HITACHI
S10α SERIES

2α SERIES

SOFTWARE MANUAL
OPERATION

RPDP/S10 For Windows®

Applicable to :

HITACHI-S10/2α	NESP-S25E
HITACHI-S10/2α E	NESP-2α E
HITACHI-S10/2α H	NESP-2α H
HITACHI-S10/2α Hf	NESP-2α Hf
S10mini model S	
S10mini model H	
S10mini model F	
S10mini model D	

SAE-3-133(B)

HITACHI

NOTE

PROHIBITION

TRADEMARKS

FIRST EDITION, NOVEMBER, 1998, SAE - 3 - 133 (A) (out of print)
SECOND EDITION, SEPTEMBER, 2003, SAE - 3 - 133 (B)

BI-KB-TN<IC-NS> (FL-MW20, AI8.0)

LIMITED WARRANTY

Hitachi, Ltd., warrants its products to be manufactured in accordance with published specifications and free from defects in materials and/or workmanship.

Hitachi, Ltd., warrants its products against defects in parts and workmanship for one full year from date of purchase.

HITACHI, LTD., MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED EXCEPT AS PROVIDED HEREIN, INCLUDING WITHOUT LIMITATION THEREOF, WARRANTIES AS TO MARKETABILITY FOR A PARTICULAR PURPOSE OF USE, OR AGAINST INFRINGEMENT OF ANY PATENT. IN NO EVENT SHALL HITACHI BE LIABLE FOR ANY DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY NATURE, OR COSTS, CHARGES, LOSSES OR EXPENSES RESULTING FROM ANY DEFECTIVE PRODUCT OR THE USE OF ANY PRODUCT.

SOFTWARE UP–TO DATE POLICY

Hitachi, Ltd., constantly reviews its software so as to incorporate the latest technology. Hitachi reserves the right to make changes to any software to improve reliability, function, or design. Hitachi cannot be held responsible for any errors in its software.

⚠ 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.

⚠ “RUN/STOP” SWITCH CAUTION

The “RUN/STOP” switch only stops execution of the ladder logic program or HI-FLOW program. Digital and analog outputs are left in the active state when execution stops, unless the optional rungs described in the CPU manual have been added. The “RUN/STOP” switch does not affect the operation of C-language or FA-BASIC language programs. Outputs can still be produced in response to C-language or FA-BASIC programs, or by the action of programmers typing in commands in these languages, while the “RUN/STOP” switch is in the “STOP” position.

DO NOT DEPEND ON THE STOP SWITCH TO STOP MOVING PARTS OR TO PREVENT UNEXPECTED MOTION OR ENERGIZATION. USE HARDWIRED SAFETY DISCONNECT AND LOCK OUT POWER AND CONTROL VOLTAGES BEFORE WORKING ON ELECTRICAL CIRCUITS OR PARTS THAT CAN MOVE.

PREFACE

We greatly appreciate your purchase of the RPDP/S10 system.

This manual describes how to create real-time programs that, under the HITACHI S10/2α series CPMS and its debugger, run on a personal computer in which Microsoft® Windows® is installed (simply called the personal computer throughout this manual).

This manual is intended for those users who have knowledge of the personal computer, Windows®, and MS-DOS®.
Development of real-time programs requires the developer to procure an MCP68K C Compiler Package (containing crossing C compiler MCC68K and crossing assembler ASM68K) separately. In addition to this package, it may also require the procurement of a text editor, depending on the development environment.
This manual applies to the following versions of system.

System name/version
RPDP/S10 SYSTEM For Windows® 03-03

For Microsoft® Windows® 95, 98, 2000, and MS-DOS®, crossing C compiler MCC68K, and crossing assembler ASM68K, refer to the their respective manuals.
For CPMS, refer to the manual listed below.

<Related manual>
SOFTWARE MANUAL GENERAL DESCRIPTION & MACROS COMPACT PMS V5
(Manual number SAE-3-201)

See the following list when you use the NESP (Nissan Electronic Sequence Processor) series.

【HITACHI-S10α series】		【NESP series】
HITACHI-S10/2α	·········	NESP-S25E
HITACHI-S10/2αE	·········	NESP-2αE
HITACHI-S10/2αH	·········	NESP-2αH
HITACHI-S10/2αHf	·········	NESP-2αHf

<Trademarks>
- Microsoft® Windows® operating system, Microsoft® Windows® 95 operating system, Microsoft® Windows® 98 operating system, Microsoft® Windows® 2000 operating system, Microsoft® Windows® XP operating system, MS-DOS® are registered trademarks of Microsoft Corporation in the United States and/or other countries.
- Ethernet is a registered trademark of Xerox Corp.
- MCP68K, MCC68K, and ASM68K are trademarks of MICROTEC: A Menter Graphics Company in the United States.

Other product names written in this manual are the trademarks of each manufacturer.

Systems Supported by Windows® 2000 and Windows® XP

The systems supported by Microsoft® Windows® 2000 operating system (hereafter abbreviated as Windows® 2000) and Microsoft® Windows® XP operating system (hereafter abbreviated as Windows® XP) are shown in the following table.

Systems of earlier versions than those shown in the following table are not supported by Windows® 2000 and Windows® XP but supported by only Microsoft® Windows® 95 operating system (hereafter abbreviated as Windows® 95) and Microsoft® Windows® 98 operating system (hereafter abbreviated as Windows® 98). (The system names in the following table are hereafter abbreviated as each system.)

<Table of Systems Supported by Windows® 2000 and Windows® XP>

No.	System name	Type	Version	Windows® 2000	Windows® XP
1	S10Tools SYSTEM	S-7890-01	07-05	√	√
2	LADDER CHART SYSTEM	S-7890-02	07-05	√	√
3	HI-FLOW SYSTEM	S-7890-03	07-02	√	√
4	CPMS LOADING SYSTEM	S-7890-04	07-04	√	√
5	CPMSE LOADING SYSTEM	S-7890-05	07-04	√	√
6	CPMS DEBUGGER SYSTEM	S-7890-06	07-02	√	√
7	CPMSE DEBUGGER SYSTEM	S-7890-07	07-02	√	√
8	GP-IB LOADING SYSTEM	S-7890-08	07-01	√	√
9	BACKUP RESTORE SYSTEM	S-7890-09	08-01	√	√
10	RPDP/S10 SYSTEM	S-7891-10	03-03	√ (*2)	ns (*1)
11	NX/Tools-S10 SYSTEM	S-7890-13	07-02	√	√
12	4α LADDER CHART SYSTEM	S-7890-17	07-05	√	√
13	4αH LADDER CHART SYSTEM	S-7890-18	07-05	√	√
14	LADDER COMMENT CONVERTER SYS	S-7890-19	06-01	√	√
15	HIGH SPEED REMOTE I/O SYSTEM	S-7890-21	07-01	√	√
16	CPU LINK SYSTEM	S-7890-22	07-01	√	√
17	4ch ANALOG PULSE COUNTER SYS	S-7890-23	07-01	√	√
18	EXTERNAL SERIAL LINK SYSTEM	S-7890-24	07-02	√	√
19	S10ET LINK SYSTEM	S-7890-25	07-02	√	√
20	J.NET SYSTEM	S-7890-27	07-02	√	√
21	OD.RING/SD.LINK SYSTEM	S-7890-28	07-03	√	√
22	ET.NET SYSTEM	S-7890-29	07-01	√	√
23	FL.NET SYSTEM	S-7890-30	07-03	√	√
24	D.NET SYSTEM	S-7890-31	07-04	√	√
25	LADDER CHART MONITOR SYSTEM	S-7890-34	07-04	√	√
26	HI-FLOW MONITOR SYSTEM	S-7890-35	07-01	√	√
27	IR.LINK SYSTEM	S-7890-36	07-02	√	√
28	Crossing C compiler (manufactured by Mentor graphics company)	MCP68K	5.3	√ (*2)	ns (*1)

√: Supported ns: Not supported

(*1) Crossing C compiler (No.28) is not supported by Windows® XP. Use it on Windows® 2000.

(*2) Crossing C compiler (No.28) must be a version supported by Windows® 2000 (later than version 5.3) as a premise.

Precautions on Using Windows® 2000

To install, uninstall or execute the Crossing C Compiler (MCP68K) and the RPDP/S10, set the user name to "Administrator" on the [Log On to Windows] window that is displayed when the PC is started. If the user name is set to any other than "Administrator", the Crossing C Compiler (MCP68K) and the RPDP/S10 cannot be installed, uninstalled or executed correctly.

Hardware and Software Requirements

Using each system requires the following hardware and software.

<Personal Computers (hereafter abbreviated as PC)>

Item \ OS	Windows® 95 (*1) Windows® 98 (*1)	Windows® 2000 (*1)	Windows® XP (*1) (*2)
CPU	Pentium 133 MHz or more	Pentium 300 MHz or more	
Memory (RAM)	32 MB or more	64 MB or more	128 MB or more
Free hard disk capacity (*3)	20 MB or more/system (However, 10 MB or more/system for OS loading and option module support software)		
Floppy disk drive	1 unit or more (required to install software by FD)		
CD-ROM drive	1 unit or more (required to install software by CD-ROM)		
Ethernet (10BASE-T)	1 port or more (required to connect a PC with the ET.NET module)		
Serial (D-sub 9-pin)	1 port or more (required to connect the PCs with a PC by RS-232C or set an IP address for the ET.NET module)		
PC card (conforming to the PC Card Standard (JEITA V4.2) TYPE II or TYPE III)	1 slot or more (required to connect a PC with the parallel interface module (LWZ400). At this time, the following GP-IB card is also required.) GP-IB card: PCMCIA-GPIB (Model: 777438-02) (manufactured by National Instruments Corporation)		
Display	Resolution of 800 × 600 pixels or more		
Microsoft® Internet Explorer	Version 4.01 or later		

(*1) For the OS service pack, refer to the attached reference materials for software.
(*2) No.10 and No.28 in <Table of Systems Supported by Windows® 2000 and Windows® XP> in "PREFACE" are excepted.
(*3) This is a capacity required to install each system. A free capacity to save user programs is also required.

Users are advised to use the CPMS and its debugger contained on system floppy disks with the version numbers indicated below. Any versions older than these will not run on the personal computers listed below.

Programmable controllers (PCs) used	System name/version
S10/2α	CPMS LOADING SYSTEM 03-00 or later CPMS DEBUGGER SYSTEM 03-00 or later
S10/2αE, 2αH, 2αHf	CPMS LOADING SYSTEM 03-00 or later CPMS DEBUGGER SYSTEM 03-00 or later

<Definitions of Terms>

N coil:	A ladder program converted into a form that can be run on the PCs by pasting a symbol on the sheet displayed on a PC.
Process:	A HI-FLOW program converted into a form that can be run on the PCs by pasting a symbol on the sheet displayed on a PC.
Compile:	To convert an application program such as a ladder chart and HI-FLOW into a form (N coil, process, etc.) that can be run on the PCs.
Build:	To compile only a corrected application program.
Rebuild:	To compile every existing application program.
Sheet:	Paper to prepare an application program of ladder chart and HI-FLOW, etc. This paper is controlled on a PC.
PCs:	An abbreviation of Programmable Controllers. This is a general term for PLC such as the S10α and S10mini series.
PLC:	An abbreviation of Programmable Logic Controller. This is an industrial electronic device to exert sequence control, having an incorporated program. The S10α and S10mini series come under this PLC.

<Note for storage capacity calculations>

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

CONTENTS

FIGURES

TABLES

1 OVERVIEW

1 OVERVIEW

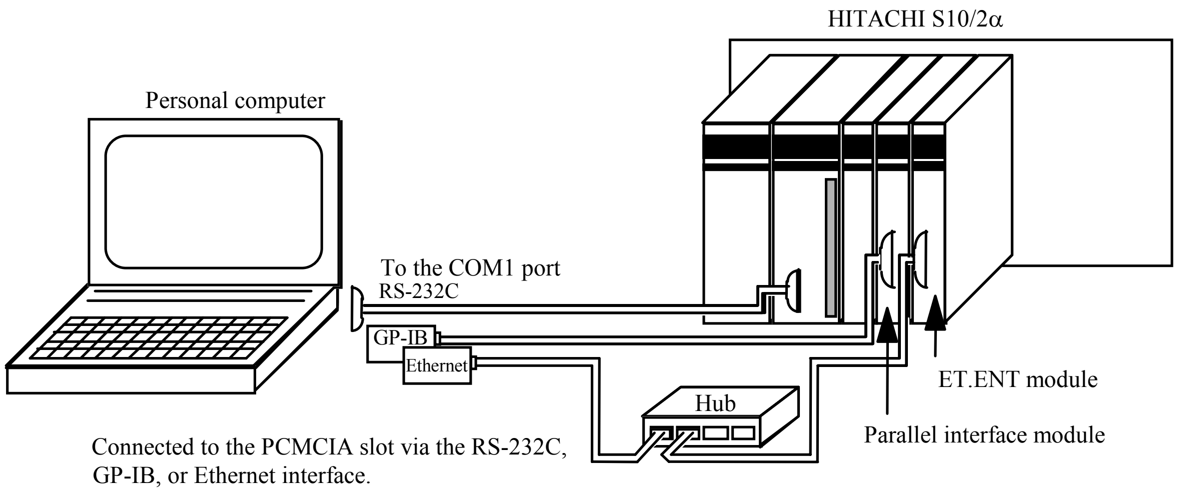
This chapter describes RPDP/S10 which provides a software development environment in which the user can develop C programs for use on HITACHI S10/2α series models.

1.1 RPDP/S10

The Real-time Program Developing Package for HITACHI S10/2α (RPDP/S10) is a tool that the user can use to develop programs for execution under the Compact Process Monitor System (CPMS). CPMS is the real-time operating system for HITACHI S10/2α series machines.
This tool runs under MS-DOS that is an operating system for personal computers.
Programs that run under CPMS perform high-speed real-time processing by using attributes and features that differ from those for programs that run under MS-DOS. To provide these attributes and features, a dedicated development system, called RPDP/S10, is used for development of real-time programs under CPMS.
Below are differences in attributes and features between programs under CPMS and those under MS-DOS.

(1) Programs under MS-DOS are loaded into main memory when they are requested to be started. By contrast, programs under CPMS are loaded into main memory in advance, minimizing the time taken from when a start request is made until their execution starts. These programs are called resident programs. The RPDP allocator manages main memory so that resident programs are placed in main memory in an efficient manner without duplication. When creating programs under CPMS, the user must determine the way the user works with main memory by using the RPDP allocator.

(2) All subprograms used by a program under MS-DOS are combined into a single program group as internal subprograms. Under CPMS, each subprogram can reside in main memory, independently of the program. This allows multiple programs to share a single subprogram to use memory more efficiently. These subprograms are called resident subprograms (RSUBs). When linking programs, the user can use the dedicated loader (sload). The loader does not link all subprograms as internal subprograms. Rather it links subprograms to the program while leaving them outside as defined as RSUBs.
With RPDP, the loader (sload) calls the linker to link programs.


HITACHI S10/2α
Personal computer
To the COM1 port
RS-232C
GP-IB
Ethernet
Hub
ET.ENT module
Parallel interface module
Connected to the PCMCIA slot via the RS-232C, GP-IB, or Ethernet interface.

(3) Under MS-DOS, data is exchanged between multiple programs. To make this possible, a pipelining feature is used for data passing and reception. In real-time processing, multiple programs (tasks) are running while maintaining their mutual association by using global (GLB) areas, which are allocated as data areas to be shared by tasks. These areas are also managed by the allocator. When a program accesses data in a global area, its absolute address is used to assure high-speed access. For example, if a program containing references to a GLB area, it is necessary to find the location of the GLB area and embed the GLB address in the text and data portions of the program. RPDP supports a loader (sload) that does the job. The loader links addresses with RSUBs and embeds constants, called values (VALs), common to all programs in the programs and their data.

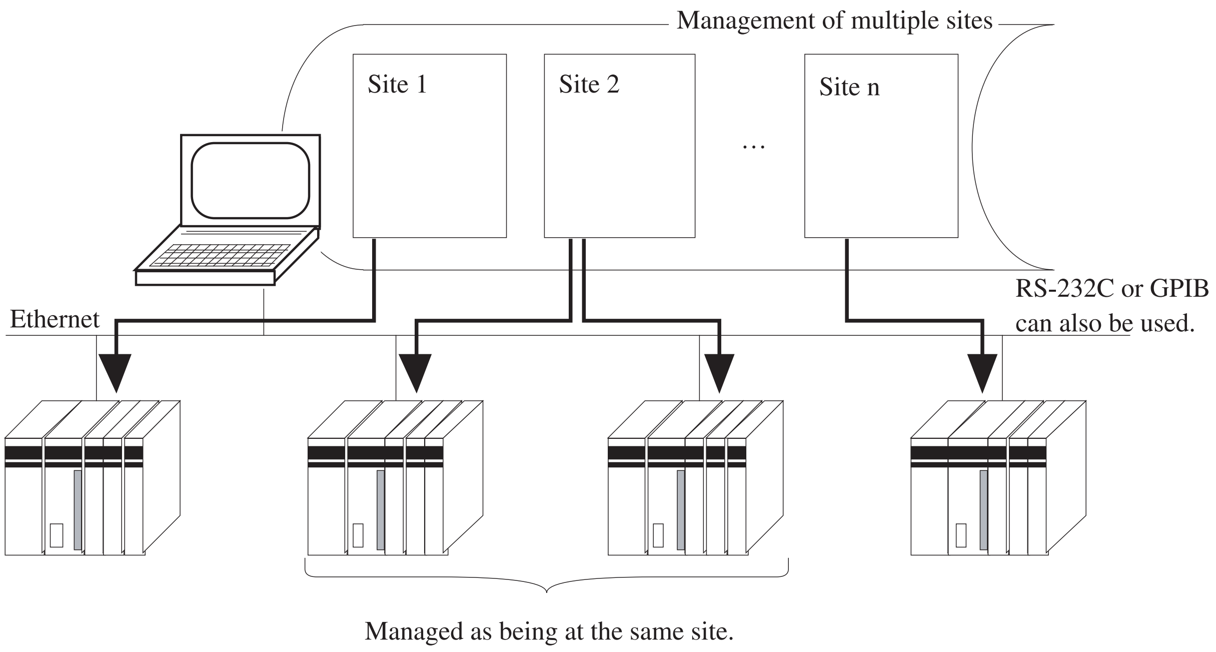
(4) When a program under MS-DOS issues a system call, a link to the MS-DOS kernel is made. In real-time processing, a link to CPMS is established by a system call. System calls supported by CPMS are called macro instructions. CPMS macro linkage libraries are also provided to enable programs to use macro instructions. As a result, programs can issue macro instructions in the same way as when using functions in C.

(5) An “sdebug” command loads programs under CPMS that were created under MS-DOS onto an actual machine on which they will actually run. The loader (sload) and builders (sctask, sdtask, sbuild, and sdbuild) are not responsible for loading onto the actual machine.

1.2 Sites

RPDP/S10 manages programs in sites. One personal computer can manage multiple sites. Programs managed at one site can be downloaded to one or more PCs (programmable controllers).

However, multiple PCs cannot be accessed at the same time. In addition, one PC cannot be handled as being at multiple sites.


Management of multiple sites
Site 1
Site 2
…
Site n
Ethernet
RS-232C or GPIB can also be used.
Managed as being at the same site.

A site can be changed to another by the “ssi” command, environment variable RSSITE, or a command with the -u option specified.

To change a site, use:	Purpose
ssi command	Setting of defaults for each personal computer
Environment variable RSSITE	Setting or modification for each MS-DOS prompt
Command with the -u option specified	Temporary modification for each command

The user should not store information on one site in multiple personal computers and access one PC from these personal computers. Also, he or she should not make multiple MS-DOS prompts active to perform multiple operations for the same site.

1.3 Crossing C Compiler

RPDP/S10 assumes that the user uses the MCP68K C Compiler Package.

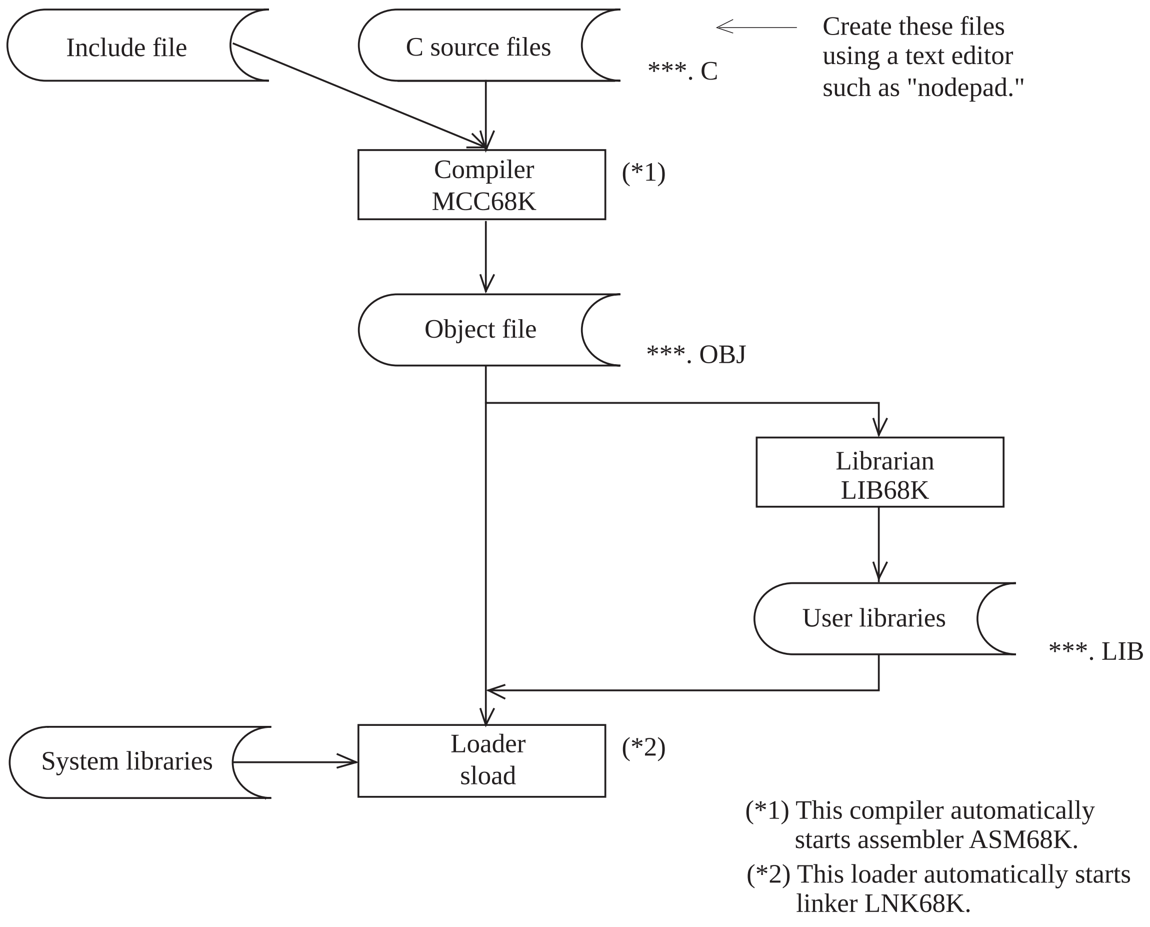
The MCP68K package includes the following commands:

- MCC68K compiler
- ASM68K assembler
- LNK68K linker
- LIB68K object module librarian

For details, refer to the manual supplied with the MCP68K package.

RPDP cannot be used with the XRAY debugger.

It is used with its own “sdebug” debugger.


Include file
C source files
***. C
Create these files using a text editor such as "nodepad."
Compiler MCC68K
(*1)
Object file
***. OBJ
Librarian LIB68K
User libraries
***. LIB
System libraries
Loader sload
(*2)
(*1) This compiler automatically starts assembler ASM68K.
(*2) This loader automatically starts linker LNK68K.

<Data flow>

THIS PAGE INTENTIONALLY LEFT BLANK.

2 PROGRAM DEVELOPMENT PROCEDURES

2.1 Overall Flow

Figure 2-1 shows an overall flow of program development for the PCs.

