. ERROR MESSAGES

No.	Error message	Explanation	User's action
16	Communication error (socket creat error)	A socket could not be created.	Contact the system administrator. (*)
17	Communication error (no buffer)	Memory could not be allocated.	It is conceivable that the error occurred because the available memory was temporarily insufficient. Verify that the available memory is adequate, and then reexecute the command.
18	Communication error (network not reached)	The network was not connected.	Contact the system administrator. (*)
19	Communication error (network down)	The interface connected to the network was down.	
20	Communication error (port No error)	A port number could not be fetched.	
21	Communication error (IP address error)	An IP address could not be fetched.	
22	Communication error (memory attach failed)	Shared memory could not be attached.	
23	Communication error (trace file cannot open)	A trace file could not be opened.	
24	Communication error (trace file cannot copy)	A trace file could not be copied.	
25	Communication error (fatal error)	A fatal error was detected.	
26	Communication error (cannot connection errno = %x)	A communication line could not be connected.	
27	Communication error (RRB errno = %x)	Memory could not be read.	
28	Memory access error	Memory in the R600 could not be read or written.	
29	target status error	An R600 command support task could not be started.	
30	Cannot get TCB	TCB information could not be read.	
31	command is already execution	The command could not be executed because another user was measuring task load ratios.	

No.	Error message	Explanation	User's action
32	No sitename given for -u option	A site name was missing.	Check the data that can be entered, then reexecute the command.
33	Site=%s not found	The specified site could not be found.	
34	Task measuring period error [second = 1-86400]	A measurement time was specified incorrectly.	
35	taskname or number set data over (max=10)!!	More than 10 task names or task numbers were given.	
36	taskname or number error(%s)	A task name or task number was specified incorrectly.	
37	%s(%s) task non exist or unmatched	The specified task was not registered in both the development machine and controller.	Register the task in both the development machine and controller.

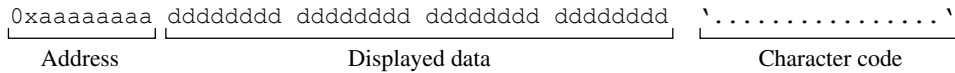
(*) For errors requiring you to “Contact the system administrator” as a remedial action, collect the following data for investigation:

- %windir%\renix\etc\log\s10v\1_RPDP_CTL, Rpdp_ctl

APPENDIX G DISPLAY FORMATS OF md AND sd OF debughr (ONLINE DEBUGGER)

(1) Display format of md subcommand

- Format of display (print) data



Address First address of the displayed data in hexadecimal.

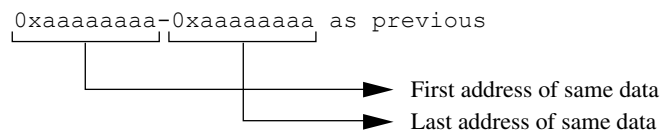
Displayed data The content at the address is displayed by the specified data length in the specified output format. Up to 16 bytes of data can be displayed on one single line.

When the floating-point data output form is used (-f -l, -fd) and the following data is at the address, the data is displayed in hexadecimal notation. Further, the corresponding character string is displayed after the hexadecimal display.

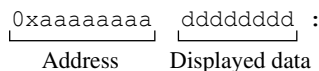
Floating-point data	Character string	Display example	
		Single precision	Double precision
Nonnumeric	Na	0x7fffffff:Na	0xfff00000 0x00000001:Na
Infinite	In	0x7f800000:In	0xfff00000 0x00000000:In
Maximum expressible value	Ma	0x7f7fffff:Ma	0x7fffffff 0xffffffff:Ma
Minimum expressible value	Mi	0xff7fffff:Mi	0xffefffff 0xffffffff:Mi

Character code The data for display is displayed in 2-byte code.

When a line has the same data as the previous line, the following message is displayed instead of repeating the data in the above format (When the -all option is specified, the display shows all consecutive data.):



- Format of modification (patch) data



G. DISPLAY FORMATS OF md AND sd OF debughr (ONLINE DEBUGGER)

● Examples

The following are examples displayed by md with various combinations of data output format options and data length options:

Four-byte display in hexadecimal (-h, -l)

```
*****|*****|*****|*****|*****|*****|*****|*****|
0x00ec0000 0000000a 00000064 000003e8 00002710 \.....d.....'\
0x00ec0010 000186a0 \.....\
```

Two-byte display in hexadecimal (-h, -w)

```
0x00ec0000 0000 000a 0000 0064 0000 03e8 0000 2710 \.....d.....'\
0x00ec0010 0001 86a0 \.....\
```