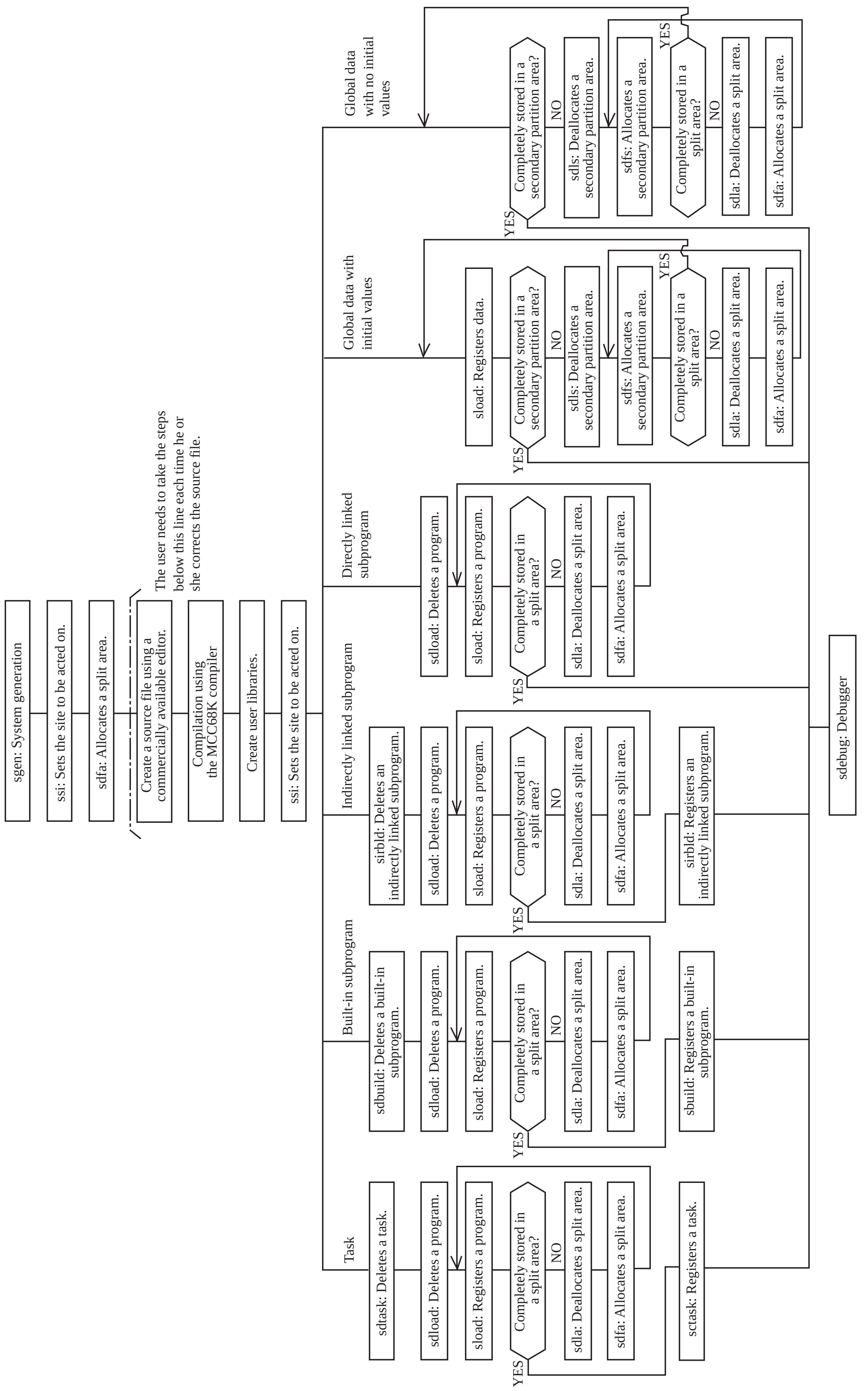

sgen: System generation
ssi: Sets the site to be acted on.
sdfa: Allocates a split area.
Create a source file using a commercially available editor.
Compilation using the MCC68K compiler
Create user libraries.
ssi: Sets the site to be acted on.
The user needs to take the steps below this line each time he or she corrects the source file.
Task
sdtask: Deletes a task.
sdload: Deletes a program.
sload: Registers a program.
Completely stored in a split area?
YES
NO
sdla: Deallocates a split area.
sdfa: Allocates a split area.
sctask: Registers a task.
Built-in subprogram
sdbuild: Deletes a built-in subprogram.
sbuild: Registers a built-in subprogram.
Indirectly linked subprogram
sirbld: Deletes an indirectly linked subprogram.
sirbld: Registers an indirectly linked subprogram.
Directly linked subprogram
Global data with initial values
sload: Registers data.
Completely stored in a secondary partition area?
sdls: Deallocates a secondary partition area.
sdfs: Allocates a secondary partition area.
Global data with no initial values
sdebug: Debugger

Figure 2-1 Overall Flow of Program Development for the PCs

2.2 Dividing Memory

Memory is divided into smaller areas for two purposes: one is to place programs and data in divided areas of main memory efficiently without duplication, and the other is to manage what is installed as common memory on the system bus.

During system generation by “sgen,” the main memory is defined as global areas each called a “garea” and having a particular use. Common memory on the system bus is also defined as global areas each intended for use with a particular type of memory card. In addition, a “garea” is divided into split areas each called an “area.” When an “area” is part of a global area (GLB), it is further divided into secondary partition areas each called an “sarea”). When a program stores data, an “area” or “sarea” name is used to indicate the location where the data is to be stored. The allocator defines divided areas in a hierarchy. The loaders and builders run according to the definition.

<Examples of Divided Areas and Their Names>

garea	os	sub		glbr		task		glbrw		ems		
area	Not managed by RPDP	a3	a4	a5		a1	a2	a6		a10		a11
sarea				s1	s2			s3	s4	s10	s11	

Unprotected

garea: Each “garea” is defined by the user at the time of system generation using “sgen.” “sgen” divides the main memory and extension memory into “gareas” for the system, subprograms (sub), read-only global data (glbr), tasks, and read/write global data (glbrw). If an extension memory is provided for added sites, a “garea” named “ems” must be defined. Do not specify its usage as “garea.” The use of “ems” is not declared as of a “garea,” but is declared when executing “sdfa.”
The “garea” named “ems” is an unprotected area.

area: Each “area” is allocated by “sdfa” and deallocated by “sdla.” More than one “area” can be allocated in one “garea.” In one “area,” more than one task, subprogram, or “sarea” may be located. When deallocating an “area,” make sure that all tasks or subprograms in it have been deleted.

sarea: Each “sarea” is allocated by “dfs” and deallocated by “sdls.” Define one “sarea” for each GLB.

2.3 Loading Programs and Creating Tasks

According to the management information determined by an allocator, the loader (sload) loads programs and data generated as load modules into “areas” or “sareas.” “sload” fetches information on CPMS resources such as a GLB from the area management information and sets the fetched information in a load module to create an executable module. The created executable module is stored in a backup file in the auxiliary storage device of the personal computer.

An executable module, when loaded as a single program, is registered as a task by a builder (sctask). The “sctask” builder sets the attributes of the task in the task control block (TCB) managed by CPMS.

2.4 Resident Subprograms

A task involves many subprograms. Of these, subprograms incorporated in the main body of the task are called internal subprograms (ISUBs). By contrast, subprograms always in main memory separately from the task are called resident subprograms (RSUBs). RSUBs can be shared by other tasks.

There are two types of RSUBs: directly linked RSUBs and indirectly linked RSUBs (IRSUBs). Directly linked RSUBs are placed at predetermined addresses in main memory. Even if the coding of a directly linked RSUB is altered, it must be placed at the same address as before the alteration. However, the altered RSUB may be too large to fit into that area. IRSUBs are used to avoid such a problem. Management tables to be linked to the calling program are set up for IRSUBs. In the presence of these tables, IRSUBs can be altered with ease by only updating the linked management table.

The “sload” loader updates directly linked RSUBs and indirectly linked RSUBs. The “sirbld” builder updates management tables for indirectly linked RSUBs.

2.5 Values

The user can register as external names constants that are shared by programs. These registered constants are called values (VALs). VALs are registered by “sdfv” and deleted by “sdlv.” When a load module is saved in the backup file, the loader sets all necessary VALs in that backup file. For this reason, any VALs must be registered before tasks and subprograms that reference them are loaded.

2.6 Programming Guide for GLBs, VALs, and RSUBs

The preceding sections discuss what GLBs, VALs, and RSUBs actually are. This section explains how to code and link these items for use in programs.

(1) Naming rules for GLBs and VALs

① Maximum length

8 characters (excluding attribute characters)

② Characters
Letters of the alphabet, digits, and _ (underscore). Each name must start with a letter and end with either of the following attribute characters:
GLB: _g
VAL: _v

③ Uniqueness
No name may be defined multiply.

(2) How to use GLBs and VALs
Table 2-1 shows how to use GLBs and VALs.

Table 2-1 How to Use GLBs and VALs

No.	Usage	Coding in C
1	Declaration of a GLB (on the referencing side)	extern long name_g [size] Explanation: Specify the above statement. name: Global name size: Size of the global area (in units of four bytes)

2	Referencing of a GLB	main () {
		long i;
		i=a_g [0];
		}
		Explanation:
		a: Global name
		Add a declaration as shown in item No. 1 above before the first line of this coding.
		(Example) extern long a_g [25]

3	Declaration of a GLB (on the referenced side)	Nothing need be declared. Set the initial value as shown in item No. 4 below.
4	Setting of initial values in the GLB	long a_g [25]= $\underbrace{\{0,\ 0,\ \cdots,\ 0\}}_{\text{Enter the initial values here.}};$ Explanation: a: Global name

5	Referencing of a VAL value	extern long vl_v
		long y=(long)&vl_v;
		main () {
		    long x;
		    x=y;
		}
		Explanation:
		vl: VAL name

(3) Notes on referencing GLB data

When referencing GLB data during creation of a program, do not define any initial values of that GLB data in the same program. To assure this, create a program that references the GLB data, keeping the following points in mind:

(a) Note on declaration of a GLB

In GLB declaration, the user can declare the size in bytes of each name, as indicated in item No. 1 within Table 2-1. The compiler and assembler do not perform validity check between the declared size and the actual size of the area allocated by the "sdfs" command. Therefore, an error is not detected even when the program references an address outside the actual area.

Example: Reference to an address outside the declared area

```
<Allocator>
sdfs usrresp0/glb2 100
<c>
extern long glb2_g [100];  ⎫
          ⋮                 ⎬ No error is detected.
glb2_g [100]= ······;       ⎭
```

(b) Referencing a relative address

A GLB can be referenced in the form of "name ± α", where α is a relative byte address in the range of -2^{31} to $2^{31} - 1$.

(c) Omitted declaration

If no GLB data is declared, then compilation continues as if a GLB data item whose size is 0 is declared.

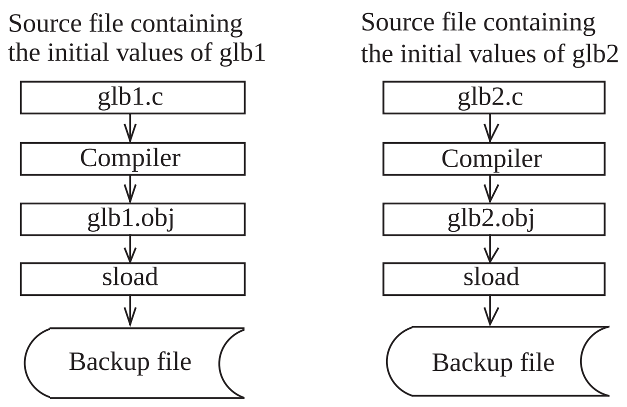
(d) Notes on handing "sload"

The user should create a file containing only the initial values of GLB data and work with the GLB data separately. The user may not define multiple GLB subareas with initial values in one single source program.

For each GLB subarea, create a source file containing initial GLB data. Compile and load the source file for only one GLB subarea at a time. Do not store initial values for two or more GLB subareas in one single source file.

A source file in which the GLB initial values are stored should not contain any data for other GLB subareas.

Do not store both the initial values for a GLB subarea and a program or subprogram in one single source file.


Source file containing the initial values of glb1
glb1.c
Compiler
glb1.obj
sload
Backup file
Source file containing the initial values of glb2
glb2.c
Compiler
glb2.obj
sload
Backup file

(4) Notes on creation of RSUBs
For details, see item (3) in Section 2.7.

(5) How to use RSUBs
The "sload" loader assumes that all undefined names of object modules created by the compiler, except GLB and VAL names, are RSUB names. Be sure to load RSUBs before the load modules using them.

Table 2-2 How to Use RSUBs

Usage	Coding in C

Referencing of an RSUB	`void rsub1();`
	`double rsub2();`
	`main() {`
	`long rusub3();`
	`        ⋮`
	`   rsub1(&i);`
	`   i=rsub3(i);`
	`   y=rsub2(x[i]);`
	`        ⋮`
	`}`
	Explanation:
	The names rsub1, rsub2, and rsub3 declared as function types can be used as RSUB names.

2.7 Restrictions on Program Creation under CPMS

The following restrictions apply to the creation of programs that run under CPMS:

(1) No support for overlay structures
CPMS does not support an overlay structure for tasks or resident subprograms. When creating two or more tasks or resident subprograms, take care so that they do not become too large.

(2) No support for bulk subroutines
CPMS does not support bulk subroutines that are placed in auxiliary memory and transferred to main memory when they are to be run. The only subprograms, including subroutines, that can be used under CPMS are resident subprograms (RSUBs) or internal subprograms (ISUBs) embedded in tasks. (Note that no auxiliary memory can be used with the S10/2α series.)

(3) Notes on creation of resident subprograms (RSUBs)
RSUBs reside in main memory and shared by multiple main programs. Therefore, RSUBs occupy areas of main memory independently of main programs that use them. Since RSUBs are used by multiple main programs at the same time, they need to be reentrant.
Only reentrant programs are used as RSUBs. A reentrant program is a program that can be called again by another program before it finishes its current run.
The following paragraphs explain how to create RSUBs properly.
An RSUB consists of two parts: a fixed part, which consists of a procedure section (text section) and a data section, and a variable part, which consists of work areas. The fixed part is shared by multiple main programs. The variable part is placed in the variable portion of each individual main program. RSUBs use the variable portion of main programs. Therefore, an RSUB needs to be programmed in such a way that the variable area used by the RSUB references the stack area. If an RSUB has a work area with no initial values (bss section), and it is programmed so that the stack area is to be read, then the RSUB cannot be shared by multiple main programs.
When creating an RSUB, note the following two points:
① All work areas must be secured in the stack area.
② Do not change any of the initial values of defined static variables.
③ If an RSUB consists of multiple routines, do not use any area to be shared by those routines.
Whether the restrictions in items ① and ② above are followed can be seen by checking that the length of the bss section in the map output by “sload” is 0.

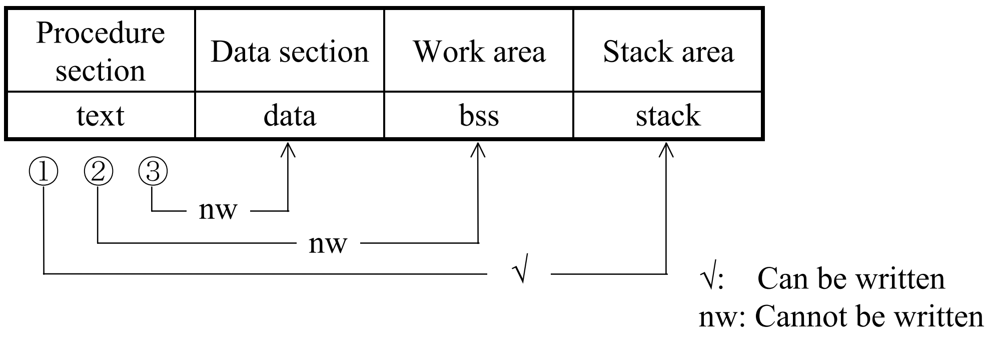


Figure 2-2 Writing to Individual Areas

① Writing to the stack area. The task can write to the stack area.
② Writing to the work area. Usually, RSUBs do not secure a work area nor do they write to the area. The task can write to the work area.
③ Writing to the data section. The task must not write to the data section.

Below are notes to be followed during the creation of RSUBs in any particular programming language.
Sample programming in C:

```
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
          ⋮
}
```

A program that writes to b1 declared at ① is non-reentrant.
A program that writes to d1 declared at ② is non-reentrant.
A program that writes to b2 declared at ③ is non-reentrant.
A program that writes to d2 declared at ④ is non-reentrant.
A program that writes to b3 declared at ⑤ is non-reentrant.
A program that writes to d3 declared at ⑥ is non-reentrant.

If programs write to s1 and s2 declared at ⑦ and ⑧, respectively, they remain reentrant. In an RSUB, the only variables that may be used are those like the variables at ⑦ and ⑧.
What variables are placed in what areas is described below.
Usually, b1 is placed in the bss area. (*)
b2 is placed in the bss area.
b3 is placed in the bss area.
d1 is placed in the data area.
d2 is placed in the data area.
d3 is placed in the data area.
s1 is placed in the stack area.
s2 is placed in the stack area.
(*) When an initial value is assigned to b1 in another program, b1 is placed in the data area.

(4) No relocatability
Programs and subprograms have no relocatability. Thus, those programs and subprograms whose run areas are already determined cannot be run in any other area. If you want to run such programs and subprograms in other areas, delete them and then register them again.

(5) Up to eight characters for names
The names of programs or subprograms may be up to eight characters long. GLB and VAL names may also be up to eight characters long. When specifying GLBs or VALs in C, suffix their names with "_g" or "_v", bringing the total length to up to 10 characters.

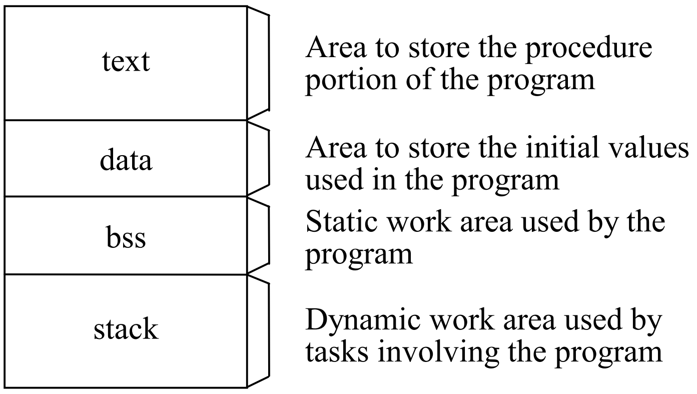
(6) GLB and VAL names
If names terminated with "_g" or "_v" are declared as external names, they are handled as GLB or VAL names. Assign names not terminated with "_g" or "_v" to programs that do not use GLBs or VALs. Names terminated with "_b" are reserved for future extension and should not be used.

(7) Unique external names
External names may not duplicate any other GLB names, program names, subprogram names, or VAL names already in use in the system.

(8) Unusable names
Because of certain restrictions on program creation, the use of some symbols as names is prohibited, and the use of some other symbols and particular statements is permitted only under limited conditions. For details, see Appendix B.

(9) Program structure

Programs under CPMS have a structure as shown below.


text
data
bss
stack
Area to store the procedure portion of the program
Area to store the initial values used in the program
Static work area used by the program
Dynamic work area used by tasks involving the program

The sizes of these areas are corrected so that they are multiples of four bytes. Also, these areas are secured in such a way that the first address of each area is a multiple of 4.

(10) Restriction on the first address

GLB areas may be re-allocated by the allocator so that their default first addresses are multiples of 4.

(11) Handling initial values

As shown below, initial values are handled differently, depending on whether MS-DOS or CPMS is used.

Area	CPMS	MS-DOS
data	Programmed value	Programmed value
bss	Unpredictable	0
stack	Unpredictable	Unpredictable

3 OUTLINE OF COMMANDS

3.1 Commands

Classification	Command	Function			Page
Compilation	MCC68K	C compiler			26
Generation	sgen	System generation			30
	ssi	Sets and displays the site to be acted on.			35
Allocation	sdfa	Allocates a split area.			39
	sdla	Deallocates a split area.			40
	sdfs	Allocates a secondary partition area (sarea).			41
	sdls	Deallocates a secondary partition area (sarea).			42
	sdfv	Defines a VAL.			43
	sdlv	Deletes a VAL.			44
Loading	sload	Stores a program, a subprogram, or data.			49
	sdload	Deletes a program or subprogram.			53
	scomp	Compares the load module and backup of a program, a subprogram, or data.			54
Building	sctask	Creates a task.			58
	sdtask	Deletes a task.			59
	sbuild	Creates a built-in subroutine.			60
	sdbuild	Deletes a built-in subroutine.			60
	sirbld	Creates or deletes an indirectly linked subprogram or table.			61
Debugging	sdebug	Online debugging			64
		Task start/stop	qu	Requests the start of a task.	66
			ab	Inhibits a task from being started.	66
			re	Releases a task from the state in which its start is inhibited.	66
			ta	Displays the status of a task.	67
			tm	Activates the cyclic start of a task.	68
			ct	Deactivates the cyclic start of a task.	68
		Memory display/ modification	md	Displays or changes memory content between addresses.	69
			sd	Displays or changes memory content between symbols.	70
		Breakpoint	br	Sets and displays breakpoints.	72
			rb	Removes breakpoints.	73
			rd	Displays the contents of registers.	74
			rr	Changes the contents of registers.	74
			go	Resumes execution from a breakpoint.	75
		System error display/clearing	el	Displays system errors.	75
			er	Clears system errors.	80
			ss	Displays the system status.	81

Commands (continued from previous page)

Classification	Command	Function			Page
Debugger	sdebug (continued from previous page)	Current time setting/display	st	Sets the current time.	81
			gt	Displays the current time.	81
		Uploading/ downloading	ld	Transfers a backup file to memory in the S10/2α.	82
			sv	Transfers memory content in the S10/2α to its corresponding backup file.	83
			cm	Compares a backup file with its corresponding memory content in the S10/2α.	84
		Others	dr	Enables DHP recording.	85
			ds	Disables DHP recording.	85
			ver	Displays version information.	85
			smd	Displays or modifies the contents of all areas of memory.	85
			help	Displays a command menu.	86
			q	Terminates the debugger.	86
			!	Executes an MS-DOS command.	86
	sdhp	Displays CPMS trace information.			87
	srpl	Loads programs.			88
Management tool	smap	Displays map information.			90
	sirmap	Displays indirectly linked map information.			99
	sadm	Displays the name corresponding to an address.			100

3.2 Environment Variables

RPDP commands use the following environment variables:

① RSSDIR=C:\hitachi\alc

This environment variable indicates the directory in which to store site information. By default, site information created by RPDP is placed in the 'C:\hitachi\alc' directory.

② RSSITE=*sitename*

This environment variable specifies a site name. When changing the site for each MS-DOS prompt program, redefine this environment variable. Usually, this environment variable is left undefined, in which case the site specified by the ssi command is used.

③ RSUTYP=*mode*

This environment variable specifies a user type, which is the access privilege level or processing mode that becomes valid when the -S option is omitted from a particular command. (Ordinary users should not use the -S option.)
When "mode" is "u", the user type specified is "user."
When "mode" is "s", it is "system." (Ordinary users should not use this setting.)
Usually, this environment variable is left undefined, in which case the command begins processing, assuming that "u" is given.

④ FX_LIB_DIR=C:\hitachi\fodu\lib

This environment variable indicates the directory in which to store libraries for use by CPMS or IRSUBs.

⑤ FODUDIR=C:\hitachi\fodu

This environment variable indicates the directory in which to store RPDP/S10-related files. When a site name is given to the "ssi" command, it is stored in file %FODUDIR%\MS_DOS\site.

⑥ MRI_68K_BIN=C:\MRI\MCC68K;C:\MRI\ASM68K

⑦ MRI_68K_LIB=C:\MRI\MCC68K\68000
(MRI_68K_LIB=C:\MRI\MCC68K\68020 for the S10/2αE, 2αH, and 2αHf)

⑧ MRI_68K_INC=C:\MRI\MCC68K

⑨ MRI_68K_TMP=C:\MRI\MCC68K\TMP

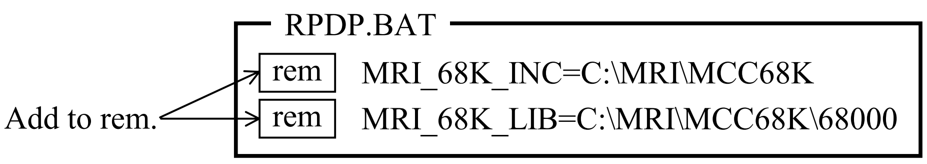
⑩ DOS16M=1$_\triangle$@1m-2m

⑪ RPDPS_10=68000
(RPDPS_10=68020 for the S10/2αE, 2αH, and 2αHf)

The environment variables numbered ① and ④ through ⑪ are defined as a result of execution of the 'RPDP.BAT' file.
(For the S10/2α, its environment is set up in the "RPDP.BAT" file.
For the S10/2αE, 2αH, and 2αHf, their environments are set up in the "RPDPE.BAT" file.)
If the installation directory of MCC68K is changed, modify or redefine the "RPDP.BAT" or "RPDPE.BAT" file. Note that reinstallation initializes the "RPDP.BAT" and "RPDPE.BAT" files.

Standard installation stores the ‘RPDP.BAT’ and ‘RPDPE.BAT’ files in the ‘C:\HITACHI\S10\C\BIN’ directory.
An environment that suits to the connections made with the PCs needs to be established, as described in Appendixes H and I.

<Precautions on using the Crossing C compiler (MCC68K) manufactured by Mentor Graphics Company>
The following environment variables are automatically set when installing the Crossing C compiler (MCC68K).
① MRI_68K_INC=C:\MGC\embeded\include\mcc68k
② MRI_68K_LIB=C:\MGC\embeded\lib
③ LM_LICENSE=C:\MGC\embeded\licence\license.dat
Because ① and ② are changed into other values (values of ⑦ and ⑧) when executing the PRDP.BAT or RPDPE.BAT, change the corresponding rows into comment statements as shown below before executing RPDP.BAT or RPDPE.BAT.



```
RPDP.BAT
rem MRI_68K_INC=C:\MRI\MCC68K
rem MRI_68K_LIB=C:\MRI\MCC68K\68000
```

3.3 Installation Procedure

The installation procedure is given below. Note that ‘RPDPE.BAT’ and ‘RPDP.BAT’ are overwritten during reinstallation. Thus, if the ‘RPDPE.BAT’ and ‘RPDP.BAT’ files are used by adding changes to them, and those changes are not reflected in their backup copies, then the user should add those changes to then after reinstallation using the backup copies.

(1) Insert the first RPDP floppy disk into the drive.

(2) Select [Add/Remove Programs] from [Control Panel] or from that of [Windows Explorer] to start ‘setup.exe’ in the floppy disk drive.

(3) Remove the floppy disk from the drive and insert another one, as instructed by the message presented.

(4) Add the string ‘C:\HITACHI\S10\C\BIN’ to a PATH in the ‘AUTOEXEC.BAT’ file. If the installation directory has been changed, specify the new installation directory.
Example: PATH=C\···········;C\HITACHI\S10\C\BIN

(5) Start the personal computer.
Be sure to load the operating system and debugger system program to the PCs before using the RPDP/S10.

- For the S10/2α
  To load the operating system to the PCs, use the CPMS loader system.
  To load the debugger system program to the PCs, use the CPMS debugger system.

- For the S10/2αE, S10/2αH, and S10/2αHf
  To load the operating system to the PCs, use the CPMSE loader system.
  To load the debugger system program to the PCs, use the CPMSE debugger system.

4 COMPILER

4.1 Required Option

To compile S10/2α programs, be sure to use MCC68K together with the following option:

```
>MCC68K -c source file name
```

Example:

```
>MCC68K -c PROG.C
```

Option

-c	Creates an object file but does not link it.

4.2 Outline of Options

This section explains the MCC68K options related to program development on the S10/2α.
For details of these options, refer to the "MCC68K Users Guide."

(1) Related options

-A	ANSI-compliant mode (default)
-nA	Non-ANSI mode (ANSI extension functions are not used.)
-f	Creates codes that use instructions output by a floating-point coprocessor. (If the PC has no floating-point coprocessor, do not use this option.)

(2) Unusable options

-G*	Debugging-related option
-g	Debugging-related option
-h	Creates codes for an HP6400 series development system.
-N*	Section-related option
-u*	Symbol name-related option
-X*	External name-related option

There are many other options that affect program execution. Understand their functions before using them.

5 LIBRARIES

5.1 Libraries

A library is a set of object modules. By collecting multiple objects in a library and passing the library to “sload,” all necessary object modules in the library are automatically linked.