One-byte display in hexadecimal (-h, -b)

```
0x00ec0000 00 00 00 0a 00 00 00 64 00 00 03 e8 00 00 27 10 \.....d.....'\
0x00ec0010 00 01 86 a0 \.....\
```

Four-byte display in decimal (-d, -l)

```
0x00ec0000          10          100          1000          10000
0x00ec0010        100000
```

Two-byte display in decimal (-d, -w)

```
0x00ec0000          0          10          0          100          0          1000          0 10000
0x00ec0010          1 -31072
```

One-byte display in decimal (-d, -b)

```
0x00ec0000          0          0          0          10          0          0          0          100          0          0          3 -24          0          0
          39          16
0x00ec0010          0          1 -122 -96
```

Single-precision real number display (-f, -l)

```
0x00ec0020          1.1200000          2.1229999          10.1230001          20.1233997
0x00ec0030          100.123451
```

Double-precision real number display (-fd)

```
0x96000000          1.0000000000000000e+00          2.0000000000000000e+00
0x96000010          -1.0000000000000000e+00          1.0000000000000000e+100
0x96000020          0x7fefffff 0xfffffff:Ma 0xfffffff 0xfffffff:Mi
```


G. DISPLAY FORMATS OF md AND sd OF debughr (ONLINE DEBUGGER)

● Examples

The following are examples displayed by sd with various combinations of data output format options and data length options:

```

*****|*****|*****|*****|*****|*****|*****|*****|
Four-byte display in 0x00ec0000 (0x000000) 0000000a 00000064 000003e8 00002710 \.....d.....'\
hexadecimal (-h, -l) 0x00ec0010 (0x000010) 000186a0 \.....'\

Two-byte display in 0x00ec0000 (0x000000) 0000 000a 0000 0064 0000 03e8 0000 2710 \.....d.....'\
hexadecimal (-h, -w) 0x00ec0010 (0x000010) 0001 86a0 \.....'\

One-byte display in 0x00ec0000 (0x000000) 00 00 00 0a 00 00 00 64 00 00 03 e8 00 00 27 10 \.....d.
hexadecimal (-h, -b) .....'\
0x00ec0010 (0x000010) 00 01 86 a0 \.....'\

Four-byte display in 0x00ec0000 (0x000000) 10 100 1000 10000
decimal (-d, -l) 0x00ec0010 (0x000010) 100000

Two-byte display in 0x00ec0000 (0x000000) 0 10 0 100 0 1000 0 10000
decimal (-d, -w) 0x00ec0010 (0x000010) 1 -31072

One-byte display in 0x00ec0000 (0x000000) 0 0 0 10 0 0 0 100 0 0 3 -24
decimal (-d, -b) 0 0 39 16
0x00ec0010 (0x000010) 0 1 -122 -96

Single-precision real 0x00ec0020 (0x000000) 1.1200000 2.1229999 10.1230001 20.1233997
number display (-f, -l) 0x00ec0030 (0x000010) 100.123451

Double-precision real 0x96000000 (0x000000) 1.1000000000000000e+00 2.0000000000000000e+00
number display (-fd) 0x96000010 (0x000010) -1.1000000000000000e+00 1.0000000000000000e+100
0x96000020 (0x000020) 0x7fefffff 0xffffffff:Ma 0xffefffff 0xffffffff:Mi

```


APPENDIX H LIST OF STACK SIZES FOR LIBRARY USE

The stack sizes used by the libraries are listed below:

(1/2)

Library	Function name	Stack size
C standard library (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
	strspn	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	

H. LIST OF STACK SIZES FOR LIBRARY USE

(2/2)

Library	Function name	Stack size
C standard library (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
fpgetround		0
fpgetsticky		0
fpsetmask		0
fpsetround		0
fpsetsticky		0
fpcheck		0
fpchecko		0
libfirad.lib	irglbad	0
	irsubad	0
libcpms.lib	memcpy (*)	28

(*) In a program or subprogram loaded by the loader, the CPMS library's memcpy() is used instead of the C standard library's memcpy().

APPENDIX E NOTE ON USE OF RPDP

(1) Recovery from suspension of the processing by the ld subcommand of the svdebug command

When processing by the ld subcommand of the svdebug command is suspended due to, for example, a communication error, an inconsistency may take place between memory in the PCs and the backup file for the site in the development machine. To prevent this, RPDP recovers the ld subcommand. In recovery processing, the suspended ld subcommand is re-executed to put the subcommand into the state after execution. Recovery processing is performed when the RPDP command is executed or the development machine is started up. When the communication error is not removed, however, an error message such as “Communication error (connection time-out)” or “Communication error (inter-PU communication timeout)” is displayed, disabling the use of the RPDP command. To cope with this, perform the operation shown below.

Operation

```
sitecntlhr -rssrcv site_name [Enter]
```

After the operation, the command can be executed without the need for RPDP to perform recovery processing. However, note that since the ld subcommand is not yet recovered, no other ld subcommand must be executed for the site after the operation.

The error description is given below.

Cannot use ld sub comannd after RSSRCV set (NO.2100-44)

This restriction continues until the svrpl command is executed for downloading into the site.

(2) Notes on use

[Notes on operation]

- When transferring a text file to be entered into the debugger (svdebug) from another machine to the current using ftp, be sure to enable ascii mode.
- The svstedel command cannot delete a directory or file if it is being accessed by the file manager or command prompt. When deleting a directory or file being accessed by the file manager, first access it and then access another directory or file by double-clicking it. Another way is to close the file manager.
- When SIGINT is issued by pressing CTRL+C during the execution of a system generation command (svgen, svconf, svsitecp, svstedel), the command may terminate abnormally. This happens when SIGINT is issued while another command is being executed by the system generation command. In this case, the site to be handled is invalidated. Delete the site using the svstedel command.

[Restrictions]

- The following names are reserved. These names, including those followed by extensions such as “.c”, “.obj”, and “.txt”, cannot be used as site names, unit names, directory names, or file names.
 - AUX • CON • NUL
 - COM1 • LPT1 • PRN
 - COM2 • LPT2
 - COM3 • LPT3
 - COM4 • LPT4
 - COM5 • LPT5
 - COM6 • LPT6
 - COM7 • LPT7
 - COM8 • LPT8
 - COM9 • LPT9

- A file on disk in another machine cannot be specified in the svdebug ld, cm, or sv subcommand with the -f option specified.

APPENDIX F DISPLAY FORMAT OF svmap

The following map information is to be output:

- (1) Header and footer
- (2) Global area information
- (3) Split area information
- (4) Secondary partition area information
- (5) Program information
- (6) Subprogram information
- (7) Task information
- (8) Global information
- (9) VAL information
- (10) IRSUB entry information
- (11) IRGLB entry information
- (12) ULSUB entry information
- (13) Information about the physical memory's free space

<Map information output forms>

The map information can be output in the following forms:

- (1) Hierarchical map output
- (2) Listing in the order of addresses
- (3) Listing in the order of names
- (4) Listing in the order of numbers
- (5) Specified name output

The hierarchical map output form is used to hierarchically output the map information about resources that are arranged in a logical space on an individual global or split area basis.

The listing forms are used to output specified information in the order of addresses, names, or numbers. Further, the name of a resource can be specified to output the information about that name.

The map information output formats are described below.
 The underlined () portions in the following display formats are the output map information, which varies with the map output target.

(1) Header and footer

The map information is to be output with a header and a footer attached respectively to its beginning and the end. The header and footer formats are as shown below.

(a) Header

```

** allocator map **                                YYYY/MM/DD hh:mm:ss
site name = site
  
```

**** allocator map **:**
 Displays a header string.

**** allocator map **:** This header is used for normal map output.
**** allocator map (CON) **:** This header is used for PCs logical space map output (-CON specified).

YYYY/MM/DD hh:mm:ss:
 Displays a time at which the map output command (svmap) was started.
 YYYY: Year (4-digit)
 MM: Month
 DD: Day
 hh:mm:ss: hour:minute:second

site: Displays the name of a site that is targeted for map information display.

(b) Footer

```

** map output end **
  
```

F. DISPLAY FORMAT OF svmap

(2) Global area information

The map of global areas defined at the time of system generation is to be displayed. The start address of a global area's logical space is fixed.

```

< garea >
gname      laddr      paddr      size
$MAP       20000000  paddr     size
$TASK      30000000  paddr     size
$GLBR      40000000  paddr     size
$GLBRW     50000000  paddr     size
$IRSUB     60000000  paddr     size
    
```

gname: Name of a global area.

laddr: Logical address of the beginning of a global area.

paddr: Physical address of the beginning of a global area.

size: Size of a global area.

Table F-1 Real-Time Source Management Status

Symbol	Status	Meaning
@	not-build	Loaded into a backup file only.
+	defined-POC	
.	defined	Loaded into a backup file as well as a real machine's memory.
-	defined-CON	Loaded into a real machine's memory only. Deleted from the development machine only after a download.
*	unmatch	Loaded into a backup file as well as a real machine's memory but mismatched.
_	non_exist2	Downloaded without executing dload for an IRSUB/built-in subprogram for which dbuild has been executed.