5.2 Librarian

The LIB68K librarian edits a library containing object modules generated by MCC68K.
The following paragraphs briefly explain how to use the librarian. For details, refer to the description about LIB68K in the ASM68K User’s Guide.

● Usage (command-line interface)

(1) Adding a module
```
>LIB68K-a    object file name    library file name
```
• If the specified library is not found, the librarian creates a new library under the specified name and presents a warning message.

(2) Replacing a module
```
>LIB68K-r    object file name    library file name
```
• If the specified library is not found, the librarian creates a new library under the specified name and presents a warning message.
• If the specified module is not found in the library, the librarian adds the module to the library and presents a warning message.

(3) Fetching a module
```
>LIB68K-e    object module name    library file name
```
• An object file is created without the module being deleted from the library.

(4) Deleting a module
```
>LIB68K-d    object module name    library file name
```

(5) Listing the contents of a library
```
>LIB68K-l    library file name
```
The default extension of object file names is “.OBJ”, and the default extension of library file names is “.LIB”.

LIB68K may display warning messages when the user uses some commands supported by RPDP/S10. This is because those commands use LIB68K commands. However, that presents no problem.

6 GENERATOR

6.1 sgen (System Generation)

The “sgen” command sets up a file environment with necessary information for the controller to be acted on. The user should define the environment variables described in Section 3.2 and Appendixes H and I before initiating this command.

(1) Operation

```
>sgen
+++ site generation +++

 site name (1-8 chars)                            :{site}
 site type (S10/2A, S10/2AE, S10/2AH)             :{type}
 total memory size (in K-byte)                    :{size}
 C programming area top addr                      :{addr}
Garea definition start
 task area size (in K-byte)                       :{tsize}
 sub program area size (in K-byte)                :{ssize}
 ir sub program max number                        :{irsmax}
 read only global data area size (in K-byte)      :{grsize}
 read/write global data area size (in K-byte)     :{grwsize}
 ir global data max number                        :{irgmax}

 site name                                        =site

 site type                                        =type
 total memory size (K-byte)                       =size
 C programming area top addr                      =addr
Garea information
 task area size (K-byte)                          =tsize
 sub program area size (K-byte)                   =ssize
 ir sub program max number                        =irsmax
 read only global data area size (K-byte)         =grsize
 read/write global data area size (K-byte)        =grwsize
 ir global data max number                        :irgmax
site information ok? (y/n) : {ans}

site directory initial start
site directory initial end

+++ site generation end +++
```

(2) Options

sgen [△-c△site1△site2]
[△-d△site△]

-c	Copies all site information at a time.
-d	Deletes all site information at a time.
site	Site name (site1, name of the site from which to copy; site2, name of the site to which to copy)

(3) Operands

site	Site name
type	Controller type (*1)
size	Total size of main memory size>tsize+ssize+grsize+grwsize
addr	First address of the C program area (default: 0x160000)
tsize	Size of the “garea” for tasks
ssize	Size of the “garea” for subprograms (including the management tables for “irsub” and “irglb” programs) (*2) ssize × 1024 ≥ irsmax × 6+irgmax × 4+8
grsize	Size of the read-only global “garea”
grwsize	Size of the read/write global “garea”
irsmax	Maximum “irsub” number
irgmax	Maximum “irglobal” number
ans	If the displayed data matches the data the user entered, enter a “y.” Site information will then be created. If not, enter an “n” to enter the data again.

(*1) Controller types

S10/2A: HITACHI S10/2α
S10/2AE: HITACHI S10/2αE
S10/2AH(f): HITACHI S10/2αH

(*2) When an “irsub” or “irglobal” is registered, an area for a management table is automatically allocated. Therefore, the size of the area actually allocated by the “sdfa” command is the specified size less the size of the management table.

(4) Generation result

① New creation
- The directory specified by the environment variable RSSDIR is created.
- A site information definition file is created in the site directory.
- A site backup file is created. (The backup file is cleared to 0.)

② Copying all site information at a time
All files in the site directory are copied to the specified site at a time.

③ Deleting all site information at a time
All files in the site directory are deleted at a time.

Note: Except for the “garea” size definition, the contents of the site information definition file (‘sysdef’) after site generation may be modified with a text editor.

Example: New creation

```
>sgen
+++ site generation +++

 site name (1-8 chars)                             :pcs01
 site type (S10/2A, S10/2AE, S10/2AH)              :S10/2A
 total memory size (in K-byte)                     :4096
 C programming area top addr                       :0x160000
Garea definition start
 task area size (in K-byte)                        :1024
 sub program area size (in K-byte)                 :1024
 ir sub program max number                         :1024
 read only global data area size (in K-byte)       :512
 read/write global data area size (in K-byte)      :1024
 ir global data max number                         :1024

 site name                                         =pcs01

 site type                                         =S10/2A
 total memory size (K-byte)                        =4046
 C programming area top addr                       =0x160000
Garea information
 task area size (K-byte)                           =1024
 sub program area size (K-byte)                    =1024
 ir sub program max number                         =1024
 read only grobal data area size (K-byte)          =512
 read/write global data area size (K-byte)         =1024
 ir global data max number                         =1024
site information ok? (y/n) : y

site directory initial start
site directory initial end
```

<<LIB68K displays a warning message at this time. However, that presents no problem.>>

```
+++ site generation end +++
```

Example: Copying all site information at a time

```
>sgen△-c△pcs01△pcs02
site (pcs01) ---> site (pcs02) copied
```

Example: Deleting all site information at a time

```
>sgen△-d△pcs02
site (pcs02) delete ok? (y/n) : y
site (pcs02) deleted
```

Example: To extend a site

```
>sgen
+++ site generation +++

 site name (1-8 chars)                              :pcs01
 site type                                          =S10/2A
 total memory size (in K-byte)                      =4096
 C programming area top addr                        =0x160000
Garea information
 task area size (K-byte)                            =1024
 sub program area size (K-byte)                     =1024
 ir sub program max number                          =1024
 read only global data area size (K-byte)           =512
 read/write global data area size (K-byte)          =1024
 ir global data max number                          =1024

 add extended memory size (in K-byte)               :4096

 site name                                          =pcs01
 site type                                          =S10/2A
 total memory size (K-byte)                         =8192
 C programming area top addr                        =0x160000
Garea information
 task area size (K-byte)                            =1024
 sub program area size (K-byte)                     =1024
 ir sub program max number                          =1024
 read only global data area size (K-byte)           =512
 read/write global data area size (K-byte)          =1024
 ir global data max number                          =1024
 extended area size (K-byte)                        =4608
site informatin ok? (y/n) : y

site directory updata start
site directory updata end
```

6.2 ssi (Sets and displays the site to be acted on.)

Operation

ssi [△siten] .... Setting and display

[No parameter] .... Display

siten site name

- Specify a site name for each personal computer. When changing the site name for a particular MS-DOS prompt, set a new site name in the environment variable RSSITE.
- The site name set by this command is valid until a new site name is set by another “ssi” command.

Example: Display mode

```
>ssi
  site name                                        =pcs01
  site type                                        =S10/2A
  total memory size (K-byte)                       =4096
  C programming area top addr                      =0x160000
Garea information
  task area size (K-byte)                          =1024
  sub program area size (K-byte)                   =1024
  ir sub program max number                        =1024
  read only global data area size (K-byte)         =512
  read/write global data area size (K-byte)        =1024
  ir global max number                             =1024
```

Example: When the specified site could not be found

```
>ssi△pcs02
site (pcs02) not found!!
```

6 GENERATOR

6.1 sgen (System Generation)

The “sgen” command sets up a file environment with necessary information for the controller to be acted on. The user should define the environment variables described in Section 3.2 and Appendixes H and I before initiating this command.

(1) Operation

```
>sgen
+++ site generation +++

 site name (1-8 chars)                              :{site}
 site type (S10/2A, S10/2AE, S10/2AH)               :{type}
 total memory size (in K-byte)                      :{size}
 C programming area top addr                        :{addr}
Garea definition start
 task area size (in K-byte)                         :{tsize}
 sub program area size (in K-byte)                  :{ssize}
 ir sub program max number                          :{irsmax}
 read only global data area size (in K-byte)        :{grsize}
 read/write global data area size (in K-byte)       :{grwsize}
 ir global data max number                          :{irgmax}

 site name                                          =site

 site type                                          =type
 total memory size (K-byte)                         =size
 C programming area top addr                        =addr
Garea information
 task area size (K-byte)                            =tsize
 sub program area size (K-byte)                     =ssize
 ir sub program max number                          =irsmax
 read only global data area size (K-byte)           =grsize
 read/write global data area size (K-byte)          =grwsize
 ir global data max number                          :irgmax
site information ok? (y/n) : {ans}

site directory initial start
site directory initial end

+++ site generation end +++
```

(2) Options

```
sgen [△-c△site1△site2]
     [△-d△site△]
```

-c	Copies all site information at a time.
-d	Deletes all site information at a time.
site	Site name (site1, name of the site from which to copy; site2, name of the site to which to copy)

(3) Operands

site	Site name
type	Controller type (*1)
size	Total size of main memory size>tsize+ssize+grsize+grwsize
addr	First address of the C program area (default: 0x160000)
tsize	Size of the “garea” for tasks
ssize	Size of the “garea” for subprograms (including the management tables for “irsub” and “irglb” programs) (*2) ssize×1024 ≥ irsmax×6+irgmax×4+8
grsize	Size of the read-only global “garea”
grwsize	Size of the read/write global “garea”
irsmax	Maximum “irsub” number
irgmax	Maximum “irglobal” number
ans	If the displayed data matches the data the user entered, enter a “y.” Site information will then be created. If not, enter an “n” to enter the data again.

(*1) Controller types

S10/2A: HITACHI S10/2α
S10/2AE: HITACHI S10/2αE
S10/2AH(f): HITACHI S10/2αH

(*2) When an “irsub” or “irglobal” is registered, an area for a management table is automatically allocated. Therefore, the size of the area actually allocated by the “sdfa” command is the specified size less the size of the management table.

(4) Generation result

① New creation

- The directory specified by the environment variable RSSDIR is created.
- A site information definition file is created in the site directory.
- A site backup file is created. (The backup file is cleared to 0.)

② Copying all site information at a time

All files in the site directory are copied to the specified site at a time.

③ Deleting all site information at a time
All files in the site directory are deleted at a time.

Note: Except for the “garea” size definition, the contents of the site information definition file (‘sysdef’) after site generation may be modified with a text editor.

Example: New creation

```
>sgen
+++ site generation +++

 site name (1-8 chars)                            :pcs01
 site type (S10/2A, S10/2AE, S10/2AH)             :S10/2A
 total memory size (in K-byte)                    :4096
 C programming area top addr                      :0x160000
Garea definition start
 task area size (in K-byte)                       :1024
 sub program area size (in K-byte)                :1024
 ir sub program max number                        :1024
 read only global data area size (in K-byte)      :512
 read/write global data area size (in K-byte)     :1024
 ir global data max number                        :1024

 site name                                        =pcs01

 site type                                        =S10/2A
 total memory size (K-byte)                       =4046
 C programming area top addr                      =0x160000
Garea information
 task area size (K-byte)                          =1024
 sub program area size (K-byte)                   =1024
 ir sub program max number                        =1024
 read only grobal data area size (K-byte)         =512
 read/write global data area size (K-byte)        =1024
 ir global data max number                        =1024
site information ok? (y/n) : y

site directory initial start
site directory initial end
```

<<LIB68K displays a warning message at this time. However, that presents no problem.>>

```
+++ site generation end +++
```

Example: Copying all site information at a time

```
>sgen△-c△pcs01△pcs02
site (pcs01) ---> site (pcs02) copied
```

Example: Deleting all site information at a time

```
>sgen△-d△pcs02
site (pcs02) delete ok? (y/n) : y
site (pcs02) deleted
```

Example: To extend a site

```
>sgen
+++ site generation +++

 site name (1-8 chars)                              :pcs01
 site type                                          =S10/2A
 total memory size (in K-byte)                      =4096
 C programming area top addr                        =0x160000
Garea information
 task area size (K-byte)                            =1024
 sub program area size (K-byte)                     =1024
 ir sub program max number                          =1024
 read only global data area size (K-byte)           =512
 read/write global data area size (K-byte)          =1024
 ir global data max number                          =1024

 add extended memory size (in K-byte)               :4096

 site name                                          =pcs01
 site type                                          =S10/2A
 total memory size (K-byte)                         =8192
 C programming area top addr                        =0x160000
Garea information
 task area size (K-byte)                            =1024
 sub program area size (K-byte)                     =1024
```

```
  ir sub program max number                            =1024
  read only global data area size (K-byte)             =512
  read/write global data area size (K-byte)            =1024
  ir global data max number                            =1024
  extended area size (K-byte)                          =4608
 site informatin ok? (y/n) : y

 site directory updata start
 site directory updata end
```

6.2 ssi (Sets and displays the site to be acted on.)

Operation

ssi [△siten] .... Setting and display

[No parameter] .... Display

siten site name

- Specify a site name for each personal computer. When changing the site name for a particular MS-DOS prompt, set a new site name in the environment variable RSSITE.
- The site name set by this command is valid until a new site name is set by another "ssi" command.

Example: Display mode

```
>ssi
  site name                                      =pcs01
  site type                                      =S10/2A
  total memory size (K-byte)                     =4096
  C programming area top addr                    =0x160000
Garea information
  task area size (K-byte)                        =1024
  sub program area size (K-byte)                 =1024
  ir sub program max number                      =1024
  read only global data area size (K-byte)       =512
  read/write global data area size (K-byte)      =1024
  ir global max number                           =1024
```

Example: When the specified site could not be found

```
>ssi△pcs02
site (pcs02) not found!!
```

THIS PAGE INTENTIONALLY LEFT BLANK.

7 ALLOCATOR

7.1 Command Language Specification

(1) Classification of names

Names handled by the allocator are classified as shown below.

- Names
  - Site names
  - Area names
    - Global area names
    - Split area names
    - Secondary partition area names
  - External names
    - Executable module names, program names (for programs and subprograms)
    - Value names
  - (Task names)

(2) Rules

(a) Site names, area names, external names, and task names

- The first character must be a letter from “a” to “z.”
- Only the letters (“a” to “z”), digits (“0” to “9”), and underscore (“_”) can be used.
- Each name must be up to eight characters long.

(b) Note

Underscores (“_”) and uppercase letters (“A” to “Z”) are specific to system mode. Do not use these characters in user mode. However, the allocator does not check characters used.

(3) Numeric data

The allocator commands support decimal and hexadecimal numbers in the following formats:

Decimal	127	(Decimal numbers start with a digit from “1” to “9.”)
Hexadecimal	0x7F	(Hexadecimal numbers start with the symbol “0x.”)

(4) Options

- -a$_{\triangle}$xyz format

  Always specify options in this format.
- -a format

  When specifying both -a and -b, do not specify them in the form of “-ab.”

(5) Spaces allocatable by the allocator

The allocator can support only spaces in main memory; that is, “sdfa” and “sdfs” can only allocate spaces in main memory.

7.2 sdfa (Allocates a split area.)

Function

This command allocates an area in a specified global area (garea).

Format

sdfa△gname/aname△size [△option]

Explanation

gname	Name of a global area
aname	Name of the area to be allocated
size	Size of the area to be allocated. This size must be a multiple of four bytes. When the specified number is not a multiple of 4, a warning message appears and that number is rounded up to the nearest whole multiple of 4 to continue processing.

Options

-p	Allocates an area to store a task.
-s (lowercase)	Allocates an area to store a subprogram.
-d	Allocates a global area with initial values.
-w	Allocates a global area with no initial values. If a read-only area (glbr) is specified as a global area, this option may not be used.
-S (uppercase)	Specifies that the access privilege level is “system.” If this option is omitted, the default privilege level set in advance is used. (This option is provided for system programs, and no ordinary users are allowed to use it.)
-u△site	Name of the site to be processed by the allocator. If this option is omitted, processing is performed on the default site set in advance.
-l△n	Address (location) of the area to be allocated. Use a multiple of 4 to specify a byte address relative to the beginning of the global area. If the specified number is not a multiple of 4, a warning message appears and that number is rounded up to the nearest whole multiple of 4 to continue processing. If this option is omitted, the first free area found is automatically allocated.

Notes

- The options -p, -s, -d, and -w are mutually exclusive in the command line.
- If none of -p, -s, -d, and -w is specified, -p is assumed.
- Table 7-1 shows the permitted combinations of allocated areas, arguments, and options.

Table 7-1 Permitted Combinations of Areas Allocated by "sdfa," Arguments, and Options

Parameter \ Area type		Task	Subprogram	Global: With initial values	Global: With no initial values
gname		Name of global area			
aname		Name of area to be allocated			
size		Number of bytes to be allocated (multiple of 4)			
type		-p (default)	-s	-d	-w
Options	-S (uppercase)	Specifiable when the access privilege level is "system." (If this option is omitted, the default privilege level is used.)			
	-u site	Site name. (If this option is omitted, the default site name is used.)			
	-l n (*)	Multiple of 4 that indicates a byte address relative to the beginning of the global area. (If this option is omitted, an area is automatically allocated.)			

(*) n: If the specified number is not a multiple of 4, it is rounded up to the nearest whole multiple of 4.

7.3 sdla (Deallocates a split area.)

Function

This command deallocates an area allocated by "sdfa."

Format

sdla△aname [△option]

Explanation

aname	Name of the area to be deallocated

Options

-S (uppercase)	Specifies that the access privilege level is "system." If this option is omitted, the default privilege level set in advance is used. (This option is provided for system programs, and no ordinary users are allowed to use it.)
-u△site	Name of the site to be processed by the allocator. If this option is omitted, processing is performed for the default site set in advance.

Notes

If a secondary partition area for a task or subprogram is allocated in the specified split area, an error will result. For a global area with or without initial values, the secondary partition areas in a specified split area therein are also deallocated.

7.4 sdfs (Allocates a secondary partition area [sarea].)

Function

This command allocates a global secondary partition area (sarea) in the area that has already been allocated by "sdfa."

Format

sdfs△aname/sname△size [△option]

Explanation

aname	Name of a split area
sname	Global name of the external area to be allocated
size	Size of the secondary partition area to be allocated (in bytes)

Options

-S (uppercase)	Specifies that the access privilege level is "system." If this option is omitted, the default privilege level set in advance is used. (This option is provided for system programs, and no ordinary users are allowed to use it.)
-u△site	Name of the site to be acted on by the allocator. If this option is omitted, processing is performed for the default site set in advance.
-l△n	Address (location) of the secondary partition area to be allocated. Use a multiple of 4 to specify a byte address relative to the beginning of the split area. If the specified number is not a multiple of 4, a warning message appears and that number is rounded up to the nearest whole multiple of 4 to continue processing. If this option is omitted, the first free area found is automatically allocated.
-a△n	Alignment value for use in allocating an secondary partition area. Specify the nth power of 2, where $0 \leq n \leq 12$. If this option is omitted, a 2 is assumed.

Notes

- The -a option is valid only when the -l option is omitted.
- The -a and -l options are mutually exclusive. When both are specified, an error will result.
- The alignment value specified by the -a option must be from 2 to 12. The number may be a 0 or 1 only for special purposes. Usually, do not specify a 0 or 1.
- Table 7-2 shows the permitted combinations of allocated areas, arguments, and options.
- LIB68K may display a warning message when this command is used. However, that presents no problem.

Table 7-2 Permitted Combinations of Areas Allocated by “sdfs,” Arguments, and Options

Parameter \ Area type		Task	Subprogram	Global	
				With initial values	With no initial values
aname		☒	☒	Name of split area	
sname		☒	☒	Name of secondary partition area	
size		☒	☒	Number of bytes	
Options	-S (uppercase)	☒	☒	Specifiable when the access privilege level is “system.” (If this option is omitted, the default privilege level is used.)	
	-u site	☒	☒	Site name. (If this option is omitted, the default site name is used.)	
	-l n	☒	☒	Byte address relative to the beginning of the split area. (If this option is omitted, an area is automatically allocated.)	
	-a n	☒	☒	Number of alignment boundaries ($0 \le n \le 12$). (If this parameter is omitted, a 2 is assumed.)	

Note: No values may be specified in the cases indicated by ☒. If a value is given, an error will result.

7.5 sdls (Deallocates a secondary partition area [sarea].)

Function

This command deallocates a secondary partition area (sarea) allocated by “sdfs.”

Format

sdls△sname [△option]

Explanation

aname	External name of the area to be deallocated

Options

-S (uppercase)	Specifies that the access privilege level is “system.” If this option is omitted, the default privilege level set in advance is used. (This option is provided for system programs, and no ordinary users are allowed to use it.)
-u△site	Name of the site to be acted on by the allocator. If this option is omitted, processing is performed on the default site set in advance.

7.6 sdfv (Defines a VAL.)

Function

This command registers external reference information for values.

Format

sdfv$_\triangle$ename$_\triangle$value [$_\triangle$option]

Explanation

ename	External name
value	Value to be assigned to the external name

Options

-S (uppercase)	Specifies that the access privilege level is “system.” If this option is omitted, the default privilege level set in advance is used. (This option is provided for system programs, and no ordinary users are allowed to use it.)
-u$_\triangle$site	Name of the site to be acted on by the allocator. If this option is omitted, processing is performed on the default site set in advance.

Notes

- A negative decimal number may be specified in the following format as the value to be assigned to an external name.
  -123
  Explanation: The decimal number -123 is specified.
- The value to be specified as “value” must be in the following range:
  $-2^{31} \leq \text{value} \leq 2^{31}-1$
- LIB68K may display a warning message when this command is used. However, that presents no problem.

7.7 sdlv (Deletes a VAL.)

Function

This command deletes external reference information registered by “sdfv.”

Format

sdlv$_\triangle$ename [$_\triangle$option]

Explanation

ename	External name

Options

-S (uppercase)	Specifies that the access privilege level is “system.” If this option is omitted, the default privilege level set in advance is used. (This option is provided for system programs, and no ordinary users are allowed to use it.)
-u$_\triangle$site	Name of the site to be acted on by the allocator. If this option is omitted, processing is performed on the default site set in advance.

8 LOADER

8.1 Execution Environment of the Loader

(1) Input to the loader
Make sure that the load modules to be input to the loader satisfy the conditions listed in Table 8-1.

Table 8-1 Input Conditions of Load Modules

Load module / Option	TEXT	DATA	BSS
Registration of programs	(*1) >0	–	–
Registration of subprograms and built-in subroutines	>0	–	–
Registration of data	–	(*1) (*2) >0	–

Symbols: TEXT, executable portion; DATA, data with initial values; BSS, area with no initial values.
(Legend) –: Processing is possible when the size is 0 or greater.
>0: An error will result when the size is not greater than 0.
(*1) An error will result when a value definition is encountered.
(*2) An error will result when no data with initial values is found in “glb.”

Any load module input to the loader has one of the following structures:

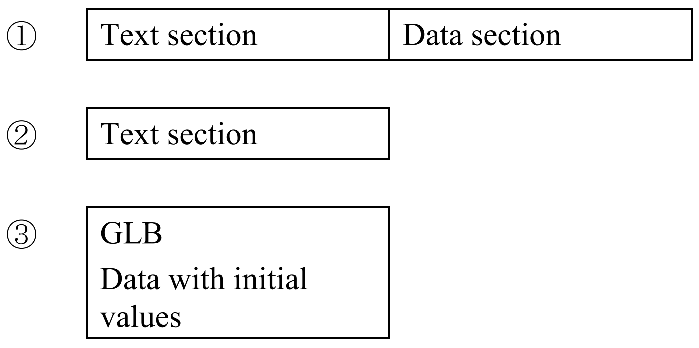


Figure 8-1 Load Module Structures

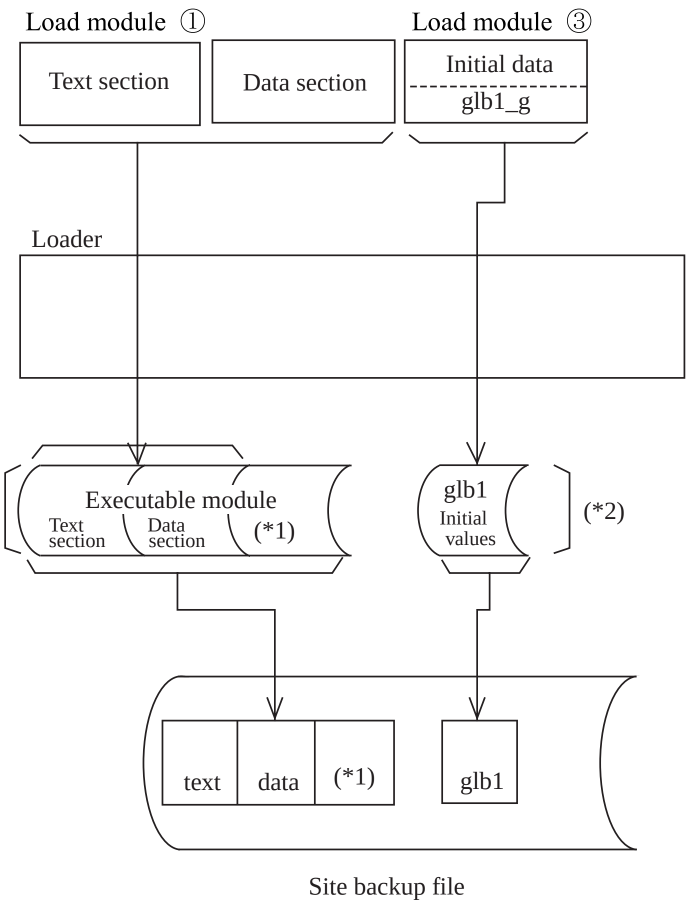
Explanation of Figure 8-1:
① Load module generated from a program or subprogram that has both a text section and data section.

② Load module generated from a program or subprogram that consists of only a text section. It is loaded in the same way as the load module described in ① above.

③ Load module generated from a program that contains the initial values of a GLB.　It is loaded as data.

(2)　Processing by the loader

Loading by the loader is explained below using the load module structures numbered ① and ③ in Figure 8-1 as examples.



(*1) bss area used during programming

(*2) This area actually does not exist as a file.　It is shown to help the user understand the processing.

Figure 8-2　Processing by the Loader

Explanation of Figure 8-2:

① The global initial data in the load module created by “sload” is loaded into the area corresponding to the secondary partition area registered by “sdfs,” in a split area managed by the allocator.

② As an executable module, the text section and data section are loaded into the area specified by a loader command.
③ As in ① and ②, this load module is loaded into the site backup file.