(When -CON is specified, the display does not show s, date, lddate, or svdate.)

(4) Secondary partition area information

The information about secondary partition areas is to be displayed.

```

< sarea >
garea/aname/sname          raddr  size  laddr  date  svdate
garea/aname/sname         s_k raddr size  laddr YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
garea/aname/              raddr size  laddr
    
```

- garea:** Shows the GAREA name of a parent global area.
The meaning of the displayed GAREA name is the same as for “(3) Split area information.”
- aname:** Shows the name of a split area.
- sname:** Shows the name of a secondary partition area.
A blank secondary partition area name field represents an unoccupied area.
- s:** Indicates the resource status.
- k:** For details on the resource status, see Table F-1.
- raddr:** Indicates the owner type (s: system; u: user).
- size:** Shows the relative byte address of the beginning of a secondary partition area in relation to the beginning of a split area in 8-digit hexadecimal notation at all times.
- laddr:** Shows the size of a secondary partition area in 8-digit hexadecimal notation at all times.
- date:** Shows the start logical address of a secondary partition area in 8-digit hexadecimal notation at all times.
- svdate:** Shows the time at which a secondary partition area was allocated with svdfs or the time at which a program or subprogram was loaded into a backup file with svload.
- lddate:** Shows the time of downloading into the PCs memory.
When such a download is not completed, this field is blank.
- svdate:** Shows the time at which the debugger’s sv subcommand was used for storage into a backup file.
When such a storage operation is not completed, this field is blank.

The display shows “date,” “lddate,” and “svdate” only when the -f option is specified (“s,” “date,” “lddate,” and “svdate” do not appear on the display when -CON is specified).

(5) Program information

The information about programs is to be displayed.

```

< task-program >
tn  tname      tnox rmtn lvl sp  pname      st mt n  texttop  lastaddr  tsize  dsize  ssize (part )  bsize  extra  oswork  datatop  bsstop  date      lddate
   s  k  st  mt n  texttop  lastaddr  tsize  dsize  ssize (part )  bsize  extra  oswork  datatop  bsstop
tn  tname      s  k  st  mt n  texttop  lastaddr  tsize  dsize  ssize (part )  bsize  extra  oswork  datatop  bsstop  YYYY/MM/DD hh:mm:ss  YYYY/MM/DD hh:mm:ss
  (a) Task information field
  (b) Program information field
  (c) Detailed information field
    
```

The program information display format is the same for “(7) Task information.”

For the meaning of each field, see under “(7) Task information.”

There are the following two differences between program information display and task information display:

- By default, sorting is performed in the order of program names when a program is specified (-p) or in the order of task numbers when task information is specified (-t).
- When information is displayed with a name specified, the specified name is handled as a program name if a program is specified (-p) or as a task name if task information is specified (-t).

(6) Subprogram information

The information about subprograms is to be displayed.

```

< IRSUB > [max_entry=mentry, use_entry=umentry]
irno  entname      st  laddr  subname      offset  texttop  bsslst  tsize  dsize  bsize  extra  ssize (part )  datatop  bsstop  date      lddate
   s  k  st  laddr  subname      000000  s  k  texttop  bsslst  tsize  dsize  bsize  extra  ssize (part )
irno  entname      s  k  st  laddr  subname      offset  s  k  texttop  bsslst  tsize  dsize  bsize  extra  ssize (part )  datatop  bsstop  YYYY/MM/DD hh:mm:ss  YYYY/MM/DD hh:mm:ss
irno  entname      s  k  st  laddr  subname      offset  s  k  texttop  bsslst  tsize  dsize  bsize  extra  ssize (part )  datatop  bsstop  YYYY/MM/DD hh:mm:ss  YYYY/MM/DD hh:mm:ss
    
```

```

< ULSUB >
pnt  typ  b  ent  subname      st  k  texttop  bsslst  tsize  dsize  bsize  extra  ssize (part )  datatop  bsstop  date      lddate
   s  k  texttop  bsslst  tsize  dsize  bsize  extra  ssize (part )  datatop  bsstop  YYYY/MM/DD hh:mm:ss  YYYY/MM/DD hh:mm:ss
    
```

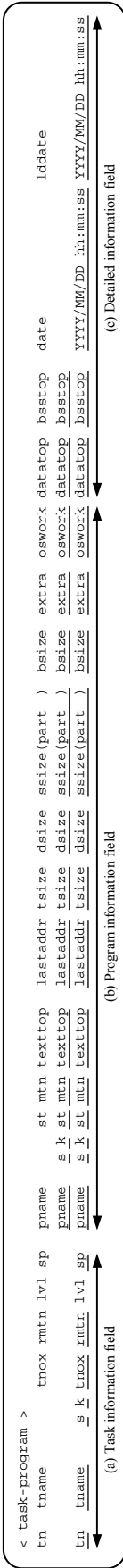
The subprogram information display format is the same as for “(10) IRSUB entry information” and “(12) ULSUB entry information.”

The information about both IRSUBs and built-in subprograms appear on the display.

For the meaning of each field, see under “(10) IRSUB entry information” and “(12) ULSUB entry information.”

(7) Task information

The information about tasks is to be displayed.



(a) Task information field

- tn: Shows a task number in 4-digit decimal notation at all times.
- tname: Shows the name of a task.
- s: Shows the status of a task.
- k: For resource status, see Table F-1.
- tnox: Indicates the owner type (s: system; u: user).
- rmt n: Shows a task number in 4-digit hexadecimal notation at all times.
- l v l: Shows a multitask number (used stack position) in 4-digit hexadecimal notation at all times. Reads "0001" for a single task.
- s p: Indicates a task level in 2-digit hexadecimal notation at all times.
- s p: Shows the end logical address of the stack used by a task in 8-digit hexadecimal notation at all times.
- Note: For a program not generated as a task, the task information field is blank.

(b) Program information field

- pname: Shows the name of a program.
- s: Shows the status of a program.
- k: For resource status, see Table F-1.
- s t: Indicates the owner type (s: system; u: user).
- s t: Indicates the usage of a program.
- l s: Indicates that the program is not registered as a task.
- l m: Indicates that the program is loaded as a multitask and not registered as a task.
- c s: Indicates that the program is registered as a task.
- c m: Indicates that the program is loaded as a multitask and registered as a task.

mt n:

Indicates the number of tasks contained in a multitask in 4-digit hexadecimal notation at all times. Reads "0001" for a single task.

t e x t t o p:

Shows the start logical address of a text division in 8-digit hexadecimal notation at all times.

l a s t a d d r:

Shows the end logical address of a program in 8-digit hexadecimal notation at all times.

t s i z e:

Shows the size of a text division in 6-digit hexadecimal notation at all times.

d s i z e:

Shows the size of a data division in 6-digit hexadecimal notation at all times.

s s i z e (p a r t):

Shows the size of a stack in 6-digit hexadecimal notation at all times. The "(part)" section shows the stack size for use by the program specified for the loader.

b s i z e:

Shows the size of a bss division in 6-digit hexadecimal notation at all times.

e x t r a:

Shows the redundancy byte size specified for a "load" operation in 6-digit hexadecimal notation at all times.

o s w o r k:

Shows the size of OS work in 6-digit hexadecimal notation at all times.

(c) Detailed information field

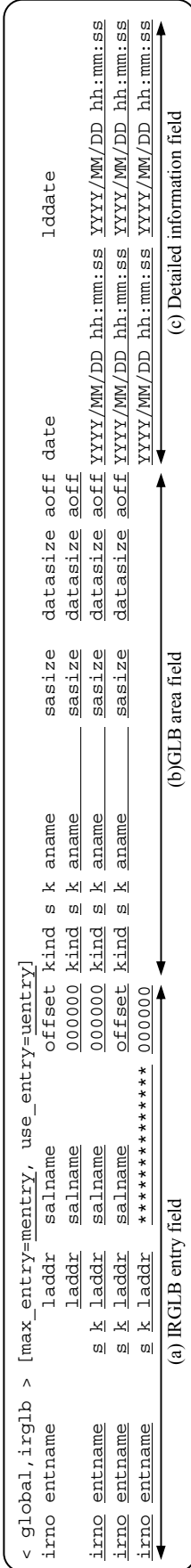
- d a t a t o p: Shows the start logical address of a data division in 8-digit hexadecimal notation at all times.
 - b s s t o p: Shows the start logical address of a BSS division in 8-digit hexadecimal notation at all times.
 - d a t e: Shows the time at which a task was generated with svctask. If no such task is generated, this field is blank.
 - l d d a t e: Shows the time of downloading into the PCs memory. If such a download is not completed, this field is blank.
- The detailed information field appears only when the -f option is specified.

(10) IRSUB entry information
The information about IRSUB entries is to be displayed.