(3) Unique names
Make sure that no duplicate program name, subprogram name, built-in subroutine name, global name, or value name appears in the combined system and user domain.

(4) External reference check between the system and user
The system cannot reference user information.
The user can only reference subprograms of the system. Table 8-2 shows the permitted combinations of externally referencing items and externally referenced items.

Table 8-2 Permitted Combinations of Externally Referencing Items and Externally Referenced Items

Referenced item / Referencing item		Subprogram		Global		Value	
		S	U	S	U	S	U
Program	S	√	P	√	P	√	P
	U	√	√	P	√	P	√
Subprogram	S	√	P	√	P	√	P
	U	√	√	P	√	P	√
Global	S	√	P	√	P	√	P
	U	√	√	P	√	P	√

(Legend)	√ : Permitted	S : System
	P : Prohibited	U : User

(5) Reference check by program attribute and allocated area
The loader generates programs that are executable under CPMS. However, they may or may not be able to reference memory locations outside their addressing spaces, depending on their program attribute and allocated area. This is because there are no computer instructions available that enable access to such external locations. Therefore, a check is made to see if they are referencing only permitted memory locations, according to the decision criteria listed in Table 8-3.

Table 8-3 Permitted Combinations of Referencing Programs and Referenced Programs

Referenced side / Referencing side	+P	+S, +U	+D	VAL
+P	P	√	√	√
+S, +U	P	√	√	√
+D	P	√	√	√

(Legend)	√ : Permitted	P : Prohibited
	+P : Program	+S : Subprogram
	+D : Built-in subprogram	+D : Data loading

8.2 sload (Loads a program, a subprogram, or data.)

Function

This command stores a program or subprogram in a backup file under a specified name and, at the same time, creates a program management table in the executable module management file. The command also performs the same processing for data.

Format

sload△pname [△option]

Explanation

pname	Program or subprogram name to be registered in the program management table. When the +D option is specified for data loading, it needs to be accompanied by a GLB name. The name specified as "pname" should be a character string of up to eight characters and should begin with a letter.

Options

-S (uppercase)	Specifies that the operational mode is "system." If this option is omitted, the default operational mode set in advance is used. (This option is provided for system programs, and no ordinary users are allowed to use it.)
-u△site	Name of the site to be acted on by the loader. If this option is omitted, processing is performed on the default site set in advance.
-C△n (uppercase)	First address of the program or subprogram. n is a multiple of 4 specified as the first address. If the specified number is not a multiple of 4, a warning message appears and that number is rounded up to the nearest whole multiple of 4 to continue processing.

-p$_{\triangle}$n	Relative byte address indicating the loading start location in the area. This option is effective for a program or subprogram. If this option is omitted, registration is automatically performed. This option cannot specified together with -C. n is a multiple of 4 specified as the relative byte address. If the specified number is not a multiple of 4, a warning message appears and that number is rounded up to the nearest whole multiple of 4 to continue processing.
-a$_{\triangle}$aname	Area into which the program or subprogram is to be loaded. This option may not be omitted when a program or subprogram name is specified.
-f$_{\triangle}$cmdfile	(This option is used for ordinary purposes. It forces "sload" to automatically set a loading start address.) cmdfile specifies a command file containing the object file and library file to be linked.

Format of cmdfile

`load` Main object file path	Place the main object file path at the beginning.
`load` Subordinate object file path	Zero or more subordinate object file paths may be written.
`load` Subordinate library file path	Zero or more subordinate library file paths may be written.
`load C:\HITACHI\FODU\LIB\site name.LIB`	When using an IRSUB, insert this line.
`load C:\HITACHI\FODU\LIB\CPMS.LIB`	When using CPMS macros, insert this line.
`load C:\HITACHI\FODU\LIB\IRAD.LIB` ↵	When using an "irsubad" or "irglbad," insert this line.

↵: Return code

Add a return code (press the [Enter] key) after the last line input of cmdfile. When there is no return code in the last line, cmdfile cannot be recognized correctly but it may become an error at the time of loading.

-i$_{\triangle}$n [$_{\triangle}$m]	This option is used for special purposes. It loads the result of executing LNK68K with a user-specified load address.)
n	File name of the absolute load module (S code) file output by LNK68K
m	File name of the map file output by LNK68K. If the map file name is omitted, the program or subprogram is processed, assuming that it consists of only text. Also, the data is processed, assuming that it contains nothing other than the data values.

Note: The -f and -i options are mutually exclusive. If neither -f nor -i is specified, -i a.out is assumed.

-w△n	Stack area size in bytes. This option may not be omitted if a program or subprogram name is specified. The option specifies the size of the stack area used by the program or subprogram. n is a multiple of 4 within the range of 0 to 4,194,304 (0x400000). If the specified number is not a multiple of 4, a warning message appears and that number is rounded up to the nearest whole multiple of 4 to continue processing.
+P (uppercase)	Specifies that the thing to be loaded is a program.
+S (uppercase)	Specifies that the thing to be loaded is a subprogram.
+U (uppercase)	Specifies that the thing to be loaded is a built-in subroutine.
+D (uppercase)	Specifies that the thing to be loaded is global data. If none of the +P, +S, +U, and +D options is given, +P is assumed.
-m△n	Used when multiple tasks need to be created. n is the number of tasks to be created and is within the range of 2 to 128.

Permitted combinations of options

Table 8-4 shows the permitted combinations of options that may be specified to the loader.

Table 8-4 Permitted Combinations of Options

Option / Type	-S	-u	-C	-p	-a	-i	-w	-m	+P	+S	+U	+D
Program	△	△	△	△	√	△	√	△	△	–	–	–
Subprogram	△	△	△	△	√	△	√	M	–	√	–	–
Built-in subprogram	△	△	△	△	√	△	√	M	–	–	√	–
Data	△	△	M	M	M	△	M	M	–	–	–	√

(Legend) △: Optional √: Required M: May not be used –: Irrelevant

Notes

- Those which are registered as subprograms cannot be registered as built-in subroutines. When you want to use them as built-in subroutines, register them in advance using the +U option (built-in subroutine).
- The stack area for built-in subroutines is allocated in the area reserved for the system. Make sure that the stack area is not greater than 1 KB.
- LIB68K may display a warning message when this command is used. However, that presents no problem.

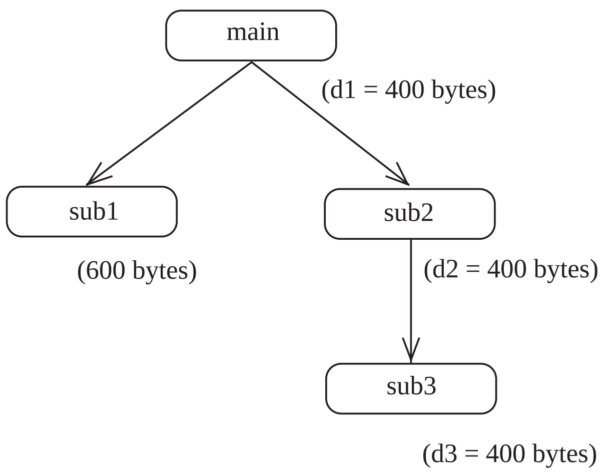
Calculating the stack size

When programs use the stack area, specify its size (D) as follows:

(1) Calculating D

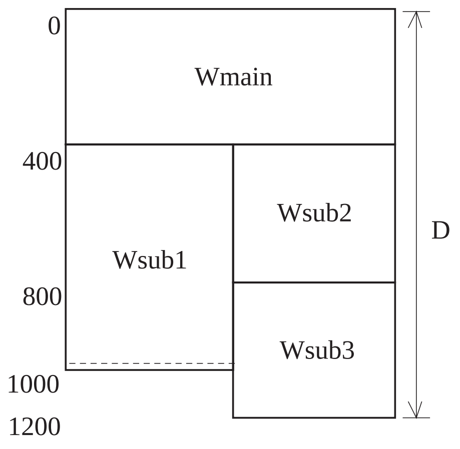
The value of D is the maximum value that can be obtained by adding the stack sizes specific to the internal subprograms constituting a program according to the parent- child relationships. In the example below, D is 1200 bytes.

(a) Parent-child relationships of programs



Each value in parentheses is the stack size specific to an internal subprogram.

(b) Layout of stack areas used by programs



W is the size of the stack specific to the associated internal subprogram.

In the above example, D = Wmain + Wsbu2 + Wsub3

= 400 + 400 + 400 = 1200

As in this example, when the size of the stack used by each subprogram is known from, for instance, the information output by the compiler, the total size of the individual stack areas can be calculated with ease. If, however, the stack sizes for the individual programs are unknown, obtain them from the source program, as described below.

(2) Calculating the stack size, "di", in bytes for each program (main program or subprogram)

① When the program or subprogram does not have the following:

function call (subroutine call)

di=56+J

② When the program or subprogram has the following:

function call (subroutine call)

di=maxarg+64+J

(Note) J: Auto variable area size

maxarg: Maximum argument value × 4

If the value of J is not determined accurately, T can be used instead of J because of the relationship shown below.

J ≤ T= (number of auto variables) × 4 + (number of variables for double-precision real-type data) × 8

Auto variables include the "struct" and "register" variables.

Example 1: When there is no “function call” (example in C)

```
func (i)
int  i;
{return;}
```

In this case

$$56 + \underbrace{1 \times 4}_{\text{Number of auto variables} \times 4} = 60$$

Example 2: When there is “function call”

```
main () {
int     i1, i2, i3, i4, i5, i6;
f1 (i1, i2) ;
f2 (i1, i2, i3, i4, i5) ;
f3 (i1, i2, i3)
}
```

In this case

$$\underbrace{5 \times 4}_{\text{maxarg}} + 64 + \underbrace{6 \times 4}_{\text{Number of auto variables} \times 4} = 108$$

8.3 sdload (Deletes a program or subprogram.)

Function

This command deletes a program or subprogram registered by an “sload” command from the external name and program/subprogram management file. The backup file is not cleared to 0.

Format

sdload$_{\triangle}$pname [$_{\triangle}$option]

Explanation

pname	Name of the program or subprogram to be deleted. “pname” is a string of up to eight characters, starting with a letter. Specifiable characters are alphanumeric characters and underscores (“_”).

Options

-S (uppercase)	Specifies that the operational mode is “system.” If this option is omitted, the default operational mode set in advance is assumed. (This option is provided for system programs, and no ordinary users are allowed to use it.)
-u$_{\triangle}$site	Name of the site to be acted on by the loader. If this option is omitted, processing is performed on the default site set in advance.
+P (uppercase)	Deletes a program.
+S (uppercase)	Deletes a subprogram.
+U (uppercase)	Deletes a built-in subroutine. If none of the +P, +S, and +U options is specified, +P is assumed.

8.4 scomp (Compares a program, a subprogram, or data.)

Function

This command compares the contents of the backup file of a program, subprogram, or global data with its load module, and edits and outputs the result.

Format

scomp△pname [△option]

Explanation

pname	Name of the program or subprogram to be compared. When global data is to be compared, "pname" is ignored and the global names used in the program are subjected to processing.

Options

-f△cmdfile -i△n [m]	} Same as "sload"
-u△site	Name of the site to be acted on by the loader. If this option is omitted, processing is performed on the default site set in advance.
-S (uppercase)	Specifies that the operational mode is "system." If this option is omitted, the default operational mode set in advance is assumed. (This option is provided for system programs, and no ordinary users are allowed to use it.)
+P (uppercase)	Compares a program.
+S (uppercase)	Compares a subprogram.
+D (uppercase)	Compares global data.
+U (uppercase)	Compares a built-in subroutine. If none of the +P, +S, +U, and +D options is given, +P is assumed.

Messages output by the "scomp" command are explained below.

● Message format on normal termination

```
** comp list **
user name=XXXXXXXX  mode=XXXX  program type=XXXXX
program name=XXXXXXXX
** compare end **
```

● Message format on abnormal termination

```
** comp list **
user name=XXXXXXXX  mode=XXXX  program type=XXXXX
program name=XXXXXXXX
scomp:text size unmatched (No=0095)
```
→ Indicates that they differs from each other in text size.

```
  scomp:data size unmatched (No=0096) →
```
Indicates that they differ from each other in data size.

```
** compare error **
<header>
loc=”XXXXXXXX”  new=”XXXXXXXX”  old=”XXXXXXXX”
<text>
loc=”XXXXXXXX”  new=”XXXXXXXX”  old=”XXXXXXXX”
<data>
loc=”XXXXXXXX”  new=”XXXXXXXX”  old=”XXXXXXXX”
** compare error end **
```

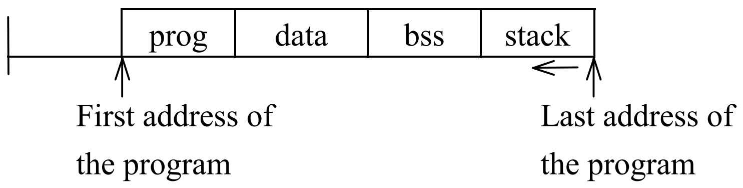
- Explanation

user name	Site name	
mode	Operational mode	(sys, user)
program type	Program attribute	(pgm, sub, ulsub, data)
program name	Program name	
loc	If any discrepancy is found in the comparison, the number of bytes starting from the beginning of the header, text, or data section is displayed here to indicate the location of the discrepancy. For global data, the number of bytes starting from its beginning is displayed.	
new	Data of the “a.out” file (load module)	
old	Data of the program registered in the backup file (executable module)	

8.5 Program Layout

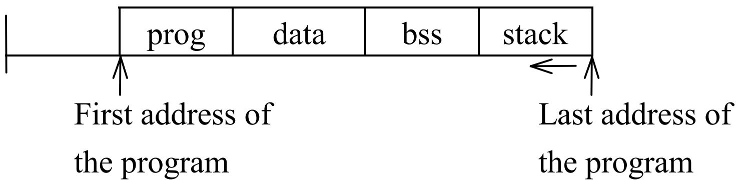
This section describes how a program is loaded into and arranged in the CPMS system.

(1) Program containing subprograms



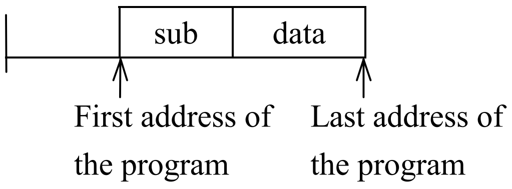
prog	Executable program
data	Data with initial values to be referenced by the executable program
bss	Area with no initial values to be referenced by the executable program
stack	Stack area used by the executable program. This stack area includes stack areas used by subprograms. This field indicates how stacks are used.

(2) Program containing no subprograms



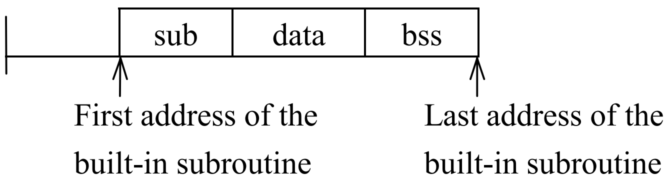
prog, data, bss	Same as for a program containing subprograms
stack	Stack area used by the executable program

(3) Subprogram



sub	Subprogram
data	Data with initial values to be referenced by the subprogram
Note:	Use only reentrant routines as subprograms. When creating subprograms, perform programming in such a way that the "bss" area is not used.

(4) Built-in subroutine


sub
data
bss
First address of the built-in subroutine
Last address of the built-in subroutine

sub	Built-in subroutine
data	Data with initial values to be referenced by the built-in subroutine
bss	Data with no initial values to be referenced by the built-in subroutine
Note:	The system stack is used.

9 BUILDER

9.1 sctask (Creates a task.)

Function

This command creates a task from the executable module stored by the loader.

Format

sctask△pname△tname△-t△n [△option]

Explanation

pname	Program name of the executable module to be used as a resource to create a task
tname	Name of the task to be created
-t△n	Task number. A user task is identified with its task number from 1 to the maximum user task number 114, and a system task with its task number from 1 to 128. If a task number in use is specified, an error will result. The task number 128 is reserved for the debugger task, and 127 for FA-BASIC. The task numbers 115 to 128 are reserved for system tasks. The user cannot use these task numbers.

Options

-u△site	Name of the site to be acted on by the builder. If this option is omitted, processing is performed on the default site set in advance.
-v△n	Execution level of 1 to 4 at the initial start of the task. If this option is omitted, n is assumed to be 4.
-r△n	Number of 1 to 128 for use in creating a work section when multiple tasks are to be created from the program. This number may not be greater than the value of the -m option, which specifies the number of tasks to be created from the program specified by a parameter of the load command. If this option is omitted, the minimum work section creation number not in use is assumed.
-S (uppercase)	Specifies that a system task is to be created. If this option is omitted, the default task type set in advance is assumed. (This option is provided for system programs, and no ordinary users are allowed to use it.)

Notes

- Even when the type of the executable module is “u”, a system task can be created by specifying the -S option.
- Table 9-1 lists the defaults of the options.
- Table 9-2 shows the relationships between types of task created and options used for the purpose.

Table 9-1 Defaults of Options

Option	Default	Remarks
-u	Default name	Set in advance
-v	4	
-S	Default	Preset value of the environment variable RSUTYP
-r	Minimum “rmtn” not in use	

Table 9-2 Relationships between Types of Task Created and Options Used

Option / Task type	pname	tname	-u△site If this option is omitted, the default site is assumed.	-v△n If this option is omitted, n is 4 for “user” or 0 for “system.”	-S If this option is omitted, the default is assumed.	-t△n	-r△n If this option is omitted, the minimum “rmtn” not in use is assumed. (*)
Single task	◎	◎	√	√	√	◎	M
Multiple tasks	◎	◎	√	√	√	◎	√

◎: Required √: Optional M: May not be used

(*) rmtn: Work section creation number

9.2 sdtask (Deletes a task.)

Function

This command deletes an already-created task.

Format

sdtask△tname [△option]

Explanation

tname	Name of the task to be deleted

Options

-u△site	Name of the site to be acted on by the builder. If this option is omitted, processing is performed on the default site set in advance.
-S (uppercase)	Specifies that a system task is to be deleted. If this option is omitted, the default task type set in advance is assumed.

9.3 sbuild (Creates a built-in subroutine.)

Function

This command creates a system-specific subprogram (built-in subroutine) that performs processing in the event of an error.

Format

sbuild△subname△-p△n [△option]

Explanation

subname	Name of the built-in subroutine to be created
-p△n	Place where to include the built-in subroutine:

Place of inclusion		n
SDS	System Down Subroutine	2
CPES	CPU Error Subroutine	3
EXS	Exit Subroutine	4
ABS	Abort Subroutine	5
PCKS	Parameter Check Subroutine	8

Option

-u△site	Name of the site to be acted on by the builder. If this option is omitted, processing is performed on the default site set in advance.

9.4 sdbuild (Deletes a built-in subroutine.)

Function

This command deletes an existing built-in subroutine.

Format

sdbuild△subname△-p△n [△option]

Explanation

subname	Name of the built-in subroutine to be deleted
-p△n	Place where the built-in subroutine is included. For the value of n, see the description of "sbuild" above.

Option

-u△site	Name of the site to be acted on by the builder. If this option is omitted, processing is performed on the default site set in advance.

9.5 sirbld (Creates or deletes an indirectly linked subprogram or table.)

Function

This command creates an indirectly linked subprogram or global data or deletes it for maintenance purposes. The command also saves the definition information stored in the current address table to a map information file.

Format

sirbld△irno△name [△option]

Explanation

irno	Registration number of the indirectly linked subprogram or indirectly linked global data (in decimal)
name	Name of the indirectly linked subprogram or indirectly linked global data. Up to eight characters

Options

-g	Specifies that indirectly linked global data is to be created or deleted.
-S (uppercase)	Specifies that an indirectly linked subprogram is to be created or deleted. Either -g or -S must be specified.
-u△site	Name of the site to be acted on by the builder. If this option is omitted, processing is performed on the default site set in advance.
-s△name	External name handled by the allocator. This option takes effect only when the external name handled by the allocator is not specified by a parameter.
-o△n	Offset in hexadecimal or decimal. This option takes effect only when an address is given in the form of "external name + offset." If the specified value is preceded with the symbol "0x", it is handled as a hexadecimal number; otherwise, it is handled as a decimal number.
-a△n	Absolute address in hexadecimal. This option takes effect only when an absolute value is given as the address.
-d	Specifies deletion.

Notes

- If the user requests this command to register or delete an indirectly linked subprogram, the command registers or deletes the corresponding linkage subprogram (site name.lib) created by RPDP. The -s and -a options are mutually exclusive on the command line.
- For registration, specify both the -s and -o options.
- LIB68K may display a warning message when this command is used. However, that presents no problem.

10 sdebug (ONLINE DEBUGGER)

10.1 Starting the Debugger

Start “sdebug” as follows:

Format

sdebug [△option]

++ debugger start --> site (site) ++

*

Options

-i△fname	Name of the file to which to output key inputs.
-o△fname	Name of the file to which to output operation results.
-r△fname	Name of the command file, which may be the file created by the -i option.
-s△command	Directly executes a debugger command.
-u△site	Name of the site to be acted on by the debugger. If this option is omitted, processing is performed on the default site set in advance.
-initial	Enables a number of C programs to be loaded at a time by the “ld” command.
-debug	Specifies debug mode, in which the “smd” command may be used.

Result

Upon normal termination, this command returns a 0. Upon abnormal termination, it returns a 1. If one of the commands listed in the table below is issued with the -s option specified and results in an error, this command returns a 255.

Notes

- When more than one option is specified, any option(s) that follow -s are regarded as commands.
  sdebug△-i△fname△-s△command.................-i is regarded as an option.
  sdebug△-s△command d△-i△fname..............-i is regarded as part of the command following the -s option.
- When an asterisk (“*”) is displayed, the debugger is ready to accept any of the commands listed in the table below.
- Be careful when starting “sdhp” or “sadm” with the -o option specified in this command. If the file name specified by the -o option in this command matches the file name specified by the -o option in “sdhp” or “sadm,” the result displayed on the screen will be stored in the file improperly.
- Do not specify the -o option in “sdhp” or “sadm” when using that command and the -o option of this command together.
- None of the breakpoint-related commands “br,” “rb,” “rr” (those that change the content of a register), and “go” can be used together with the -s option of this command.

Online debugger commands

Classification	Command	Function	Remarks
Task start/stop	qu	Requests a task be started.	
	ab	Inhibits a task from being started.	
	re	Releases a task from the state in which its start is inhibited.	
	ta	Displays the status of a task.	
	tm	Activates the cyclic start of a task.	
	ct	Deactivates the cyclic start of a task.	
Memory display/ modification	md	Displays or changes memory content between addresses.	Dynamic display is supported.
	sd	Displays or changes memory content between symbols.	Dynamic display is supported.
Breakpoint-related	br	Sets and displays breakpoints.	This command cannot be used together with the -s option.
	rb	Removes breakpoints.	This command cannot be used together with the -s option.
	rd	Displays the contents of a register(s).	
	rr	Changes the contents of a register(s).	This command cannot be used together with the -s option.
	go	Resumes execution from a breakpoint.	This command cannot be used together with the -s option.
System error display/clearing	el	Displays system errors.	
	er	Clears system errors.	
	ss	Displays the system status.	
Current time setting/display	st	Sets the current time.	This command can be used only where an extension memory with a clock is installed.
	gt	Displays the current time.	This command can be used only where an extension memory with a clock is installed.
Uploading/ downloading	ld	Transfers a backup file to memory in the S10/2α.	
	sv	Transfers memory content in the S10/2α to its corresponding backup file.	
	cm	Compares the backup file with the contents of memory in the S10/2α.	
Enabling or disabling DHP recording	dr	Enables DHP recording.	
	ds	Disables DHP recording.	
Others	ver	Displays version information for CPMS.	
	smd	Displays or modifies the contents of all areas in memory.	
	sadm	Displays the “sarea” name associated with an address.	Stand-alone start is supported.
	sdhp	Displays DHP.	Stand-alone start is supported.
	help	Displays the command menu.	
	q	Terminates the debugger.	
	!	Executes an MS-DOS command.	

10.2 Debugger Commands

(1) qu (Starts a task.)

Function

This command starts a specified task.

Format

*qu△tn [,fact]

*qu△tname [,fact]

Explanation

tn	Task number (1 to 128)
fact	Start factor (0 to 16). If this option is omitted, a 0 is assumed.
tname	Task name

Result

OK (0)	Normal termination
NG (≠0)	Parameter error or macro error (for example, the task to be started is not in idle state.)

(2) ab (Inhibits a task from being started.)

Function

This command inhibits a specified task or tasks from being started.

Format

*ab△tn1 [-tn2]

*ab△tname

Explanation

tn1	Task number (1 to 128)
tn2	Last task number (1 to 128)
tname	Task name

Result

OK (0)	Normal termination
NG (≠0)	Parameter error or macro error. If, however, tn1-tn2 is given, this command always terminates normally.

(3) re (Releases a task from its start-inhibited state.)

Function

This command releases a specified task or tasks from their start-inhibited state.

Format

*re△tn1 [-tn2]

*re△tname

Explanation

tn1	Task number (1 to 128)
tn2	Final task number (1 to 128)
tname	Task name

Result

OK (0)	Normal termination
NG (≠0)	Parameter error or macro error. If, however, tn1-tn2 is given, this command always terminates normally.

(4) ta (Displays the status of a task.)

Function

This command displays the status of a specified task.

Format

*ta$_{\triangle}$tn1

*ta$_{\triangle}$tname

Explanation

tn1	Task number (1 to 128)
tname	Task name

Result

tn=*** (0x**) tname=******** task state=***...* (0x********)

tcb top=0x***...*

task top=0x***...* stack=0x***...* level=**

tn	Task number
tname	Task name
task state	Task status. (The values of the status bits are presented in hexadecimal.) dormant, idle, ready, timer wait, break stop, running

MSB 0 15 LSB

R	Q	A	C	W	B	0	....	0

R: The task is being executed.

Q: The task is waiting for execution.

A: The starting of the task is inhibited.

C: The task is waiting for processing by the CPU.

W: The task is waiting for a time to expire.

B: The task is stopped at a breakpoint.

tcb top	First address of the TCB
task top	First address of the task

stack	First address of the task stack pointer
level	Initial start level of the task

(5) tm (Activates the cyclic start of a task.)

Function

This command activates the cyclic start process for a specified task.