(a) IRSUB entry information field			(b) Subprogram information field			(c) Detailed information field													
<code>< IRSUB ></code>	<code>[max_entry=entry, use_entry=entry]</code>	<code>offset</code>	<code>texttop</code>	<code>bslast</code>	<code>tsize</code>	<code>dsize</code>	<code>bsize</code>	<code>extra</code>	<code>ssize</code>	<code>(part)</code>	<code>datatop</code>	<code>bsstop</code>	<code>date</code>	<code>lddate</code>					
<code>irno</code>	<code>entname</code>	<code>s k</code>	<code>st</code>	<code>laddr</code>	<code>subname</code>	<code>000000</code>	<code>s k</code>	<code>texttop</code>	<code>bslast</code>	<code>tsize</code>	<code>dsize</code>	<code>bsize</code>	<code>extra</code>	<code>ssize</code>	<code>(part)</code>	<code>datatop</code>	<code>bsstop</code>	<code>date</code>	<code>lddate</code>
		<code>s k</code>	<code>st</code>	<code>laddr</code>	<code>subname</code>	<code>offset</code>	<code>s k</code>	<code>texttop</code>	<code>bslast</code>	<code>tsize</code>	<code>dsize</code>	<code>bsize</code>	<code>extra</code>	<code>ssize</code>	<code>(part)</code>	<code>datatop</code>	<code>bsstop</code>	<code>date</code>	<code>lddate</code>
		<code>s k</code>	<code>st</code>	<code>laddr</code>	<code>subname</code>	<code>000000</code>	<code>s k</code>	<code>texttop</code>	<code>bslast</code>	<code>tsize</code>	<code>dsize</code>	<code>bsize</code>	<code>extra</code>	<code>ssize</code>	<code>(part)</code>	<code>datatop</code>	<code>bsstop</code>	<code>date</code>	<code>lddate</code>
		<code>s k</code>	<code>st</code>	<code>laddr</code>	<code>subname</code>	<code>offset</code>	<code>s k</code>	<code>texttop</code>	<code>bslast</code>	<code>tsize</code>	<code>dsize</code>	<code>bsize</code>	<code>extra</code>	<code>ssize</code>	<code>(part)</code>	<code>datatop</code>	<code>bsstop</code>	<code>date</code>	<code>lddate</code>

- mentry:** Shows the number of registrable IRSUB entries in 6-digit decimal notation at all times.
- uentry:** Shows the number of currently used IRSUB entries in 6-digit decimal notation at all times.
- (a) IRSUB entry information field
- irno:** Indirect link table entry number. If the IRSUB is not built, this “irno” field is blank.
 - entname:** IRSUB entry name.
 - s:** Shows the status of an IRSUB entry.
For resource status, see Table F-1.
 - k:** Indicates the owner type (s: system; u: user).
(If the IRSUB is not built, this field is blank.)
 - st:** Shows the IRSUB entry type and assignment.
i.l: Indicates the top entry function of an IRSUB that is not built.
m.l: Indicates the multi-entry function of an IRSUB that is not built.
i.b: Indicates the top entry function of an IRSUB that is built.
m.b: Indicates the multi-entry function of an IRSUB that is built.
- laddr:** Shows the logical space address of an entry point in 8-digit hexadecimal notation at all times.
- subname:** Shows the name of a subprogram that contains an entry.
- offset:** Shows the relative offset between the beginning of a subprogram and an entry point in 6-digit hexadecimal notation at all times.
- (b) Subprogram information field
- s:** Indicates the status of a subprogram.
For resource status, see Table F-1.
 - k:** Indicates the owner type (s: system; u: user).
- texttop:** Shows the start logical address of a text division in 8-digit hexadecimal notation at all times.
- bslast:** Shows the end logical address of a bss division in 8-digit hexadecimal notation at all times.
The end logical address is a value that includes the redundancy byte size (extra).
- tsize:** Shows the size of a text division in 6-digit hexadecimal notation at all times.
- dsiz:** Shows the size of a data division in 6-digit hexadecimal notation at all times.
- bsiz:** Shows the size of a bss division in 6-digit hexadecimal notation at all times.
- extra:** Shows the redundancy byte size specified for a “load” operation in 6-digit hexadecimal notation at all times.
- ssize (part):** Shows the size of a stack in 6-digit hexadecimal notation at all times. The “(part)” section shows the stack size for use by the subprogram specified for the loader.
- (c) Detailed information field
- datatop:** Shows the start logical address of a data division in 8-digit hexadecimal notation at all times.
- bsstop:** Shows the start logical address of a BSS division in 8-digit hexadecimal notation at all times.
- date:** Shows the time at which a build was performed with svbuild. If no such build was performed, this field is blank.
- lddate:** Shows the time of downloading into the PCs memory. If such a download is not completed, this field is blank.
- The detailed information field appears only when the -f option is specified.

(11) IRGLB entry information



mentry: Shows the number of registrable GLB entries in 6-digit decimal notation at all times.
uentry: Shows the number of currently used GLB entries in 6-digit decimal notation at all times (including the 13 GLB entries for use by the OS).

(a) IRGLB entry field

irno: Indirect link table entry number. (If the entry is not built as an IRGLB, this field is blank.)
entname: IRGLB entry name. (If the entry is not built as an IRGLB, this field is blank.)
s: Indicates the IRGLB entry status. For resource status, see Table F-1. (If the entry is not built as an IRGLB, this field is blank.)
k: Indicates the owner type (s: system; u: user). (If the entry is not built as an IRGLB, this field is blank.)

laddr: Shows the logical space address of an entry point in 8-digit hexadecimal notation at all times. (If the entry is not built as an IRGLB, this field shows the GLB (sarea) address.)

salname: Shows the name of a secondary partition area containing an entry. If the entry address is given as an absolute address, this field displays a string of asterisks (*).

offset: Shows the relative offset between the beginning of a secondary partition area and an entry point in 6-digit hexadecimal notation at all times.

(b) GLB area field

kind: Secondary partition area type.
glbi: Represents a read/write global area with an initial value.
glbw: Represents a read/write global area without an initial value.
glbr: Represents a read-only global area with an initial value.
s: Indicates the GLB status. For resource status, see Table F-1.

k: Indicates the owner type (s: system; u: user).
aname: Shows the name of a parent split area.
sasize: Indicates the area size in 8-digit hexadecimal notation at all times.
datasize: Indicates the data size in 8-digit hexadecimal notation at all times.
aoff: Shows the relative byte address in relation to the beginning of a split area in 8-digit hexadecimal notation at all times.

(c) Detailed information field

date: Shows the time at which a build was performed to create an IRGLB with svirglb or svdfs -e. If an IRGLG is not built, this field is blank.
lddate: Shows the time of downloading into the PCs memory. If such a download is not completed, this field is blank.

The detailed information field appears only when the -f option is specified.

(13) Information about the physical memory's free space

```
< physical memory >
garea      use      free      total
$TASK      xxx  kbyte  yyy  kbyte  zzz  kbyte
$GLBR      xxx  kbyte  yyy  kbyte  zzz  kbyte
$GLBRW     xxx  kbyte  yyy  kbyte  zzz  kbyte
$IRSUB     xxx  kbyte  yyy  kbyte  zzz  kbyte
```

use: Size of the physical memory used by a GAREA.
 free: Size of a free physical memory space for a GAREA.
 total: Physical memory size that was allocated to a GAREA at the time of site construction.

(14) Hierarchical map output

The hierarchical map output is to be generated in the following form.

(a) garea/area hierarchical map (+gn, gname, -G, and -a specified)

```
** allocator map **
site name = site
< garea >
gname      laddr  paddr  size
gname      laddr  paddr  size
< area >
garea/aname      raddr  size  laddr  bkupfile  date  lddate  svdate
garea/aname      s  k  raddr  size  laddr  bkupfile  YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
garea/          raddr  size  laddr
** map output end **
```

Note: The display shows "date," "lddate," and "svdate" only when the -f option is specified.

(b) garea/area/sarea hierarchical map (+gn, gname, -G, -a, and -e specified)

```

** allocator map **
      YYYY/MM/DD hh:mm:ss
site name = site
< garea >
gname      laddr  paddr  size
gname      laddr  paddr  size
< area >
garea/area raddr  size
garea/area s k raddr size
garea/area raddr size
date      lddate  svdate
YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
< sarea >
garea/aname/sname
garea/aname/sname
garea/aname/
      raddr  size  laddr  lddate  svdate
      raddr size laddr lddate svdate
      YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
** map output end **

```

Note: The display shows “date,” “lddate,” and “svdate” only when the -f option is specified.

(c) area/sarea hierarchical map (+gn, aname, -a, and -e specified)

```

** allocator map **
      YYYY/MM/DD hh:mm:ss
site name = site
< area >
garea/aname      raddr  size  laddr  bkupfile  date  svdate
garea/aname      raddr size laddr bkupfile YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
< sarea >
garea/aname/sname
garea/aname/sname
      raddr  size  laddr  lddate  svdate
      raddr size laddr lddate svdate
      YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
** map output end **

```

Note: The display shows “date,” “lddate,” and “svdate” only when the -f option is specified.