Format

*tm△tn, cyct [,fact]

*tm△tname, cyct [,fact]

Explanation

tn	Task number (1 to 128)
tname	Task name
cyct	Start interval in ms (1 to 86400000)
fact	Start factor (1 to 16) If this option is omitted, a 0 is assumed.

Result

OK (0)	Normal termination
NG (1)	The timer table is full.

(6) ct (Deactivates the cyclic start of a task.)

Function

This command deactivates the cyclic start process for a specified task.

Format

*ct△tn [,fact]

*ct△tname [,fact]

Explanation

tn	Task number (1 to 128)
tname	Task name
fact	Start factor to be canceled (0 to 16) If this option is omitted, a 0 is assumed.

Result

OK (0)	Normal termination
NG (1)	A timer is not yet registered.

(7) md (Displays or changes memory content between addresses.)

Function

This command displays or modifies the contents of memory specified by addresses.

Format

```
*md
*1 strage (s,m,*) :  {s}
                     {m}
                     {*}
                     {e}
                     {nothing}
*2 addr :  {addr1[{-addr2}]}   {-h}  {[-1]}
           {addr1[{,len}]}     {-d}  {[-w]}
                                     {[-b]}
                               {-f}
           {e}
0xaaaaaaaa-0xdddddddd :  {[0x]data}
                         {nothing}
                         {e}
```

Explanation

1 strage (s,m,)

s	Specifies that the backup file be modified or displayed.
m,{nothing}	Specifies that memory in the actual machine be modified or displayed.
*	Specifies that both the backup file and memory in the actual machine be modified or displayed.
e	Terminates this command.

*2 addr

addr1-addr2	Specifies that data between first address addr1 and final address addr2 be displayed.
addr1,len	Specifies that data starting from first address addr1 be displayed by the number of bytes specified by “len.”
-h	Specifies that data be output in hexadecimal.
-d	Specifies that data be output in decimal.
-f	Specifies that data be output in single-precision floating-point format.
-l	Specifies that the data length be four bytes.
-w	Specifies that the data length be two bytes.
-b	Specifies that the data length be one byte.
e	Terminates this command.

0xaaaaaaaa 0xdddddddd

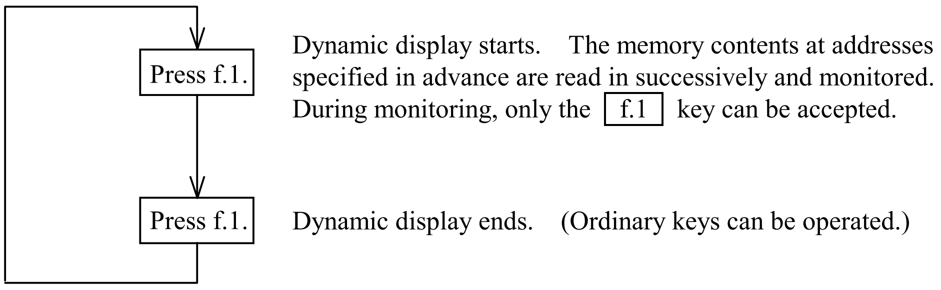
[0x]data	New data. When it is preceded with the symbol “0x”, it is handled as a hexadecimal number.
{nothing}	Specifies that none of the data should be changed.
e	Specifies that control be returned to “*2addr” for address input.

Note

If both a data output format and data length are omitted, those specified by the last “md” command take effect. By default, -h (hexadecimal) and -l (four bytes) are assumed. No data in memory in the actual machine can be changed in units of one byte.

● Dynamic memory display

Dynamic memory display is enabled by the following operation.


Press f.1.
Dynamic display starts. The memory contents at addresses specified in advance are read in successively and monitored. During monitoring, only the f.1 key can be accepted.
Press f.1.
Dynamic display ends. (Ordinary keys can be operated.)

(8) sd (Displays or changes memory content between symbols.)

Function

This command displays or modifies the contents of memory specified by a symbol(s) such as a program name.

Format

```
*sd
*1 name:name  [-t]
              [-s]
              [-g]
*2 strage (s,m,*):  {s}
                    {m}
                    {*}
                    {*n}
                    {e}
                    {nothing}
*3 baddr:    {addr [{-h}[{-1}]]   ]}
             {     [{-d}[{-w}]]  }}
             {     [      [{-b}]]  }}
```

```
                {          [{-f}                ]}
                {*n}
                {e}
*4 raddr:       {addr1   [{-addr2}  ]}
                {          [{,len} ]}
                {          [{,*         }]}
                {          [{-*         }]}
                {*}
                {*n}
                {e}
0xaaaaaaaa (0x11111111) 0xdddddddd: {[0x]data}
                                    {nothing}
                                    {*n}
                                    {e}
```

Explanation

*1 name

name	Name of the area to be modified or displayed
-t	Specifies that the name is a program name.
-s	Specifies that the name is a subprogram name.
-g	Specifies that the name is a global name.

Note: If none of -t, -s, and -g is given, -g is assumed.

2 strage (s,m,)

s	Specifies that the backup file be modified or displayed.
m,{nothing}	Specifies that memory in the actual machine be modified or displayed.
*	Specifies that both the backup file and memory in the actual machine be modified or displayed.
*n	Prompt number to return control to previous processing (only n=1 may be specified).
e	Terminates this command.

*3 baddr

addr	Address relative to the beginning of the area to be acted on.
*n	Prompt number to return control to previous processing (n must be 1 or 2).
-h	Specifies that data be output in hexadecimal.
-d	Specifies that data be output in decimal.
-f	Specifies that data be output in single-precision floating-point format.
-l	Specifies that the data length be four bytes.
-w	Specifies that the data length be two bytes.

-b	Specifies that the data length be one byte.
e	Terminates this command.

*4 raddr

addr1-addr2	Specifies that data between the first address addr1 and the final address addr2 is displayed. (These addresses are relative to “addr” of “baddr.”)
addr1, len	Specifies that as many data bytes as specified by “len” be displayed, starting from the address addr1. (This address is relative to “addr” of “baddr.”)
addr1,*	Specifies that the data in the area indicated by a specified symbol be displayed, starting from the address addr1 and continuing up to the end of the area. (This address is relative to “addr” of “baddr.”)
addr1-*	Specifies that the data in the area indicated by a specified symbol be displayed, starting from the address addr1 and continuing up to the end of the area. (This address is relative to “addr” of “baddr.”)
*	Specifies that all data in the area be displayed.
*n	Prompt number to return control to previous processing (n must be 1, 2, or 3).
e	Terminates this command.

0xaaaaaaaa (0x11111111) 0xdddddddd

[0x]data	New data. If it is preceded with the symbol “0x”, it is handled as hexadecimal data.
{nothing}	Specifies that none of the data be modified.
e	Specifies that control be returned to *4raddr relative-address input.

Note

If both a data output format and data length are omitted, those specified by the last “md” command take effect. By default, -h (hexadecimal) and -l (four bytes) are assumed.

This command also supports dynamic display.

For information on how to start dynamic display, see item (7) above.

No data in memory in the actual machine can be modified in units of one byte.

(9) br (Sets and displays breakpoints.)

Function

This command sets breakpoints or displays those currently set.

Format

*br [$_\triangle$pname$_\triangle$break1$_\triangle$......$_\triangle$break5]

Explanation

pname	Name of the program in which to set breakpoints.
break1 to break5	Breakpoints (relative addresses in the program).

Result

When breakpoints are set correctly, the following message appears:

```
break reset
name=program name      radder=relative address in the program
object=machine language instruction code
```

If neither “pname” nor “break” is given, all breakpoints currently set are displayed as shown below.

```
    break point
    name=program name      radder=relative address in the program
    object=machine language instruction code
     *
```

Note

Up to five breakpoints can be set for each S10/2α. When a set breakpoint is reached, the following message appears:

```
    break!!
    tn=task number      name=program name      radder=relative address in the program
```

If a command such as “rb,” “rd,” “rr,” or “go” fails, issue “br” without parameters to check the status of the breakpoints. If a mismatch is found in the information on breakpoints between the personal computer and S10/2α, change the information in the personal computer so that it matches that in the S10/2α.

(10) rb (Removes breakpoints.)

Function

This command deletes the breakpoints currently set.

Format

*rb [△pname△break1△......△break5]

Explanation

pname	Name of the program from which to delete breakpoints.
break1 to break5	Breakpoints (relative addresses in the program).

Result

If neither “pname” nor “break” is given, all breakpoints currently set are deleted. When they are deleted correctly, the following message appears:

```
    break reset
    name=program name      radder=relative address in the program
    object=machine language instruction code
```

(11) rd (Displays the contents of registers.)

Function

This function displays the contents of registers that have been existent since a breakpoint was reached.

Format

*rd

Result

OK (0)	Upon normal termination, the contents of registers are displayed as shown below.
NG (1)	No breakpoint interruption is in progress.

```
pc=0x********  sr=0x****
d0=0x********  d1=0x********  d2=0x********  d3=0x********
d4=0x********  d5=0x********  d6=0x********  d7=0x********
a0=0x********  a1=0x********  a2=0x********  a3=0x********
a4=0x********  a5=0x********  a6=0x********  a7=0x********
```

(12) rr (Changes the contents of registers.)

Function

This command changes the contents of registers while a breakpoint interruption is in progress.

Format

```
*rr
register name [d0-d7]    :data register
              [a0-a6]    :address register
              [pc]       :program counter
              [sr]       :status register
*rx
data:datax
```

Explanation

rx	Register abbreviation (d0 to d7, a0 to a6, pc, or sr)
datax	New data

Result

OK (0)	Normal termination
NG (1)	No breakpoint interruption was in progress. Or, an invalid register abbreviation was given.
NG (3)	A breakpoint interruption from another terminal was in progress.

Notes

This command takes effect only when a task is halted at a breakpoint.

The high-order five bits of the status register cannot be changed. Any attempt to change these bits is ignored.

(13) go (Resumes execution from a breakpoint.)

Function

This command resumes a task from the address of a breakpoint at which the task has been halted.

Format

*go

Result

OK (0)	Normal termination
NG (1)	The task in which a breakpoint interruption was in progress was at a stop. Or, the breakpoints were already deleted.
NG (2)	No breakpoint interruption was in progress.
NG (3)	A breakpoint interruption from another terminal was in progress.

Notes

This command takes effect only when the task is halted at a breakpoint.
If the result is NG(1), issue a "br" command with no parameters to display the breakpoints currently set.

(14) el (Displays system errors.)

Function

This command displays the error log in memory in the S10/2α.

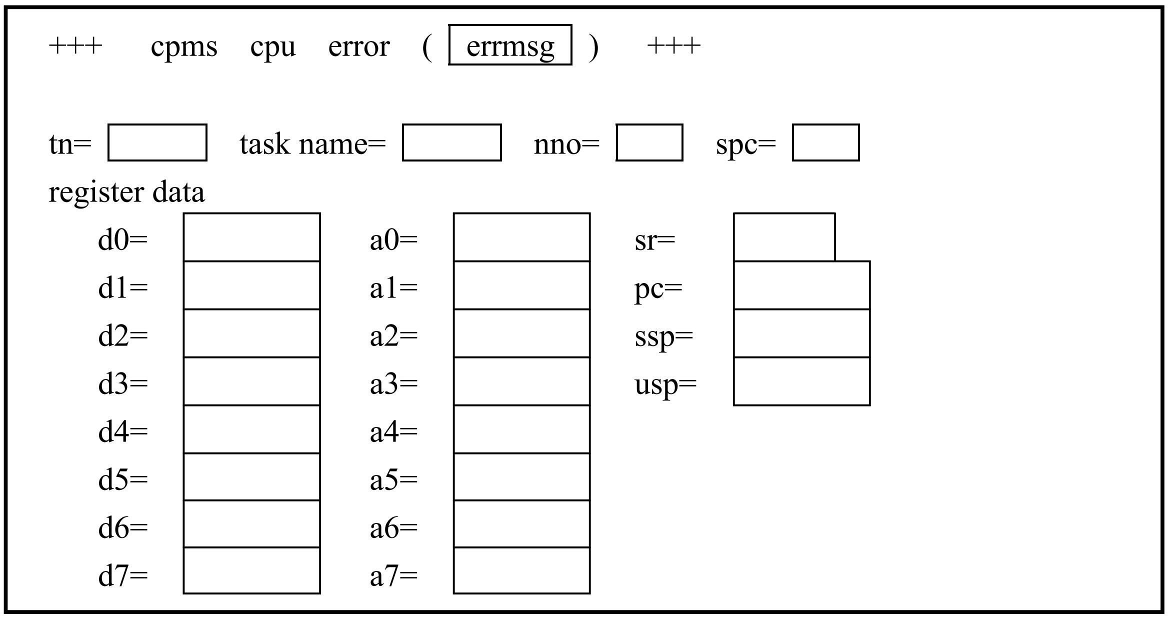
Format

*el

Note

For details of errors, refer to the manual supplied with CPMS.

- Output format 1 (other than address errors and bus errors; for 2α)



```
+++   cpms  cpu  error  ( errmsg )   +++

tn=        task name=        nno=       spc=
register data
    d0=            a0=            sr=
    d1=            a1=            pc=
    d2=            a2=            ssp=
    d3=            a3=            usp=
    d4=            a4=
    d5=            a5=
    d6=            a6=
    d7=            a7=
```

Example

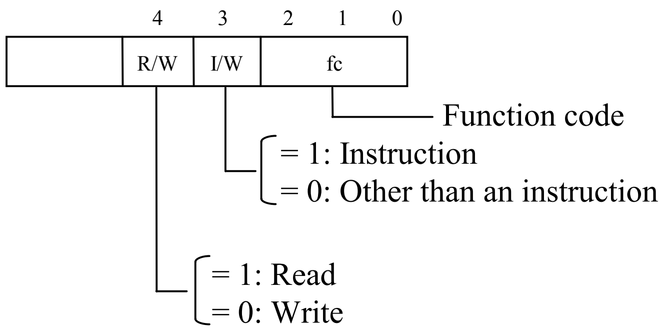
```
*el
+++ cpms cpu error (standard memory protect error) +++
tn=0x80   task name=fmcdbgt   nno=0x00   spc=0x0e01
register data
d0=0x00000000   a0=0x000f0c38   sr=Z
d1=0x00000001   a1=0x00001324   pc=0x00100a00
d2=0x000fa480   a2=0x00000080   ssp=0x000f8778
d3=0x2204000f   a3=0x00000200   usp=0x00100f8c
d4=0x000f0c38   a4=0x00101368
d5=0x000f9220   a5=0x00101328
d6=0x000f466c   a6=0x00100fdc
d7=0x00000000   a7=0x00100f8c
```

● Output format 2 (other than address errors and bus errors; for 2α)

```
+++   cpms  cpu  error  ( errmsg )   +++

tn=      task name=      nno=      spc=
register data
     d0=            a0=            fc=
     d1=            a1=            aa=
     d2=            a2=            ir=
     d3=            a3=            sr=
     d4=            a4=            pc=
     d5=            a5=            ssp=
     d6=            a6=            spc=
     d7=            a7=
```

tn:	Task number
task name:	Task name (not displayed when tn = 0)
nno:	N-coil number
spc:	Sequence program counter
fc:	



aa:	Accessed address
ir:	Instruction address
errmsg:	CPU error message (See page 79.)

● Output format 3 (for 2αE)

```
+++   cpms  cpu  error  ( errmsg )   +++

tn=          task name=          nno=        spc=
register data
     d0=              a0=              sr=
     d1=              a1=              pc=
     d2=              a2=              usp=
     d3=              a3=              msp=
     d4=              a4=              isp=
     d5=              a5=              vo=              vbr=
     d6=              a6=              sfc=             dfc=
     d7=              a7=              carc=            caar=

   Error-specific message (displayed in one of the formats shown below)
```

Formats of error-specific messages

- Exception handling interrupt after execution of an instruction; for coprocessor

```
insa  =
```

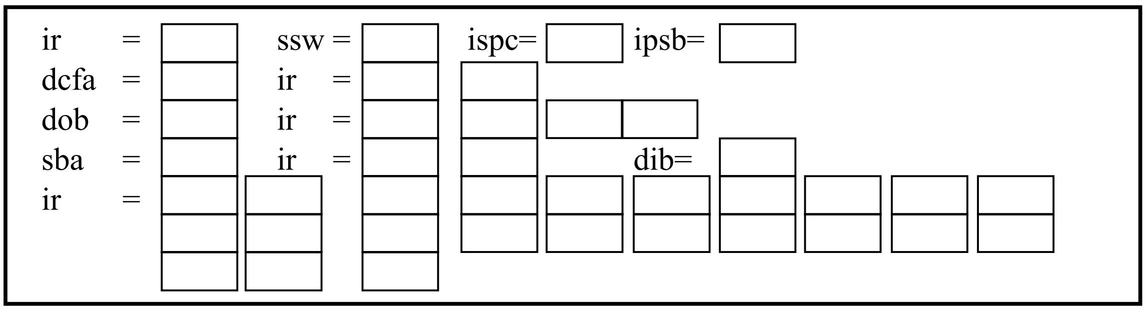
- Exception handling interrupt during execution of an instruction; for coprocessor

```
insa  =
ir    =
```

- Short bus cycle fault

```
ir    =            ssw  =            ispc=           ipsb=
dcfa  =            ir   =
dbc   =            ir   =
```

• Long bus cycle fault



CPU error messages

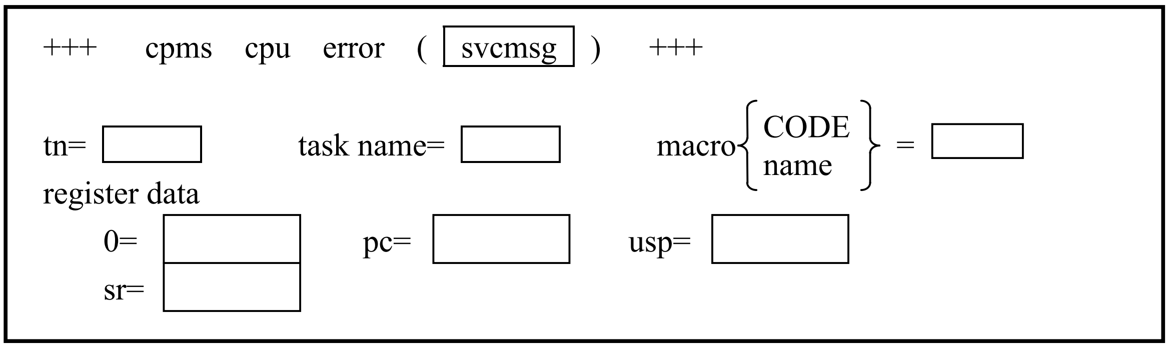
No.	Error message	Explanation
1	bus error	(Self-explanatory)
2	odd address access error	Attempt to access a word or long word at an odd-numbered address
3	illegal instruction	Attempt to execute an illegal instruction
4	zero divide	Attempt to execute a division instruction for division by zero
5	privilege violation	Attempt to execute a privileged instruction in user mode
6	nesting error	(Self-explanatory)
7	extension ram project error	Protection error with extension RAM
8	extension ram parity error	Parity error with extension RAM
9	S_mode illegal instruction	Illegal instruction in S-mode
10	standard memory protect error	Protection error with standard memory
11	S_ram parity error	Parity error with S_RAM
12	OS_ram parity error	Parity error with OS_RAM
13	wdt error	Watchdog timer error
14	ssp stack fence over	SSP limit exceeded
15	invalid interrupt	(Self-explanatory)

Example

```
*el
+++ cpms cpu error (odd address access error) +++
tn=0x7c    task name=pdbsend    nno=0x00    spc=0x0e01
register data
d0=0x00000000    a0=0x00ff00ff      fc=0x0012
d1=0x00000002    a1=0x00114ebc      aa=0x00ff00ff
d2=0x00000000    a2=0x00000000      ir=0x04e75
```

```
d3=0x00000201      a3=0x0011baa0     sr=Z
d4=0x00000004      a4=0x0017bb2a     pc=0x001400da
d5=0c00000000      a5=0x00111ce8     ssp=0x000f8770
d6=0x00000000      a6=0x00ff00ff     usp=0x0017ba50
d7=0x00000001      a7=0x0017ba50
```

● Output format 4 (for SVC errors)



svcmsg: SVC error message

No.	Error message	Explanation
1	SVC code error	Invalid SVC code
2	parameter error	Invalid parameter
3	parameter odd address error	Odd-numbered address specified in a parameter

macro CODE: Macro code (displayed when the error message "SVC code error" appears)

macro name: Macro name (displayed when an error message other than "SVC code error" appears)

Example

```
*el
+++ cpms svc error (parameter error) +++
tn=0x7e   task name=cvtest   macro name=rleas
register data
   a0=0x00000000   pc=0x00140090   usp=0x0017466c
   sr=
```

(15) er (Clears system errors.)

Function

This command clears error information.

Format

*er

Result

OK (0)   This result is always returned.

(16) ss (Displays the system status.)

Function

This command displays the status of the system.

Format

```
*ss
```

Result

The command displays the system status in the following format:

```
CPU status=****
```

****: RUN, SIMU, or STOP

(17) st (Sets the current time.)

Function

This command sets a new current time for the current time being managed by the controller.

Format

```
*st
YYYY.MM.DD.HH:MT:SS: yyyy.mm.dd.hh:mt:ss
```

Explanation

yyyy	Year (four digits of the calendar year)
mm	Month
dd	Day of month
hh	Hours
mt	Minutes
ss	Seconds

Note

A new current time can be set only where an extension memory with a clock is used.

(18) gt (Displays the current time.)

Function

This command displays the current time being managed by the controller.

Format

```
*gt
```

Result

```
yyyy.mm.dd.hh:mt:ss
```

yyyy	Year (four digits of the calendar year)
mm	Month
dd	Day of month
hh	Hours
mt	Minutes
ss	Seconds

Note

The current time can be displayed only where an extension memory with a clock is used.

(19) ld (Transfers the backup file to memory in the controller.)

Function

This command transfers the contents of the backup file to memory in the controller.

Format

```
*ld△ {-C}
     {-t△pname}
     {-s△sname}
     {-g△gname}
     {-a△aname}
     {-m△addr,len}
     {-T△tno}
     {-U△uno}
     {-S△sno}
     {-G△gno}
     {-f△fname}
```

Explanation

-C	Specifies batch loading. (Batch loading is enabled only when “sdebug” with the “initial” option specified is started.)
-t△pname	Specifies that only the program specified by “pname” be loaded.
-s△sname	Specifies that only the subroutine specified by “sname” be loaded.
-g△gname	Specifies that only the global data specified by “gname” be loaded.
-a△aname	Specifies that only the contents of the split area specified by “aname” be loaded.
-m△addr,len	Specifies that loading be performed according to a specified first address (addr) and a specified number of bytes (len).
-T△tno	Specifies that “tcb” for a task number (tno) be loaded.
-U△uno	Specifies that “uslcb” for a point number (uno) be loaded.
-S△sno	Specifies that the indirectly linked subroutine’s address table corresponding to an indirectly linked subroutine number (sno) be loaded.
-G△gno	Specifies that the indirectly linked global data’s address table corresponding to an indirectly linked global number (gno) be loaded.
-f△fname	Specifies that the file (fname) output by the sv command be loaded.

Result

The addresses indicating the loaded range are displayed in the following format:

```
address:0x********-0x********
```

Note

When an indirectly linked subroutine or indirectly linked global data is loaded, its management table is also loaded. In addition, when a task or user built-in subroutine is loaded, the appropriate “tcb” or “uslcb” is also loaded.

Before loading into memory in the controller, make sure that the task is in the dormant state.

(20) sv (Transfers the contents of memory in the controller to the backup file.)

Function

This command transfers the contents of memory in the controller to the backup file.

Format

```
*sv△ {-C}
     {-t△pname}
     {-s△sname}
     {-g△gname}
     {-a△aname}
     {-m△addr,len}
     {-f△fname}
```

Explanation

-C	Specifies that a batch transfer be performed.
-t△pname	Specifies that only the program specified by “pname” be transferred.
-s△sname	Specifies that only the subroutine specified by “sname” be transferred.
-g△gname	Specifies that only the global data specified by “gname” be transferred.
-a△aname	Specifies that only the contents of the split area specified by “aname” be transferred.
-m△addr,len	Specifies that a transfer be performed according to a specified first address (addr) and a specified number of bytes (len).
-f△fname	Specifies that a transfer be performed to the file specified by “fname.” If this option is omitted, a transfer to the backup file is assumed. If an error is detected during a transfer, the specified file is deleted, terminating the command.

Result

The addresses indicating the address space of a transfer destination are displayed in the following format:

```
address:0x********-0x********
```

(21) cm (Compares the backup file with the contents of memory in the controller.)

Function

This command compares the backup file with the contents of memory in the controller.

Format

```
*cm△ {-C}
     {-t△pname}
     {-s△sname}
     {-g△gname}
     {-a△aname}
     {-m△addr,len}
     {-f△fname}
```

Explanation

-C	Specifies that a batch comparison be performed.
-t△pname	Specifies that only the program specified by “pname” be compared.
-s△sname	Specifies that only the subroutine specified by “sname” be compared.
-g△gname	Specifies that only the global data specified by “gname” be compared.
-a△aname	Specifies that only the contents of the split area specified by “aname” be compared.
-m△addr,len	Specifies that a comparison be performed according to a specified first address (addr) and a specified number of bytes (len).
-f△fname	Specifies that a comparison be performed between the file specified by “fname” and memory in the controller. (Only the file specified by “sv” command may be used.) If this option is omitted, a comparison with the backup file is assumed. A file format that can be specified is the same as for the “ld” command (i.e., the a.out file format).

Result

- Upon normal comparison, the address range is displayed in the following format:

```
address:0x********-0x********
++ compare OK +++
```

- If any discrepancy is found during comparison, the unlike data is displayed in units of two bytes(word).

```
address:0x********-0x********
address=0x********    memory data=0x****    backup data=0x****
```

(22) dr and ds (Enable or disable DHP recording.)

Function

These commands are started by “sdebug” and toggle between DHP recording enable mode and DHP recording disable mode.

Format

```
*dr{-a}
*ds
```

Explanation

dr	Enters DHP recording enable mode.
-a	Records detailed DHP information.
ds	Enters DHP recording disable mode.

(23) ver (Displays version information.)

Function

This command displays the version number and revision number of CPMS.

Format

```
*ver
```

Result

```
CPMS 3.0
```

(24) smd (Displays or modifies the contents of all areas in memory.)

Function

This command displays or modifies the contents of all areas in memory in the actual machine, without checking the specified address range. The command also accepts an address range which would otherwise result in an access error.