(15) Default display format

(a) Default display format (no detailed information)

If you specify the -u option and leave all the other options unspecified, the system outputs split area/secondary partition area listings sorted by address, task/program/IRSUB/built-in subroutine/IRGLB listings sorted by number, and VAL listings sorted by name with no detailed information attached. The employed display form is shown below:

```

** allocator map **
site name = site
< garea >
gname laddr paddr size
gname laddr paddr size
< area >
garea/aname raddr size kind bkupfile
garea/aname s k raddr size kind bkupfile
garea/aname raddr size
< sarea >
garea/aname/sname raddr size laddr
garea/aname/sname s k raddr size laddr
garea/aname/ raddr size
< task-program >
tn tname tnox rmtn lvl sp pname size laddr
tn tname s k tnox rmtn lvl sp pname size laddr
tn tname s k tnox rmtn lvl sp pname size laddr
< IRSUB > [max_entry=entry, use_entry=entry]
irno entname st laddr subtype offset
entname s k st laddr subtype 000000
irno entname s k st laddr subtype offset
irno entname s k st laddr subtype 000000
irno entname s k st laddr subtype offset
< ULSUB >
pnt typ ent b subtype texttop bsslast tsize dsize extra ssize(part )
pnt typ ent b subtype s k texttop bsslast tsize dsize extra ssize(part )
< global,irglb > [max_entry=entry, use_entry=entry]
irno entname laddr salname offset kind s k aname ssize datasize aoff
irno entname s k laddr salname 000000 kind s k aname ssize datasize aoff
irno entname s k laddr salname 000000 kind s k aname ssize datasize aoff
irno entname s k laddr ***** offset kind s k aname ssize datasize aoff
< value > [max_entry=entry, use_entry=entry]
ename valhex valdec
ename k valhex valdec
** map output end **

```

(b) Default display format (detail display)

If you specify the -u and -f options and leave all the other options unspecified, the system outputs split area/secondary partition area listings sorted by address, task/program/IRSUB/built-in subroutine/IRGLB listings sorted by number, and VAL listings sorted by name with detailed information attached. The employed display form is shown below:

```

** allocator map **
site name = site
< garea >
gname laddr paddr size
laddr paddr size
< area >
garea/aname raddr size laddr kind bkupfile
garea/aname s k raddr size laddr kind bkupfile
garea/ raddr size laddr
< sarea >
garea/aname/ sname raddr size laddr raddr size svdate
garea/aname/ sname raddr size laddr raddr size svdate
garea/aname/ sname raddr size laddr raddr size svdate
< task-program >
tn tname tnox rmtn lvl sp pname et mtn texttop lastaddr tsize dsize ssize(part) bsize extra oswork datatop bsstop date lddate
tn tname s k tnox rmtn lvl sp pname s k st mtn texttop lastaddr tsize dsize ssize(part) bsize extra oswork datatop bsstop YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
< IRSUB > [max_entry=mentry, use_entry=umentry]
irno entname laddr subname offset texttop bsblast tsize dsize ssize(part) datatop bsstop date lddate
entname s k laddr subname offset 000000 s k texttop bsblast tsize dsize ssize(part) datatop bsstop
entname s k laddr subname offset 000000 s k texttop bsblast tsize dsize ssize(part) datatop bsstop YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
irno entname s k laddr subname offset 000000 s k texttop bsblast tsize dsize ssize(part) datatop bsstop YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
< ULSUB >
pnt typ ent b subname texttop bsblast tsize dsize ssize(part) datatop bsstop date lddate
pnt typ ent b subname s k texttop bsblast tsize dsize ssize(part) datatop bsstop YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
< global,irglb > [max_entry=mentry, use_entry=umentry]
irno entname salname laddr offset kind s k aname ssize dataize aoff date lddate
irno entname s k laddr salname 000000 kind s k aname ssize dataize aoff
irno entname s k laddr salname 000000 kind s k aname ssize dataize aoff YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
irno entname s k laddr salname ***** kind s k aname ssize dataize aoff YYYY/MM/DD hh:mm:ss YYYY/MM/DD hh:mm:ss
< value > [max_entry=mentry, use_entry=umentry]
ename valhex valdec date
ename k_valhex valdec YYYY/MM/DD hh:mm:ss
** map output end **

```