Format

```
*smd
```

The subsequent format is the same as that of the “md” command, which displays or modifies the content of the memory area specified by addresses, except that the target to be accessed is not specified by “strage.”

Notes

- Accessing memory in the CPU using the “smd” command will affect the operation of the CPU. Be sure to understand fully the functions of the S10/2α before using this command. (Do not access memory carelessly.)
- This command has an effect only when the “sdebug” command with the -debug option specified has been initiated. In any other case, the command will result in an error.

(25) help (Displays a debugger menu.)

Function

This command lists the commands supported by “sdebug.”

Format

*help

Explanation

This command displays the abbreviations of the “sdebug” commands apnd provides a brief description of those commands, as shown below.

```
<Command>          <Function>
    qu      .....    task queue
    ab               task abort
    re               task release
-----------------------------------
When <next> appears, press any key to view the subsequent text.
```

(26) q (Terminates the debugger.)

Function

This command terminates the debugger.　If breakpoints are set, the command displays them and prompts the user to make a key input.

Format

*q

Note

When a message indicating breakpoints are set appears, execute the “rd” or “go” command to delete them.　Then, reissue this command.

(27) ! (Executes an MS-DOS command.)

Function

This command enables the user to use an MS-DOS command during execution of “sdebug.”

Format

*![MS-DOS command]

10.3 sdhp (Displays CPMS trace information.)

Function

This command displays CPMS trace information, called the debugging helper (DHP). It can be started alone or from “sdebug.”

Format

```
sdhp [△option]
    ≀
DHP for one screen is displayed.
    ≀
    {p}
    {-}
    {nothing}
    {q}
```

Options

-f△file	Name of the file to store the displayed DHP.
-u△site	Name of the site to be acted on. If this option is omitted, processing is performed on the default site set in advance.
-o△file	Name of the file to store the image data being displayed on the screen. (*) If a file having the same name as specified is already existent, it is deleted and a new file is created under the specified name.

Explanation

p,nothing	Displays the next page.
-	Displays the previous page.
q	Terminates the displaying of DHP.

The items displayed by “sdhp” are explained below.

```
0x0000   ad dr xx xx xx xx* xx xx xx xx xx xx xx xx xx xx
0x0010   xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
0x0020   xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
   ⋮                         ⋮
0x0070   xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
```

ad dr:	Relative address of the oldest data
*:	Delimiter between the oldest data and newest data
xx:	DHP in hexadecimal

(*) If this command is started with the -o option given to the “sdebug” command, and the file name specified by that -o option is the same as the one specified by the -o option of this command, then integrity of the file contents is unpredictable.

Note: For DHP, refer to the manual supplied with CPMS.

10.4 srpl (Loads programs.)

Function

This command loads all C programs into an actual machine.

Format

srpl [△option]

Option

-u△site:	Name of the site to be acted on. If this option is omitted, processing is performed on the default site set in advance.

Explanation

To stop the CPU, the user should operate the key switch on the controller.
Operate it as instructed by messages displayed on the screen.

11 MANAGEMENT TOOLS

11.1 smap (Displays map information.)

Function

This command lists various information items managed and maintained by the allocator.

Format

smap [△[option]···]

Options

-a	Lists split area information.
-e	Lists secondary partition area information.
-g	Lists global area information.
-p	Lists program information.
-s	Lists subprogram information.
-t	Lists task information.
-u△site	Name of the site to be acted on. If this option is omitted, processing is performed on the default site set in advance.
+a	Lists information in the order of addresses.
+g△name	Lists information on the specified name.
+n	Lists information in the alphabetical and numerical order of names.

Table 11-1 shows the allowed combinations of options. Information displayed by the “smap” command is described in items (1) to (7) below.

Table 11-1 Permitted Combinations of Options

No.	What is displayed	+ options			- options					
		g	a	n	g	a	e	p	s	t
1	Hierarchy map for a specified “garea”	0			0	0				
2	Hierarchy map for a specified “area”	0				0	0			
3	Entire hierarchy map for a specified “garea”	0			0	0	0			
4	Hierarchy map for a specified “garea” in the order of addresses	0	0		0	0				
5	Hierarchy map for a specified “area” in the order of addresses	0	0			0	0			
6	Entire hierarchy map for a specified “garea” in the order of addresses	0	0		0	0	0			
7	Hierarchy map for a specified “garea” in the order of names	0		0	0	0				
8	Hierarchy map for a specified “area” in the order of names	0		0		0	0			
9	Entire hierarchy map for a specified “garea” in the order of names	0		0	0	0	0			
10	List of requested information on a specified name	0			(0[1]	0	0	0	0	0)
11	List of requested information in the order of addresses		0		(0[2]	0	0	×	×	×)
12	List of requested information in the order of names			0	(0[3]	0	0	0	0	0)
13	All lists of requested information				(0[4]	0	0	0	0	0)

0[1]: Select one of the options marked “0” in parentheses.

0[2]: Specify the options marked “0” in parentheses. Options marked “×” may not be specified.

0[3]: Specify 0 to 6 options marked “0” in parentheses.

0[4]: Specify 0 to 6 options marked “0” in parentheses.

Nothing needs to be specified in blank fields.
If all options other than “-u” are omitted, all the lists are output in the order of addresses for “garea,” “area,” and “sarea.” For programs and subprograms, the lists are output in the order of names; and for tasks, they are output in the order of task numbers.

(1) Global area map

```
** allocator map **     site=site name
day mon dd hh:mm:ss yyyy
<garea>
    gname     paddr    lsn  laddr    uno  saddr    size
it  /gggggggg/bbbbbbbb cccc/llllllll uuuu/oooooooo jjjjjjj
** map end **
```

`day`: day of week, `mon`: month, `dd`: day, `hh`: hours, `mm`: minutes, `ss`: seconds, `yyyy`: year

i: Mode (s: system, u: user)

t: Type (o: os, t: task, s: subprogram, r: read-only global data, w: read/write global data,
a, c: global data with or without initial values for the device connected to system bus,
x: extension memory after site extension)

g: “garea” name

b: Physical address (relative address in site backup file, “{*******” when type is b or d)

c: Logical space number (*)

l: Logical address

u: Unit address (unit number of the auxiliary storage device; “{***” when the system has only main memory installed)

o: Sector address (sector address of the auxiliary storage device; “{*******” when the system has only main memory installed)

j: Size (in bytes)

(*) The logical space number is represented by the position of dedicated bits. (For example, the bits correspond to LS0, LS1, and so on, starting from the MSB, and the LSB corresponds to LS15.) In this system, LS0 is always used.
When the bit is set: Existent in the LS
When the bit is reset: Nonexistent in the LS

(2) Split area map

```
** allocator map **      site=site name
day mon dd hh:mm:ss yyyy
<area>
      gname/aname        raddr    size      lsn  laddr    uno   saddr
itkff/gggggggg/aaaaaaaa/rrrrrrrr/jjjjjjjj cccc/llllllll uuuu/oooooooo bbbbbbb
bbbbbbb
** map end **
```

`day`: day of week, `mon`: month, `dd`: day, `hh`: hours, `mm`: minutes, `ss`: seconds, `yyyy`: year

i: Mode (s: system, u: user)

t: Type (o: os, t: task, s: subprogram, r: read-only global data, w: read/write global data,
a, c: global data with or without initial values for the device connected to system bus,
b, d: global data without initial values for the device connected to system bus,
x: extension memory after site extension)

k: Area type (p: program, s: subprogram, d: global data with initial values, w: global data without initial values)

f: “ipl” flag (*1)

g: “garea” name

a: “area” name
r: Relative address (byte address indicating the position of the split area relative to the beginning of the global area)
j: Size
c: Logical space number (*2)
l: Logical address
u: Unit address (*3) (unit number of the auxiliary storage device; “{{{{” or “{***” when the system has only main memory installed)
o: Sector address (*3) (sector address of the auxiliary storage device; “{{{{{{{{” or “{*******” when the system has only main memory installed)
b: Backup file name

(*1)

0			7
Not used	rp	rc	rl

rp = 1: Start on powering up
rc = 1: IPL start
rl = 1: Restart
In this system, these bits are fixed as follows:
rp = 0
rc = 0
rl = 0

(*2) The logical space number is represented by the position of dedicated bits. In this system, LS0 is always used. (For example, the bits correspond to LS0, LS1, and so on, starting from the MSB, and the LSB corresponds to LS15.)
When the bit is set: Existent in the LS
When the bit is reset: Nonexistent in the LS

(*3) “{{…{” is displayed when the split area is allocated by the “sdfa” command. When the area is allocated by another command, “{**…*” is displayed.

(3) Secondary partition area map

```
** allocator map **                site=site name
day mon dd hh:mm:ss yyyy
<sarea>
          gname/aname/sname            raddr    size     lsn  laddr    uno   saddr
   ext-name
  d itk /gggggggg/aaaaaaaa/ssssssss/rrrrrrrr/jjjjjjjj cccc/llllllll uuuu/oooooooo
  vvvv/eeeeeeee[ yyyy/mn/dd hh:mm:ss yyyy/mn/dd hh:mm:ss yyyy/mn/dd hh:mm:ss]
                 ----------------- ------------------ ------------------
** map end **            ①      (*3)          ②      (*3)           ③       (*3)
```

`day`: day of week, `mon`: month, `dd`: day, `hh`: hours, `mm`: minutes, `ss`: seconds, `yyyy`: year

d: Loaded state (△: loaded into actual machine, *: loaded into backup file, @: not loaded into either)

i: Mode (s: system, u: user)

t: Type (o: os, t: task, s: subprogram, r: read-only global data, w: read/write global data,
a, c: global data with or without initial values for the device connected to system bus,
b, d: global data without initial values for the device connected to system bus,
x: extension memory after site extension)

k: Area type (p: program, s: subprogram, d: global data with initial values, w: global data without initial values)

g: "garea" name

a: "area" name

s: "sarea" name

r: Relative address (byte address indicating the position of the secondary partition area relative to the beginning of the split area)

j: Secondary partition area size (in bytes)

c: Logical space number (*1)

l: Logical address

u: Unit address (*2) (unit number of the auxiliary storage device; "{{{{" or "{***" when the system has only main memory installed)

o: Sector address (*2) (first sector address of the secondary partition area when the area is non-resident; first sector address of the split area when the area is resident; or "{{{{{{{{" or "********" when the system has only main memory installed)

v: Number of characters of an external name

e: External name (variable length)

yyyy, year; mn, month; dd, day; hh, hours; mm, minutes; ss, second (*3)

(*1) The logical space number is represented by the position of dedicated bits. (For examples, the bits correspond to LS0, LS1, and so on, starting from the MSB, and the LSB corresponds to LS15.) In this system, LS0 is always used.
When the bit is set: Existent in the LS
When the bit is reset: Nonexistent in the LS

(*2) "{...{" is displayed when the split area is allocated by the "sdfa" command. When the area is allocated by another command, "{*..." is displayed.

(*3) ① Date and time of loading into the backup file. (Date and time of execution of "sdfs" for global data without initial values; data and time of execution of "sload" in other cases. If no "sload" is executed after "sdload," "△······ △" is displayed.)

② Date and time of loading into the actual machine. (Date and time of execution of "debug ld." If no "ld" is executed, "△······ △" is displayed.)

③ Date and time of saving data in the actual machine. (Data and time of execution of "debug sv." If no "sv" is executed, "△······ △" is displayed.)

(4) Secondary partition area map (for VAL)

```
** allocator map **               site=site name
day mon dd hh:mm:ss yyyy
<sarea>
ext-name       vl name
itk/{{/{{/{{/ ll/vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
{{{{/{{{{{{{{ {{{{/{{{{{{{{ nnnn
/e·········e
** map end **
```

day: day of week, mon: month, dd: day, hh: hours, mm: minutes, ss: seconds, yyyy: year

i: Mode (s: system, u: user)
t: Value classification (e: value)
k: Value type (v: value)
l: Value length (number of bytes in the value)
v: Value
n: Number of characters of an external name
e: External name (variable length)

(5) Program map

```
** allocator map **               site=site name
day mon dd hh:mm:ss yyyy
<program>
       rmtn text     data     bss      stack    twork    lsn  laddr(cbn) sp
 uno  saddr    ldmid    pgm-name
itary q/kkkk mmmmmmmm dddddddd bbbbbbbb wwwwwwww eeeeeeee cccc/hhhhhhhh/zzzzzzzz
 uuuu/oooooooo nnnn/l…l vvvv/p…p
** map end **
```

day: day of week, mon: month, dd: day, hh: hours, mm: minutes, ss: seconds, yyyy: year

i: Mode (s: system, u: user)
t: Type (o: operating system, t: task)
a: Absolute classification (a: absolute, r: relocatable)

r:	Reentrant classification (r: reentrant, n: non-reentrant)
y:	Overlay classification (o: overlay; s: simple)
q:	Task classification (c: already created, d: not already created)
k:	Point number (*1) (ulsub), registration number (irsub), or subprogram; "{***" for others
m:	Procedure length
d:	Data length
b:	bss length
w:	Stack length
e:	Task work area length
c:	Logical space number (*2)
h:	First address in the program
z:	Last address in the program + 1 (stack pointer)
u:	Unit address (unit number of the auxiliary storage device; "{{{{" when the system has only main memory installed)
o:	Sector address (sector address of the auxiliary storage device; "{{{{{{{{" when the system has only main memory installed)
n:	Number of characters of a load module name
l:	Load module name (variable length)
v:	Number of characters of a program name
p:	Program name (variable length)

(*1) The point number is represented by the position of dedicated bits. (For example, the bits correspond to point number 1, point number 2, and so on, starting from the MSB, and the LSB corresponds to point number 16.)

When the bit is set: Registered at the point.

When the bit is reset: Not registered at the point.

(*2) The logical space number is represented by the position of dedicated bits. (For example, the bits correspond to LS0, LS1, and so on, starting from the MSB, and the LSB corresponds to LS15.) In this system, LS0 is always used.

When the bit is set: Existent in the LS

When the bit is reset: Nonexistent in the LS

(6) Subprogram map

```
** allocator map **     site=site name
day mon dd hh:mm:ss yyyy
<sub program>
        rmtn text     data     bss      stack    lsn  laddr    sp       uno
saddr ldmid    sub-name
itary q/kkkk mmmmmmmm dddddddd bbbbbbbb wwwwwwww cccc/hhhhhhhh/zzzzzzzz uuuu/ooo
ooooo nnnn/l…l vvvv/p××××p
** map end**
```

`day`: day of week, `mon`: month, `dd`: day, `hh`: hours, `mm`: minutes, `ss`: seconds, `yyyy`: year

i:	Mode (s: system, u: user)
t:	Type (o: operating system, s: subprogram)
a:	Absolute classification (a: absolute, r: relocatable)
r:	Reentrant classification (r: reentrant, n: non-reentrant)
y:	Overlay classification (o: overlay; s: simple)
q:	Subprogram (u: user built-in subroutine, i: indirectly linked subprogram, r: resident subprogram)
k:	Point number (*1) (ulsub), registration number (irsub), or subprogram; "{***" for others
m:	Procedure length
d:	Data length
b:	bss length
w:	Stack length
c:	Logical space number (*2)
h:	First address in the program (main memory address)
z:	Last address in the program + 1 (main memory address)
u:	Unit address (unit number of the auxiliary storage device; "{{{" when the system has only main memory installed}
o:	Sector address (sector address of the auxiliary storage device; "{{{{{{" when the system has only main memory installed)
n:	Number of characters of a load module name
l:	Load module name (variable length)
v:	Number of characters of a subprogram name
p:	Subprogram name (variable length)

(*1) The point number is represented by the position of dedicated bits. (For example, the bits correspond to point number 1, point number 2, and so on, starting from the MSB, and the LSB corresponds to point number 16.)

When the bit is set:	Registered at the point.
When the bit is reset:	Not registered at the point.

(*2) The logical space number is represented by the position of dedicated bits. (For example, the bits correspond to LS0, LS1, and so on, starting from the MSB, and the LSB corresponds to LS15.) In this system, LS0 is always used.

When the bit is set:	Existent in the LS
When the bit is reset:	Nonexistent in the LS

(7) Task map

```
** allocator map **     site=site name
day mon dd hh:mm:ss yyyy
<task>
  tn  rmtn tname  lvl eid s wdl pgm-name
i tttt mmmm ssssssss fru ll ee gg wwww nnnn p…p
```

day: day of week, mon: month, dd: day, hh: hours, mm: minutes, ss: seconds, yyyy: year

i:	Mode (s: system, u: user)
t:	Task number
m:	Multi-task number (stack position of the stack when a multi-task is used; “0000” in other cases)
s:	Task name
f:	Refreshable classification (s: serial or reusable, r: refreshable)
r:	Resident classification (r: resident, n: non-resident)
u:	Saving (When the task is non-resident: s: saved task, n: non-saved task. When the task is resident: [blank space])
l:	Task level
e:	Error ID
g:	Saved group number (“**” when the task is non-resident and not saved; “00” when the task is resident)
w:	Watchdog timer
n:	Number of characters of a program name
p:	Program name (variable length)

11.2 sirmap (Displays indirectly linked map information.)

Function

This command displays map information on indirectly linked subprograms or indirectly linked global data.

Format

sirmap [△option]

Options

-g:	Specifies that indirectly linked global data should be acted on.
-s:	Specifies that indirectly linked subprograms should be acted on. Both -g and -s may not be omitted.
-u△site	Name of the site to be acted on. If this option is omitted, processing is performed on the default site set in advance.

Output result

< $\boxed{\text{op}}$ no, list site (site name)>

irno= $\boxed{\text{irno}}$ name= $\boxed{\text{name}}$ la= $\boxed{\text{daddr}}$ ($\boxed{\text{salname}}$ + $\boxed{\text{offset}}$)

$\boxed{\text{op}}$	Distinction between indirectly linked global data (irglobal) or indirectly linked subprogram (irsub)
$\boxed{\text{irno}}$	Registration number of an indirectly linked global data or indirectly linked subprogram
$\boxed{\text{name}}$	Name of an indirectly linked global data or indirectly linked subprogram
$\boxed{\text{taddr}}$	First address
$\boxed{\text{salname}}$	External name registered by the allocator
$\boxed{\text{offset}}$	Offset from an external name registered by the allocator

Note

“salname” is displayed only when the -s option is specified in the “sirbld” command to register an indirectly linked subprogram or indirectly liked global data. “offset” is displayed only when the -a option is specified for the same purpose.

11.3 sadm (Displays the name corresponding to an address.)

Function

This command displays the name and other information corresponding to a specified logical address.

Format

```
sadm [△option]
++ address information display start → site (site name) ++
*addr: {addr}
       {q}
+++ address information display end ++
```

Explanation

addr	Address from which to get information
q	Terminates this command.

Options

-u△site	Name of the site to be acted on. If this option is omitted, processing is performed on the default site set in advance.
-o△file	Name of the file to which to output the operation result

The information displayed by "sadm" is explained below.

```
name=xxxxxxxx   type=xxx   raddr=xxxxxxxx
or
gname=xxxxxxxx   external name is not defined
```

name External name (sarea, program, subprogram) including the specified address

type Attribute of the external name

data:	sarea (global data)
pgm:	Program
sub:	subprogram

raddr Address relative to the beginning of the area identified with the external name

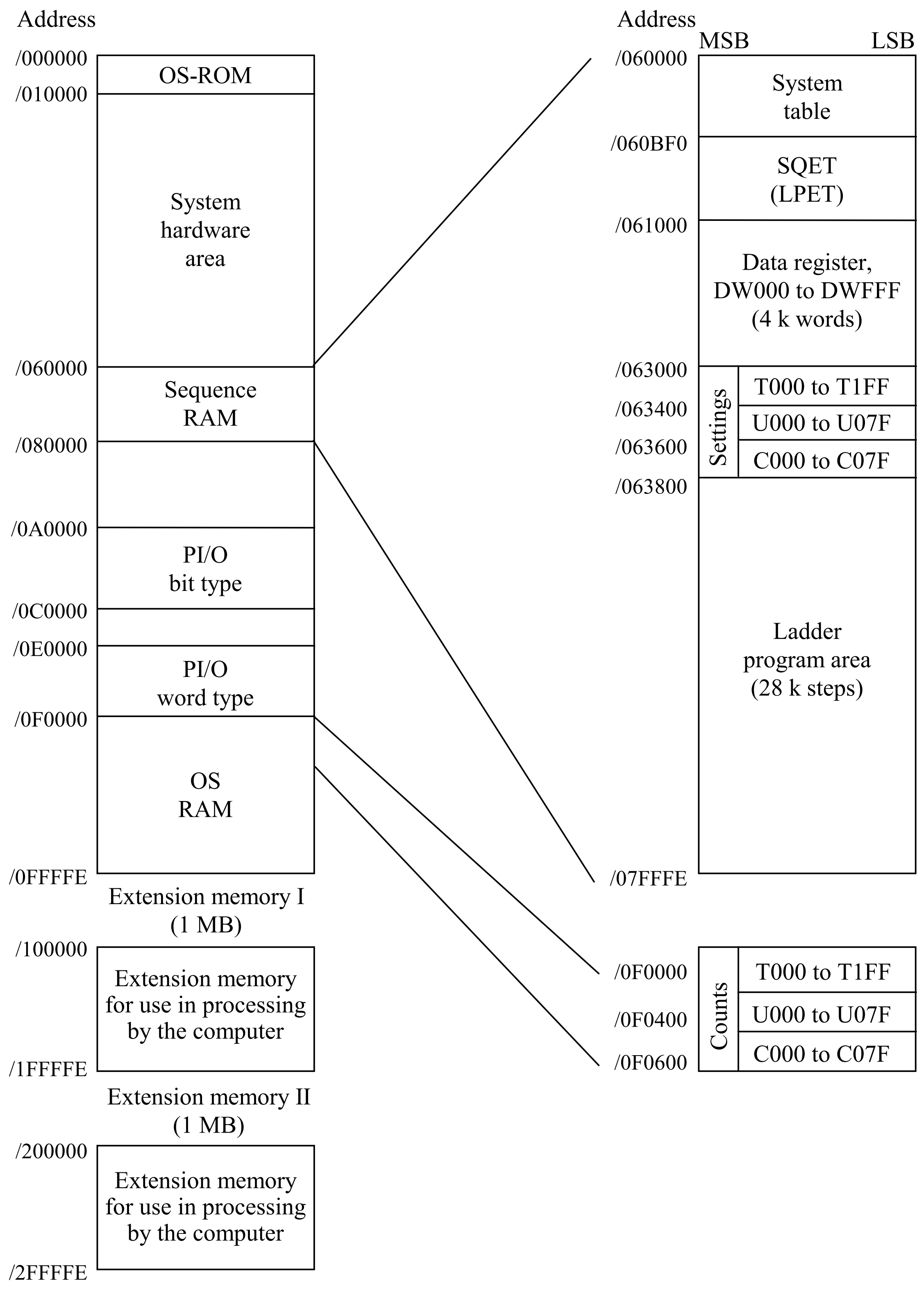
gname "garea" name containing the specified address -- only when no such external name is defined.

Note

If this command is started in "sdebug," and the -o option is specified in both "sdebug" and "sadm," then the output file may be destroyed. When starting "sadm" in "sdebug" with the -o option specified, do not specify the -o option in "sadm."

12 MEMORY MAP

12.1 HITACHI S10/2α Memory Map



12.2 PI/O Bit Form Area

Address	
/0A0000	X000 and later Contacts
/0A2000	Reserved for the system
/0A4000	Y000 and later Contacts, coils
/0A6000	Reserved for the system

Address	
/0A8000	G000 and later Contacts, coils
/0AA000	Reserved for the system
/0AC000	R000 and later Contacts, coils
/0AE000	Reserved for the system

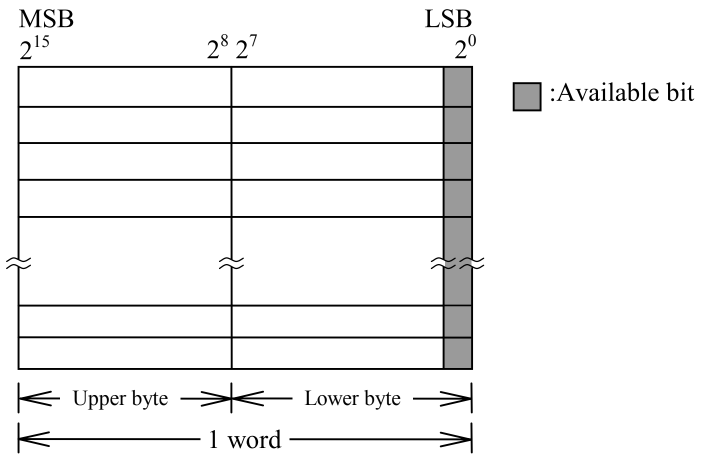
Address	
/0B0000	K000 and later Contacts, coils
/0B2000	T000 and later Coils
/0B2800	Reserved for the system
/0B3000	T000 and later Contacts
/0B3800	Reserved for the system
/0B4000	U000 and later Coils
/0B4800	U000 and later Previous coil values
/0B5000	U000 and later Contacts
/0B5800	Reserved for the system
/0B6000	CU000 and later Up coils
/0B6800	CD000 and later Down coils
/0B7000	C000 and later Contacts
/0B7800	CR000 and later Reset coils

Address	
/0B8000	N000 to NOFF Contacts, coils
/0B8800	N000 to NOFF For master control
/0B9000	P001 to P080 Contacts, coils
/0BA000	V000 and later Contacts
/0BC000	E000 and later Contacts
/0BE000	Z000 and later Contacts, coils
/0BE800	S000 and later Contacts

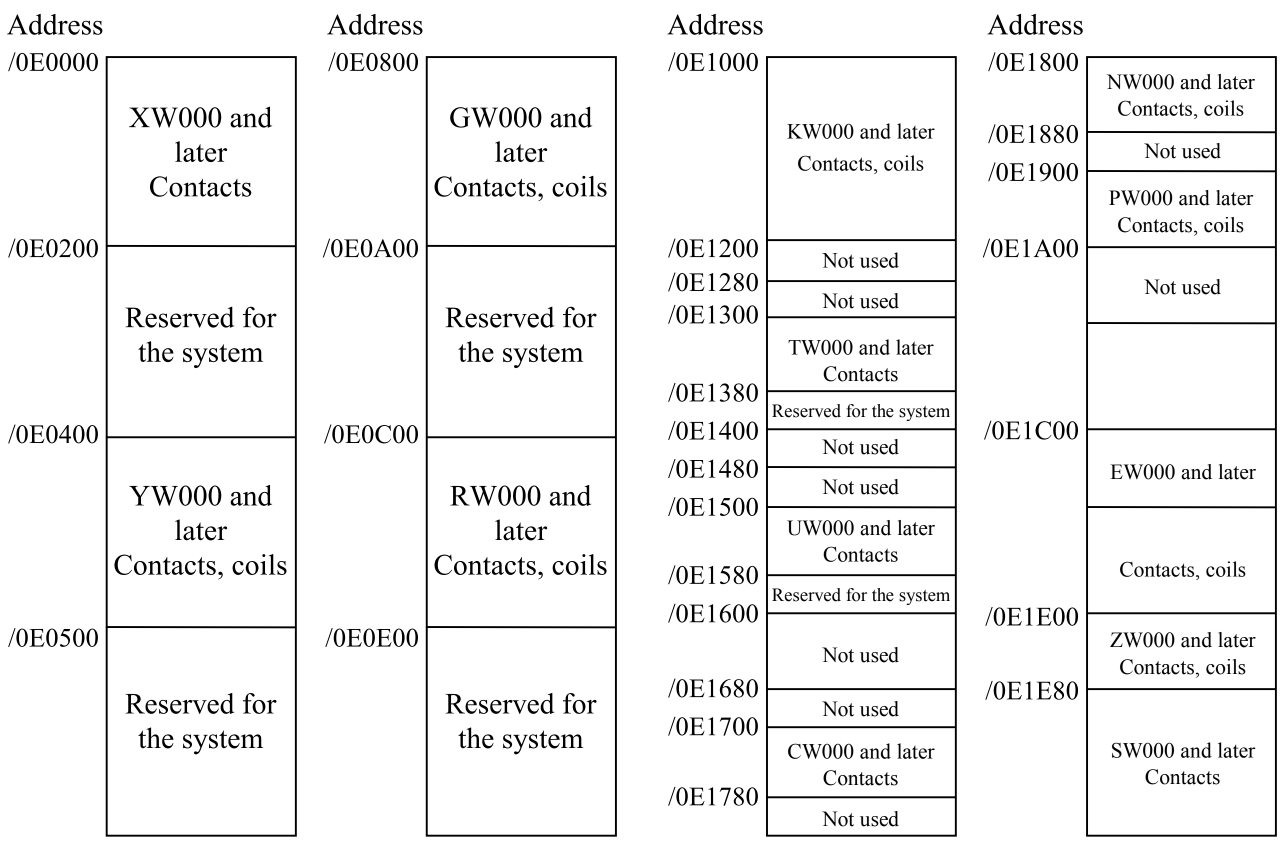
- This memory area is accessed on a word form (1 word = 2 bytes).
- In this memory area, only the LSB (least significant bit) is available.
- The byte (8-bit) form is used for addressing this memory area.

<Example of byte addressing>

Symbol	Address
X000	/0A0000
X001	/0A0002
X002	/0A0004
X003	/0A0005
⋮	⋮
X00E	/0A001C
X00F	/0A001E

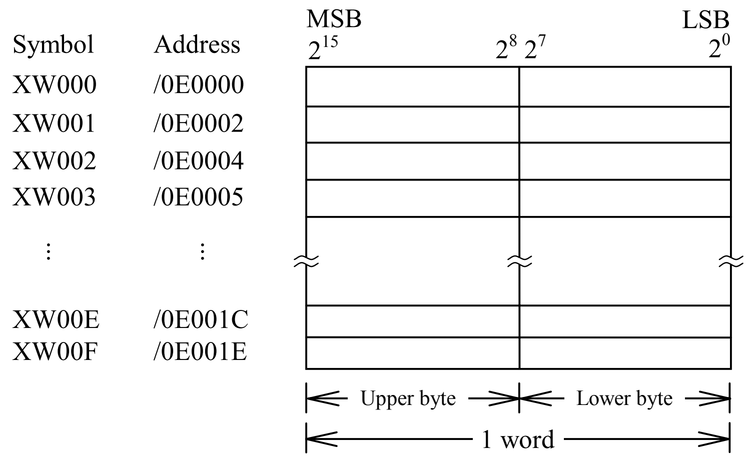


12.3 PI/O Word Form Area



■ This memory area is accessed on a word form (1 word = 2 bytes).
■ The byte (8-bit) form is used for addressing this memory area.

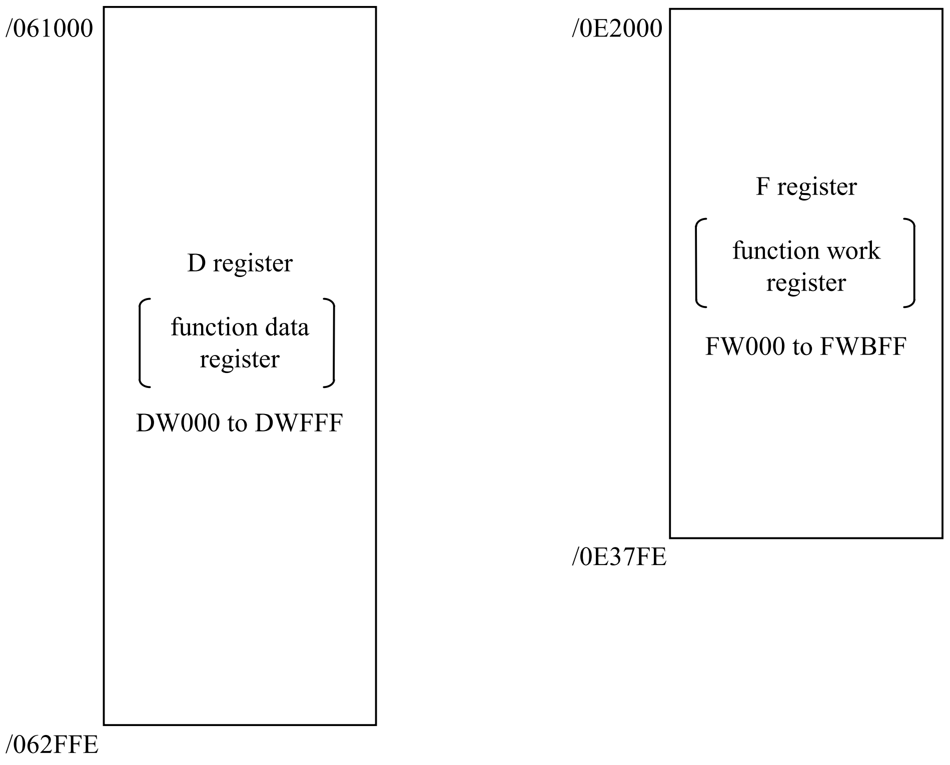
<Example of byte addressing>



<Correspondence between word and bit>

		MSB 2^{15}							LSB 2^0
XW000	/0E0000	X000	X001	X002	X003	....	X00D	X000E	X00F
XW010	/0E0002	X010	X011	X012	X013	....	X01D	X01E	X01F

12.4 User Work Area

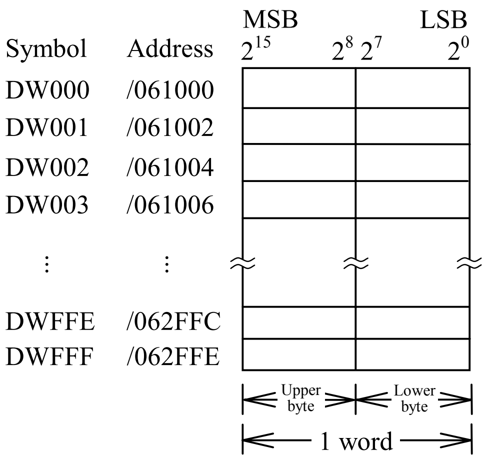


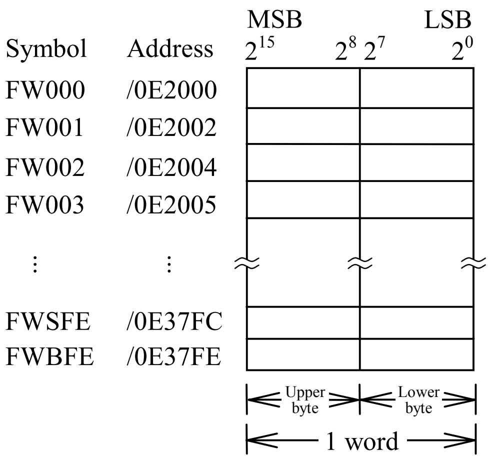
- This memory area is accessed on a word form (1 word = 2 bytes).
- The byte (8-bit) form is used for addressing this memory area.

<Example of byte addressing>

D register
(1 point equal to the length of 1 word)

F register
(1 point equal to the length of 1 word)





APPENDIXES

APPENDIX A LIBRARIES

A.1 Conditions for specifying libraries

Libraries specified in a command are accepted only when the conditions shown in Table A-1 are fulfilled.

Table A-1 Conditions for Specifying Libraries

Condition	Library name	Remarks
Created program uses CPMS macros.	cpms.lib	Refer to the CPMS General Description.
Indirectly linked addresses are referenced.	irad.lib	See "A. 3 Indirectly linked address reference subroutines," below.
An Indirectly linked subroutine is referenced.	site name.lib	
A library specific to the user is used.	User library name	

A.2 Order of specifying libraries

When using "sload" with library references, note the following points:

- Specify a library containing common subroutines later as far as possible.
- If some of the libraries specified contain the same name, specify before any other library the one that contains the object file the user wants to link.

A.3 Indirectly linked address reference subroutines

A.3.1 irglbad

Function

This subroutine fetches the global address value corresponding to a specified indirectly linked table number.

Format

```
int no;
int *irglbad (no)
```

Result

- When "no" is within the range of 1 to the maximum number, the corresponding global address is returned.
- When "no" is 0, the address of the global address management table is returned.

A.3.2 irsubad

Function

This subroutine fetches the subroutine address value corresponding to a specified indirectly linked subroutine number.

Format

int no;

int *irsubad (no)

Result

・When “no” is within the range of 1 to the maximum number, the corresponding subroutine address is returned.

・If the subroutine having a specified number is not yet loaded, a 0 is returned.

APPENDIX B NAMES AND STATEMENTS USABLE IN PROGRAMS

This chapter describes three restrictions that apply to programming in C language, assembly language, and other programming languages.

These restrictions are as follows:

・Reserved names of each programming language

・Statements that cannot be used in other operating systems

・Same names as subprograms provided by the system

B.1 Reserved names

Reserved names are those symbolic names which are set aside for special purposes according to the syntax of a programming language. No reserved names can be used for purposes other than the specified.

B.1.1 Assembly language

In assembly language, the user cannot use symbolic names contained in machine language instructions and assembly language instructions.

For the usable names, refer to the manual supplied with the crossing C compiler.

B.1.2 C language

The user cannot use symbolic names reserved according to the syntax of the C language. For these names, refer to the manual supplied with the crossing C compiler.

B.1.3 Reserved names in other programming languages

For the reserved names in each programming language other than the above, refer to the manual describing it.

B.2 Unusable statements

In CPMS, some statements that are used in other operating systems cannot be used in C standard functions.

B.2.1 Assembly language

In assembly language, there are no restrictions applied to the use of statements that are used in other operating systems.

B.2.2 C language

Functions that are used as system calls or I/O functions cannot be used.

B.3 Names used in the system

Users should be careful when using programs identified with the same name as that of a subroutine provided as standard in the system. All the subroutines provided as standard are contained in library files. If a user program to be used has the same name as a system subroutine, specify as the command file (-f option of sload) the object file in which the user program is defined. Otherwise, the subroutine stored in the library file under the same name is linked.

Listed below are the library files of the system and the names defined therein. In programming, take care so that names do not duplicate already defined names. If it is unavoidable to use a duplicate name, specify the object file to be linked and then the library file. This prevents a linkage with the subroutine from the library file.

Table B-1 lists subroutines provided in the system. (Subroutine names reserved for future extension are also listed in the table.)

Names defined in the cpms.lib file (Each attribute is folowed by a name.)

T abort	T chap	T chmod	T ctime	T cwake
T deley	T exit	T free	T gfact	T gtime
T mvmem	T queue	T rleas	T rserv	T sfact
T stime	T timer	T uspchk		

(*) T: Name defined in the text section.

APPENDIX C RECOVERY FROM FAILURES BY THE SYSTEM MANAGER

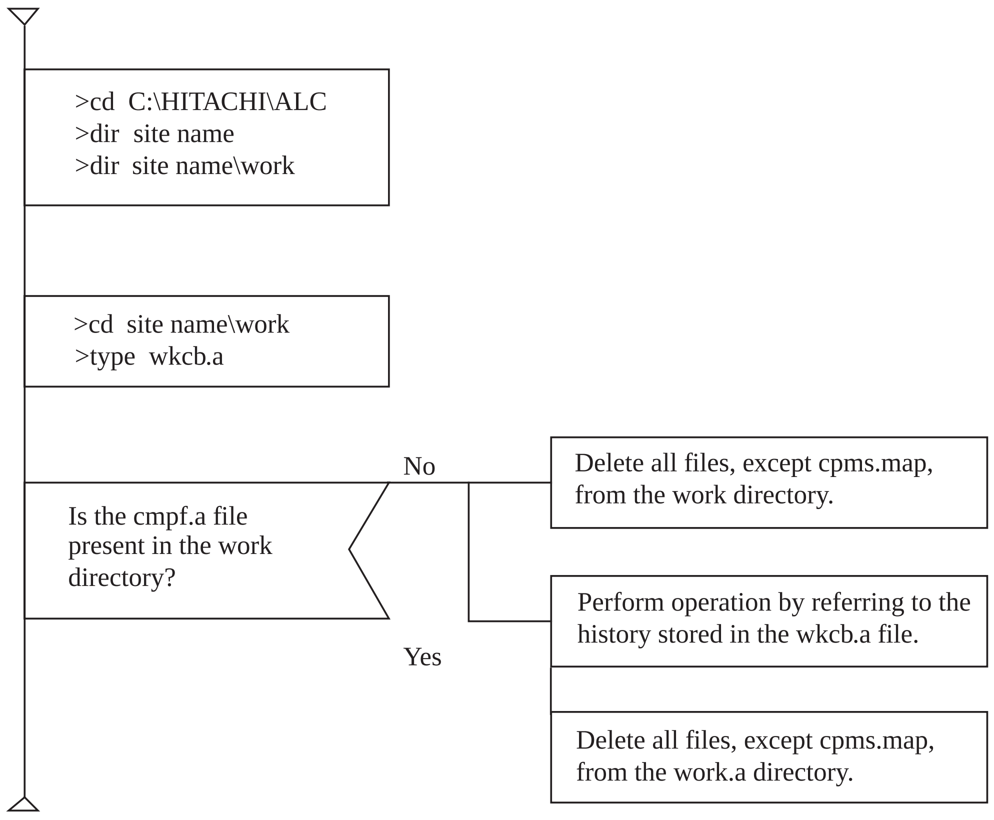
- Eliminating discrepancies

  The allocator keeps the history of a series of file accesses in the “wkcb.a” file (allocator work area control block) in the work directory (allocator work directory). In addition, the user can learn whether a series of file accesses is completed by checking whether the “cmpf.a” file is present. When a utility that accesses the file managed by the allocator causes either of the following allocator errors, correct the error by performing the procedure shown below.

  Errors that must be eliminated

  ・Error number 0003
  ・Error number 0004
  ・Error number 0005
  ・Error number 0007

  Procedure for error recovery



>cd C:\HITACHI\ALC
>dir site name
>dir site name\work
>cd site name\work
>type wkcb.a
Is the cmpf.a file present in the work directory?
No
Yes
Delete all files, except cpms.map, from the work directory.
Perform operation by referring to the history stored in the wkcb.a file.
Delete all files, except cpms.map, from the work.a directory.

Structure of the work directory

The following variables are supported as "wkcb.a" structures:

• cf. path name

• df. path name

• cd. path name

• dd. path name

Explanation

(1) cf. path name

This path name indicates that the file specified by the path name will be created. The file to be created is prepared in a completed form in the work directory. Re-link the file to the location specified by the path name. If the file is not prepared in the work directory, it indicates that the file is already re-linked.

(2) df. path name

Delete the file specified by the path name. If no such file is existent, this indicates that the file is already deleted.

(3) cd. path name

Create a directory as specified by the path name. If such a directory is existent, this indicates that one is already created.

(4) dd. path name

Delete the directory specified by the path name. If no such directory is existent, this indicates that the directory is already deleted.

Example

```
cf. C:\HITACHI\ALC\PCS01\EMF\SALMT.A
```

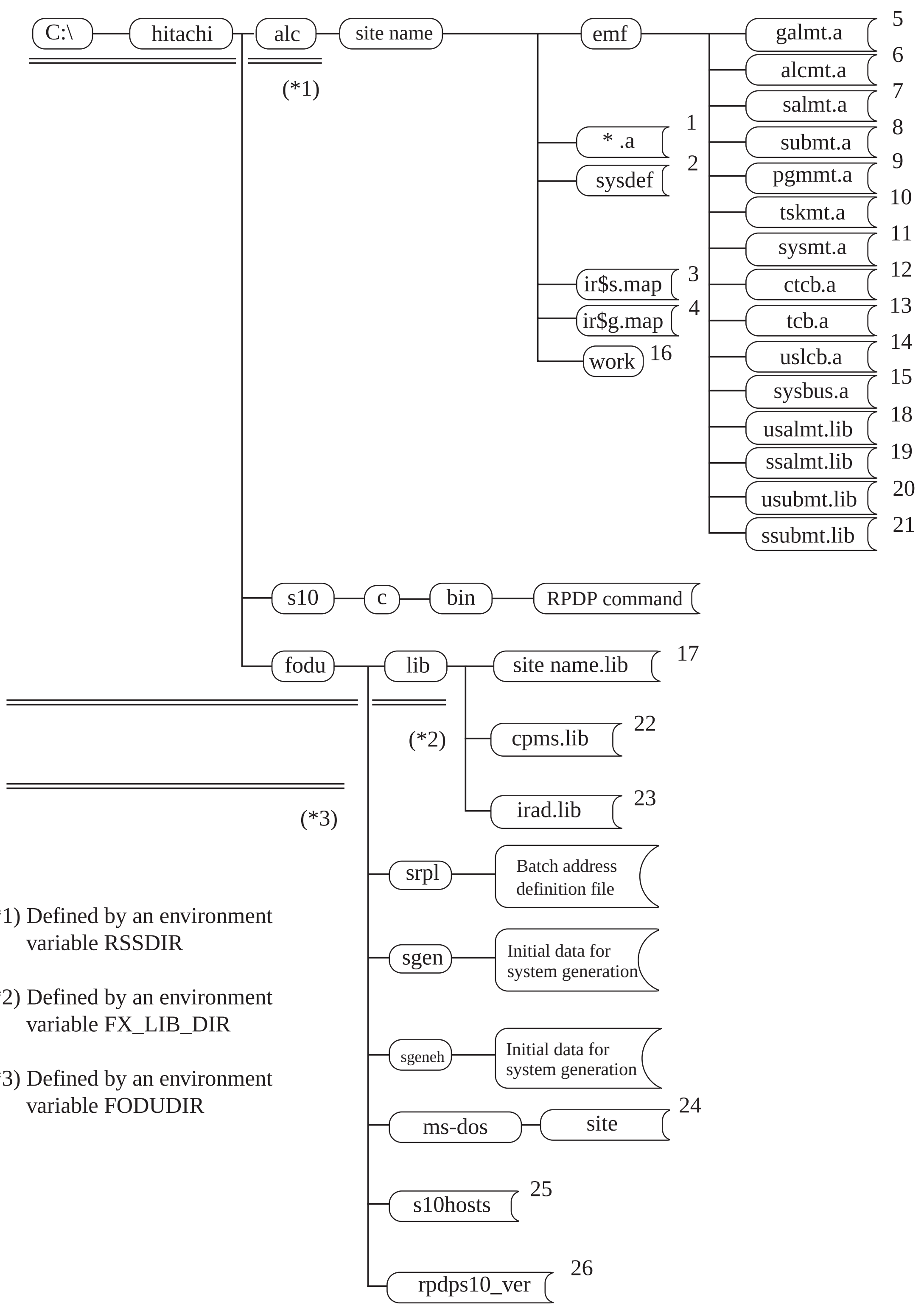
Operation

Execute DIR. If "SALMT.A" is found in C:\HITACHI\ALC\PCS01\WORK, execute the following:

```
CD△C:\HITACHI\ALC\PCS01\WORK
COPY△SALMT.A△..\EMF
DEL△SALMT.A
DEL△WKCB.A
```

APPENDIX D SITE MANAGEMENT FILES

The directory containing site management files has a structure as shown below.



(*1) Defined by an environment variable RSSDIR

(*2) Defined by an environment variable FX_LIB_DIR

(*3) Defined by an environment variable FODUDIR

Table D-1 Site Management Files (1/2)

No.	Abbreviation	Name	Contents	Initial setting	Read	Write
1	area name.A	Backup file	Backup copy of main memory or extension memory	sdfa	“scomp” and “sdebug” commands	Load this file with the “sload,” “sctask,” “sdtask,” or “sbuild” command.
2	sysdef	Site information definition file	Input data for the “sgen” command (other than the site name).	Created by the “sgen” command.	“ssi” command	Register an additional system bus card with the “sgen” command.
3	ir$s.map	“irsub” map information file	“isrub” map information	Created by the “sirbld” command.	“sirmap” command	Edit the file with the “sirbld” command.
4	ir$g.map	“irglobal” map information file	“irglobal” map information	Created by the “sirbld” command.	“sirmap” command	Register the file with the “sirbld” command.
5	galmt.a	“garea” management file	“garea” management information	Created by the “sgen” command.	“smap” command	Register an additional system bus card with the “sgen” command.
6	alcmt.a	“area” management file	“area” management information	Created by the “sgen” command.	“smap” command	Use the “sdfa” or “sdla” command to register or delete the file.
7	salmt.a	External (“sarea,” program, subprogram, and VAL) name management file	External name management information	Created by the “sgen” command.	“smap,” “sload,” and “scomp” commands	Use the “sdfs,” “sdls,” “sdfv,” “sdlv,” “sload,” or “sdload” command to register or delete the file.
8	submt.a	Subprogram management file	Subprogram management information	Created by the “sgen” command.	“scomp,” “smap,” and “sload” commands	Use the “sload,” “dload,” “sbuild,” or “dbuild” command to register or delete the file.
9	pgmmt.a	Main program management file	Main program management information	Created by the “sgen” command.	“scomp,” “load,” and “smap” commands	Use the “sload,” “sdload,” “sctask,” or “sdtask” command to register or delete the file.
10	tskmt.a	Task management file	Task management information	Created by the “sgen” command.	“smap” command	Use the “sctask” or “sdtask” command to register or delete the file.
11	sysmt.a	System management file	System management information	Created by the “sgen” command.	All allocator commands	
12	ctcb.a	Task control block for the actual machine	Task control block loaded into the actual machine	Created by the “sgen” command.		Use the “sctask” or “sdtask” command to register or delete the file.
13	tcb.a	Task control block for development	Control block to display maps	Created by the “sgen” command.	“smap” command	Use the “sctask” or “sdtask” command to register or delete the file.
14	uslcb.a	User built-in subroutine control block	User built-in subroutine control block	Created by the “sgen” command.	“sbuild” and “sdbuild” commands	Use the “sbuild” or “sdbuild” command to register or delete the file.

Table D-1 Site Management Files (2/2)

No.	Abbreviation	Name	Contents	Initial setting	Read	Write
15	sysbus.a	System bus card management file	System bus card management information	Created by the "sgen" command	"sgen" command	Register the file with the "sgen" command.
16	work	Work directory	Work files are created during command execution. Upon normal termination, these files are deleted.	Created by the "sgen" command.		
17	site name.lib	Library to link "irsub"		Created by the "sirbld" command.		Register or delete the file with the "sirbld" command.
18	usalmt.lib	Library to define user "sarea" and "value" addresses	Module to define the addresses of "sarea" and "value" resources for which the user type is "user"	"sdfs" and "sdfv" commands	"sload" command	Use the "sdfs," "sdls," "sdfv," or "sdlv" command to register or delete the file.
19	ssalmt.lib	Library to define system "sarea" and "value" addresses	Module to define the addresses of "sarea" and "value" resources for which the user type is "system"	"sdfs" and "sdfv" commands	"sload" command	Use the "sdfs," "sdls," "sdfv," or "sdlv" command to register or delete the file.
20	usubmt.lib	Library to define user subprogram addresses	Module to define the addresses of subprograms for which the user type is "user"	"sgen" command	"sload" command	Use the "sload" or "sdload" command to register or delete the file.
21	ssubmt.lib	Library to define system subprogram addresses	Module to define the addresses of subprograms for which the user type is "system"	"sload" command	"sload" command	Use the "sload" or "sdload" command to register or delete the file.
22	cpms.lib	CPMS macro linkage library	CPMS macro linkage module		"sload" command	
23	irad.lib	Indirectly linked address reference library	Module to reference indirectly linked addresses		"sload" command	
24	site	Default site name file	Default site names	"ssi" command	All commands other than "sgen"	Update the file with the "ssi" command
25	s10hosts	RPDP/S10 host definition file	IP address of the S10 and host name	Network administrator	"sdebug" command	
26	rpdps10_ver	Version file	Version information on the RPDP/S10			

APPENDIX E   ALLOCATOR ERROR MESSAGES

The allocator displays error messages in the format shown below.

(1) Errors in command lines

usage: $\underbrace{\text{command format}}_{①}$ (NO=$\underbrace{\text{XXXX}}_{②}$)

① Command format

② Error number

(2) Errors during processing

alloc: $\underbrace{\text{error message}}_{①}$ (NO=$\underbrace{\text{XXXX}}_{②}$)

① Error message

② Error number

Error messages are listed below.

Error Messages (1/4)

Error No.	Message	Nature of error	System's action	User's response
1	abnormal allocator master directory	Information managed by the allocator contained an error.	Terminates the processing and performs postprocessing.	See Appendix C.
2	abnormal allocator directory (permission denied)	Information managed by the allocator contained an error. (Failure to make an "in" mode access)	Terminates the processing and performs postprocessing.	See Appendix C.
3	abnormal allocator directory (failed to continue)	Information managed by the allocator contained an error. (When an error was detected, processing could not be continued.)	Terminates the processing and performs postprocessing.	See Appendix C.
4	abnormal allocator directory (failed to recover)	Information managed by the allocator contained an error. (Recovery from the error failed.)	Terminates the processing and performs postprocessing.	See Appendix C.
5	file access error (continue processing)	An error was detected during access to a file. (After the cause of the error is corrected, postprocessing is continued.)	Terminates the processing and performs postprocessing.	See Appendix C.
6	successfully recovered	An error was detected during access to a file. (The system was restored to the condition that existed before the start of the allocator.)	Terminates normally.	None
7	file access error (continue recovering)	An error was detected during access to a file. (After the cause of the error is corrected, the system is restored to the condition that existed before the start of the allocator.)	Terminates the processing and performs postprocessing.	See Appendix C.
8	internal logic error	Internal logic error	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
9	MAPIB illegal	An inconsistency was found in mapping information.	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.

Error Messages (2/4)

Error No.	Message	Nature of error	System's action	User's response
10	file access error	File access error	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
11	file access error	File access error	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
12	file access error	File access error during processing for 'a.open.'	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
13	file access error	File access error during error handling for 'a.open.'	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
14	file access error	File access error during processing for 'a.move.'	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
15	file access error	File access error during error handling for 'a.move.'	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
16	file access error	File access error during processing for 'a.clos.'	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
17	file access error	File access error during error handling for 'a.clos.'	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
18	internal logic error (signal)	Error during signal processing (internal logic error)	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
19	abnormal allocator directory	File access error during fault condition check.	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
20	abnormal RALB	Invalid STTUP information	Terminates the processing and restores the condition that existed before the allocator was started.	See Appendix C.
101	specified site is undefined	An undefined site name was given.	Terminates the processing and restores the condition that existed before the allocator was started.	Check the site name.
102	specified site is busy or undefined	The site was being locked by another process. Or the site was not yet defined.	Terminates the processing and restores the condition that existed before the allocator was started.	Check the site name or retry the command.
103	specified garea is undefined	An undefined global area was specified.	Terminates the processing and restores the condition that existed before the allocator was started.	Check the "garea" name.
104	specified area is undefined	An undefined split area was specified.	Terminates the processing and restores the condition that existed before the allocator was started.	Check the "area" name.
105	specified sarea is undefined	An undefined secondary partition area was specified.	Terminates the processing and restores the condition that existed before the allocator was started.	Check the "sarea" name.
106	specified external name is undefined	An undefined external name was given.	Terminates the processing and restores the condition that existed before the allocator was started.	Check the external name.
107	specified area is already defined	An already-defined split area was specified	Terminates the processing and restores the condition that existed before the allocator was started.	Change the "area" name.

Error Messages (3/4)

Error No.	Message	Nature of error	System's action	User's response
108	specified sarea is already defined	An already-defined secondary partition area was specified.	Terminates the processing and restores the condition that existed before the allocator was started.	Change the "sarea" name.
109	specified external name is already defined	An already-defined external name was given.	Terminates the processing and restores the condition that existed before the allocator was started.	Change the external name.
110	specified area is already used	An area already in use was specified.	Terminates the processing and restores the condition that existed before the allocator was started.	Change the position specification.
111	not enough space	Insufficient free space	Terminates the processing and restores the condition that existed before the allocator was started.	Delete areas that are no longer needed.
112	permission denied	Illegal access privilege level	Terminates the processing and restores the condition that existed before the allocator was started.	Specify the -S option. Or specify the -d or -w option to allocate a split area.
113	specified garea is universal space	GM space limit exceeded.	Terminates the processing and restores the condition that existed before the allocator was started.	Specify the -d or -w option.
114	can not align non-resident garea	Alignment was specified for a non-resident area.	Terminates the processing and restores the condition that existed before the allocator was started.	Specify the -d or -w option.
115	specified sarea is neither global nor bulk	An attempt was made to delete a secondary partition area other than for global data.	Terminates the processing and restores the condition that existed before the allocator was started.	Specify the -d or -w option to allocate a split area.
116	sarea is defined for the specified area	An attempt was made to delete a split area in which a program or subprogram was loaded or a split area that was divided into secondary partition areas.	Terminates the processing and restores the condition that existed before the allocator was started.	Execute "sdload" or "sdls" before deleting a split area.
117	mapping table overflow	A mapping table overflowed.	Terminates the processing and restores the condition that existed before the allocator was started.	Delete areas that are no longer needed.
120	STTUP table overflow	A startup table overflowed.	Terminates the processing and restores the condition that existed before the allocator was started.	Delete areas that are no longer needed.
201	specified options go not agree with one another	Both the XX option and the YY option were given. They are mutually exclusive.	Terminates the processing.	Check whether the options were specified correctly.
202	protection code must be from 0 to 7	The specified protection code was not within the range of 0 to 7.	Terminates the processing.	Check the -k option.
203	align parameter must be from 0 to 12	The value specified in the "align" parameter was not within the range of 0 to 12.	Terminates the processing.	Check the -a option.
204	too many characters in specified name	A name consisting of too many characters was given.	Terminates the processing.	Shorten the name to eight characters or less.
205	illegal character is found	An invalid special character was contained in the specified name.	Terminates the processing.	Check the name.

Error Messages (4/4)

Error No.	Message	Nature of error	System's action	User's response
206	illegal option is foung	An invalid option was given.	Terminates the processing.	Check the option.
207	the same option is specified twice or more	The same option was defined twice.	Terminates the processing.	Do not define the same option two or more times.
208	value extent out of range	Too large a value was given as a VAL.	Terminates the processing.	Check the range of VAL values.
209	illegal format of numeric value	Numeric data was specified in an invalid format.	Terminates the processing.	Check the format of the specified numeric data.
301	sdfa gname/aname size [option]	A command (sdfa) was specified incorrectly.	Terminates the processing.	Check the command specification.
302	sdla aname [option]	A command (sdla) was specified incorrectly.	Terminates the processing.	Check the command specification.
303	sdfs aname/sname size [option]	A command (sdfs) was specified incorrectly.	Terminates the processing.	Check the command specification.
304	sdls sname [option]	A command (sdls) was specified incorrectly.	Terminates the processing.	Check the command specification.
305	sdlv ename value [option]	A command (sdfv) was specified incorrectly.	Terminates the processing.	Check the command specification.
306	sdlv ename [option]	A command (sdlv) was specified incorrectly.	Terminates the processing.	Check the command specification.

APPENDIX F LOADER ERROR MESSAGES

The loader displays error messages in the format shown below.

(1) Errors in command lines

usage: command format (NO=XXXX)

①   ②

① Command format

② Error number

(2) Errors during processing

Command name: error message (NO=XXXX)

①   ②

alloc: error message (NO=XXXX)

③   ④ (Note)

① Error message displayed by the loader

② Number of an error message displayed by the loader

③ Error message displayed by the allocator

④ Number of an error message displayed by the allocator

(Note) The allocator displays error messages for recovery from faults. It may not display error messages for some faults.

Error messages are listed below.

Error Messages (1/7)

Error No.	Message	Nature of error	System's action	User's response
0	system error	System error	Terminates the command after displaying this error message.	(*)
2	specified area is not found	An undefined area was specified to register a program in it. (An undefined split area was specified by the -a option.)	Terminates the command after displaying this error message.	Check the split area name specified by the -a option. Then enter the command again.
3	abnormal allocator directory	There was an error detected in the management table used by the allocator. (This message is displayed by "sload" or "scomp.")	Terminates the command after displaying this error message.	See Appendix C.
4	abnormal allocator directory	There was an error detected in the management table used by the allocator. (This message is displayed by "sdload.")	Terminates the command after displaying this error message.	See Appendix C.
5	file copy error	An error was detected while a file was being copied.	Terminates the command after displaying this error message.	See Appendix C.

Error Messages (2/7)

Error No.	Message	Nature of error	System's action	User's response
6	too many characters in specified name	A name longer than eight character was given in an option. (This message is displayed by "sload" or "scomp.")	Terminates the command after displaying this error message.	Shorten the name to eight characters or less. Then enter the command again.
7	specified area is already used	An area already in use was specified by the -C (upper- case) or -p option.	Terminates the command after displaying this error message.	Check the address specified by the -C (uppercase) option or the relative address specified by the -p option. Then enter the command again.
8	specified core block is not found	No such core block was existent.	Terminates the command after displaying this error message.	Specify an existing core block. Then enter the command again.
9	executable program name is already registered	An already-registered executable program name was given.	Terminates the command after displaying this error message.	Change the program name. Then enter the command again.
10	undefined external name	External reference information was undefined.	Terminates the command after displaying this error message.	Correct the undefined reference name. Then enter the command again.
11	alloc file open or close error	An error was detected while the management table used by the allocator was being opened or closed.	Terminates the command after displaying this error message.	See Appendix C.
12	management table operation error	An error was detected while the management table used by the allocator was being operated on.	Terminates the command after displaying this error message.	See Appendix C.
13	too many characters in specified name	A name longer than eight character was given in an option. (This message is displayed by "sdload.")	Terminates the command after displaying this error message.	Shorten the name to eight characters or less. Then enter the command again.
14	sload pname -S -u ······	An option was specified incorrectly. (For example, an invalid option was given, a required option was omitted, or an option already in use was specified.)	Terminates the command after displaying this error message.	Check the option. Then enter the command again.
15	illegal character if found	A character not allowed in an option was specified. Or no "pname" was specified. (This message is displayed by "sload" or "scomp.")	Terminates the command after displaying this error message.	Remove the characters other than alphanumerics and underscores or specify "pname." Then enter the command again.

Error Messages (3/7)

Error No.	Message	Nature of error	System's action	User's response
16	illegal character is found	A character not allowed in an option was given. Or no "pname" was specified. (This message is displayed by "sdload.")	Terminates the command after displaying this error message.	Remove the characters other than alphanumerics and underscores or specify "pname." Then enter the command again.
17	logical space number is not between 0 and 14	The specified logical space number was not within the range of 0 to 14.	Terminates the command after displaying this error message.	Specify the logical space number within the range of 0 to 14. Then enter the command again.
18	core block range over	The operation range of a program was greater than the range of the specified block.	Terminates the command after displaying this error message.	Extend the range of the core block. Then enter the command again.
19	incorrect loading address	The operation range of a subprogram was greater than the range of the split area.	Terminates the command after displaying this error message.	An absolute address was specified incorrectly. Check the address, then enter the command again. Or the -C option was specified incorrectly. Correct it.
20	not enough area for loading	There was no free space large enough to load a program or subprogram.	Terminates the command after displaying this error message.	Increase the split area size. Then enter the command again.
21	system error	Although a request to allocate an area in main memory was made dynamically during processing by the loader, it failed. (System error)	Terminates the command after displaying this error message.	(*)
22	undefined external name	An undefined external name was detected. (The "system" or "user" type was specified incorrectly.)	Terminates the command after displaying this error message.	Check the classification of "system" and "user." Then enter the command again.
23	undefined external name	An undefined external name was detected. (The registered external name was not for global data.)	Terminates the command after displaying this error message.	Check the external name. Then try again from compilation.
24	specified position is out of range	A position beyond the boundaries of the area was specified for registration.	Terminates the command after displaying this error message.	Check the position. Then enter the command again.

Error Messages (4/7)

Error No.	Message	Nature of error	System's action	User's response
25	illegal area type	An incorrect area type (program or subprogram) was specified.	Terminates the command after displaying this error message.	Check the area type. Then enter the command again.
26	illegal combination of options	An option was specified incorrectly.	Terminates the command after displaying this error message.	Check the option. Then enter the command again.
27	sload -S -u site ··· +P	An option for loading a program was specified incorrectly.	Terminates the command after displaying this error message.	Check the option. Then enter the command again.
28	sload -S -u site ··· (+S, +U)	An option for loading a subprogram or built-in subroutine was specified incorrectly.	Terminates the command after displaying this error message.	Check the option. Then enter the command again.
29	sload -S -u site ··· +D	An option for loading global data was specified incorrectly.	Terminates the command after displaying this error message.	Check the option. Then enter the command again.
30	sdload -S -u pnaml site ······	An option was specified incorrectly. (For example, an option was specified incorrectly in "sdload," or a duplicate definition was made.)	Terminates the command after displaying this error message.	Check the option. Then enter the command again.
31	specified program or subprogram is not found	A request was made to delete an unregistered program or subprogram.	Terminates the command after displaying this error message.	Check the program or subprogram name. Then enter the command again.
32	can not delete a program registered as a task or uslsub	A request for deleting a program failed because the program had been registered as a task or built-in subroutine.	Terminates the command after displaying this error message.	Execute "sdtask" or "sdbuild" before executing "sdload."
33	file close error	An error was detected during an attempt to close a file.	Terminates the command after displaying this error message.	See Appendix C.
34	undefined input file name	An undefined input file was specified.	Terminates the command after displaying this error message.	Check the input file name. Then enter the command again.
35	can not open the file; undefined	An attempt was made to open an undefined file.	Terminates the command after displaying this error message.	(*)
36	can not open the file; busy	An attempt was made to open a file being locked by another process.	Terminates the command after displaying this error message.	(*)
37	file open error	An error was detected while a file was being opened.	Terminates the command after displaying this error message.	(*)
38	write error	An error was detected while a file was being written.	Terminates the command after displaying this error message.	(*)
40	read error	An error was detected while a file was being read.	Terminates the command after displaying this error message.	(*)

Error Messages (5/7)

Error No.	Message	Nature of error	System's action	User's response
41	specified initialized glb is not found	An attempt was made to delete undefined global data.	Terminates the command after displaying this error message.	(*)
42	glb data is not loaded	An attempt was made to delete global data not yet loaded.	Terminates the command after displaying this error message.	(*)
60	system error	A specified number of words could not be written to a file.	Terminates the command after displaying this error message.	(*)
61	a.out format is abnormal	A load module was in an invalid format.	Terminates the command after displaying this error message.	Try again from compilation (or assembling).
62	specified text or data is not found	An attempt was made to load a program, subprogram, or built-in subroutine that had no text. Or an attempt was made to load global data having text.	Terminates the command after displaying this error message.	For the former, add a text or data section. For the latter, remove the text section.
63	global or bulk data is found	A definition of global data with initial values, or a value definition, appeared in a load module.	Terminates the command after displaying this error message.	Remove the definition.
64	illegal load module format as a glb or bulk data	A load module was in an invalid format when an attempt was made to load global data.	Terminates the command after displaying this error message.	Define global data. Then try again from compilation (or assembling).
65	there is no data to be loaded	Data registration was specified, but there was no global data to be registered.	Terminates the command after displaying this error message.	Define global data. Then try again from compilation (or assembling).
70	illegal character found	Non-numerical data was specified in an option.	Terminates the command after displaying this error message.	Check the option. Then enter the command again.
71	illegal core block format	A core block was specified incorrectly.	Terminates the command after displaying this error message.	Check the core block number. Then enter the command again.
72	data size is larger than glb or bulk size	A data size larger than that of the area for global data was specified.	Terminates the command after displaying this error message.	Check the data. Then enter the command again.

Error Messages (6/7)

Error No.	Message	Nature of error	System's action	User's response
73	specified area is not found	A secondary partition area in which to load the specified global data was undefined.	Terminates the command after displaying this error message.	Allocate a secondary partition area. Then enter the command again.
74	illegal loading address format	A loading destination location was specified incorrectly.	Terminates the command after displaying this error message.	Check the destination location. Then enter the command again.
75	specified area is not found	A secondary partition area not intended for data loading was specified for loading global data.	Terminates the command after displaying this error message.	Change the definition of the secondary partition area so that data can be loaded into it. Then enter the command again.
77	-s option is not valid	The -c option was specified for a program.	Terminates the command after displaying this error message.	Remove the -c option. Then enter the command again.
78	too large stack length	A stack length was specified incorrectly. (An area of the specified stack length could not be allocated.)	Terminates the command after displaying this error message.	Check the stack length. Then enter the command again.
79	number of task which share the same main program is limited from 2 to 160	The specified number of tasks for multitasking was not within the range of 2 to 160.	Terminates the command after displaying this error message.	Specify the number of such tasks within the range of 2 to 160. Then enter the command again.
80	specified address or size is not on a longword boundary	A value other than a multiple of 4 was given as the stack length.	Terminates the command after displaying this error message.	Specify a multiple of 4 as the stack length. Then enter the command again.
83	illegal task data length	A task data length was specified incorrectly.	Terminates the command after displaying this error message.	Check the task data length. Then enter the command again.
84	glb data is already loaded	Global data already loaded was specified.	Terminates the command after displaying this error message.	Execute "sdload" before executing "sload."
87	specified program is not found	An unregistered executable program was specified. (This message is displayed by "scomp.")	Terminates the command after displaying this error message.	Check the executable program name. Then enter the command again.
88	internal logic error	Internal logic error. (This message is displayed by "scomp.")	Terminates the command after displaying this error message.	(*)

Error Messages (7/7)

Error No.	Message	Nature of error	System's action	User's response
89	undefined input file name	An undefined load module was specified by the -i option. (This message is displayed by "scomp.")	Terminates the command after displaying this error message.	Check the load module name. Then enter the command again.
90	scomp pname -S -u site ······	An option was specified incorrectly. (This message is displayed by "scomp.")	Terminates the command after displaying this error message.	Check the option. Then enter the command again.
91	file open or copy error	An error was detected while a file was being opened or copied. (This message is displayed by "scomp.")	Terminates the command after displaying this error message.	(*)
92	file read error	An error was detected while a file was being read. (This message is displayed by "scomp.")	Terminates the command after displaying this error message.	(*)
93	too many characters in specified name	A name longer than eight character was specified by an option. (This message is displayed by "scomp.")	Terminates the command after displaying this error message.	Shorten the name to eight characters or less. Then enter the command again.
94	illegal character is found	A character not allowed in an option was specified. (This message is displayed by "scomp.")	Terminates the command after displaying this error message.	Remove the characters other than alphanumerics and underscores. Then enter the command again.
95	text size unmatched	A mismatch was found in text length between an executable program and load module. (This message is displayed by "scomp.")	Continues processing even after displaying this error message.	There is a difference from the load module. Check the executable module against the load module.
96	data size unmatched	A mismatch was found in data length between an executable program and a load module. (This message is displayed by "scomp.")	Continues processing even after displaying this error message.	There is a difference from the load module. Check the executable module against the load module.
97	abnormal allocator directory	There was an error in the management table used by the allocator. (This message is displayed by "scomp.")	Terminates the command after displaying this error message.	(*)
98	file access error (a.backup)	An error was detected during access to a backup file. (This message is displayed by "scomp.")	Terminates the command after displaying this error message.	(*)

(*) ① Check if the free memory capacity and free hard disk capacity of your PC are enough.

② When your PC is Windows® 2000 or Windows® XP, check the contents of the event log.

③ If the above items ① and ② are OK, reinstall the RPDP/S10 and the Crossing C compiler (MCC68K).

APPENDIX G BUILDER ERROR MESSAGES

The builder displays error messages in the format shown below.

(1) Errors in command lines

usage: $\underset{\textcircled{1}}{\underline{\text{command format}}}$ (NO=$\underset{\textcircled{2}}{\underline{\text{XXXX}}}$)

① Command format

② Error number

(2) Errors during processing

Command name: $\underset{\textcircled{1}}{\underline{\text{command format}}}$ (NO=$\underset{\textcircled{2}}{\underline{\text{XXXX}}}$)

allod: $\underset{\textcircled{3}}{\underline{\text{error message}}}$ (NO=$\underset{\textcircled{4}}{\underline{\text{XXXX}}}$) (Note)

① Error message displayed by the builder

② Number of an error message displayed by the builder

③ Error message displayed by the allocator

④ Number of an error message displayed by the allocator

Note: The allocator displays error messages for recovery from faults. It may not display error messages for some faults.

Error messages are listed below.

Error Messages (1/5)

Error No.	Message	Nature of error	System's action	User's response
1	operand syntax error	An operand was specified incorrectly.	Terminates the command after displaying this error message.	Make sure what is the acceptable data. Then retry the command.
2	operand combination error	Operands were combined incorrectly.	Terminates the command after displaying this error message.	Make sure what is the acceptable data. Then retry the command.
3	too many or missing operands	There were too many or missing operands.	Terminates the command after displaying this error message.	Make sure what is the acceptable data. Then retry the command.
4	numeric value out of range	Out-of-range data was entered.	Terminates the command after displaying this error message.	Make sure what is the acceptable data. Then retry the command.

Error Messages (2/5)

Error No.	Message	Nature of error	System's action	User's response
7	no executable module exists	There was no executable module that creates a task.	Terminates the command after displaying this error message.	Check the executable module name. Then retry the command.
8	task already defined	A name already in use was specified as the name of a task to be created.	Terminates the command after displaying this error message.	Change the task name. Or delete the existing task if it is no longer needed. Then retry the command.
9	core block number out of range	A specified core block was not within the range of 1 to the maximum core block number.	Terminates the command after displaying this error message.	Specify a core block within the range of 1 to the maximum core block number. Then retry the command.
10	different core block number for load command	A specified core block number was not the same as had been specified in loading the program.	Terminates the command after displaying this error message.	Specify the same core block number as had been specified in loading the program.
11	top core block number is greater than last core block number	An out-of-sequence core block number was specified.	Terminates the command after displaying this error message.	Specify a core block number between the first and last core block numbers.
12	can not define core block number for -r option	The -r option was specified together with a core block number.	Terminates the command after displaying this error message.	Make sure what is the acceptable data. Then retry the command.
14	user task number out of range	The task number specified for a user task was not within the range of 1 to the maximum task number (114).	Terminates the command after displaying this error message.	Specify a task number within the range of 1 to the maximum task number (114).
15	system task number out of range	A task number other than 128 was specified for a system task.	Terminates the command after displaying this error message.	Specify the task number 128. Then retry the command.

Error Messages (3/5)

Error No.	Message	Nature of error	System’s action	User’s response
16	task number is already used	A task number already in use was specified.	Terminates the command after displaying this error message.	Change the task number. Or delete the task in use if it is no longer needed. Then retry the command.
17	can not find undefined TCB	There was no free TCB available.	Terminates the command after displaying this error message.	Delete tasks that are no longer needed. Then retry the command.
18	can not find undefined PCB	There was no free PCB available.	Terminates the command after displaying this error message.	Delete tasks that are no longer needed. Then retry the command.
19	value of error - processing ID out of range (0-15)	The specified error handling ID was not within the range of 0 to 15.	Terminates the command after displaying this error message.	Specify an error handling ID within the range of 0 to 15. Then retry the command.
21	work number out of range with multi-task	During creation of multiple tasks, the specified work section creation number was not within the range of 1 to the maximum work section creation number.	Terminates the command after displaying this error message.	Specify a number within the range of 1 to the maximum work section creation number.
22	can not define work number, 0 is set on load command	Although a 0 had been specified in a load command as the work section creation number, another work section creation number was given.	Terminates the command after displaying this error message.	Supply a multi-task specification. Then rerun from “sload.”
23	can not find undefined work number for multi-task	During creation of multiple tasks, there was no free work section available.	Terminates the command after displaying this error message.	If there is a work section that is no longer needed, execute “sdtask” and then retry the command. If not, execute “sload” and then retry the command.
24	work number is already defined for multi-task	During creation of multiple tasks, the specified work section creation number was already in use.	Terminates the command after displaying this error message.	Find a work section creation number not in use. Then retry the command.
25	can not define work number for non-resident task	Although no multitasking was specified, a work section creation number was specified.	Terminates the command after displaying this error message.	Delete the -r option. Then retry the command.

Error Messages (4/5)

Error No.	Message	Nature of error	System's action	User's response
26	program is already defined as resident task	A program already in use as a resident task was specified.	Terminates the command after displaying this error message.	When creating the specified task as another task, delete the program and reload the task as one of the multiple tasks. Then retry the command.
27	watch dog timer out of range (0, 2-65535)	The specified time to monitor execution was not 0 or it was not within the range of 2 to 65,535.	Terminates the command after displaying this error message.	Set the time to 0 or within the range of 2 to 65,535.
28	save area group number out of range	The specified area group number was out of range.	Terminates the command after displaying this error message.	Specify the save area number within the range of 1 to the maximum save area group number. Then retry the command.
29	can not set -c, -a, -g, -f option except for resident task	An invalid option was given for a resident task.	Terminates the command after displaying this error message.	Make sure what is the acceptable data. Then retry the command.
31	priority level of user task out of range (0-4)	The level specified for a user task was not within the range of 0 to 4.	Terminates the command after displaying this error message.	Specify a correct level. Then retry the command.
32	priority level of system task out of range (0-4)	The level specified for a system task was not within the range of 0 to 4.	Terminates the command after displaying this error message.	Make sure what is the acceptable data. Then retry the command.
33	the task is already deleted or undefined	A task to be deleted was not registered.	Terminates the command after displaying this error message.	Specify a registered task name or register such a task. Then retry the command.
34	user can not delete system task	The user attempted to delete a system task.	Terminates the command after displaying this error message.	Do not attempt to delete system tasks.
35	point number out of range	The point number specified for a built-in subroutine was out of range.	Terminates the command after displaying this error message.	Specify a correct point number for a built-in subroutine. Then retry the command.

Error Messages (5/5)

Error No.	Message	Nature of error	System's action	User's response
36	point number is already used	The point number specified for a built-in subroutine was already in use.	Terminates the command after displaying this error message.	Find a point number not in use. The retry the command.
37	subprogram is not defined	A subroutine not yet created was specified.	Terminates the command after displaying this error message.	Specify the name of a created subroutine. Then retry the command.
39	monitor task number out of range	The number specified for the task to be monitored was out of range.	Terminates the command after displaying this error message.	Specify a valid task number. Then retry the command.
40	monitor task is not defined	A non-existing task was specified to monitor it.	Terminates the command after displaying this error message.	Register the task to be monitored. Then retry the command.
41	user can not define system task	The user attempted to delete a system task.	Terminates the command after displaying this error message.	Specify the -S option. Then retry the command.
43	the subroutine is already deleted or undefined	A subroutine to be deleted was not registered.	Terminates the command after displaying this error message.	Check the names of the created subroutines. Then retry the command.
44	the point number is not defined	A point number to be deleted was not registered.	Terminates the command after displaying this error message.	Check the point numbers in use. Then retry the command.
50	abnormal allocator management table	There was an error in the management table used by the allocator.	Terminates the command after displaying this error message.	Check the management file used by the allocator. Then retry the command.

APPENDIX H COMMUNICATION (Ethernet, GP-IB, AND RS-232C)

The RPDP/S10 supports the Ethernet, GP-IB bus, and RS-232C interface for connection between the personal computer and PCs.

H.1 Ethernet-based Communication

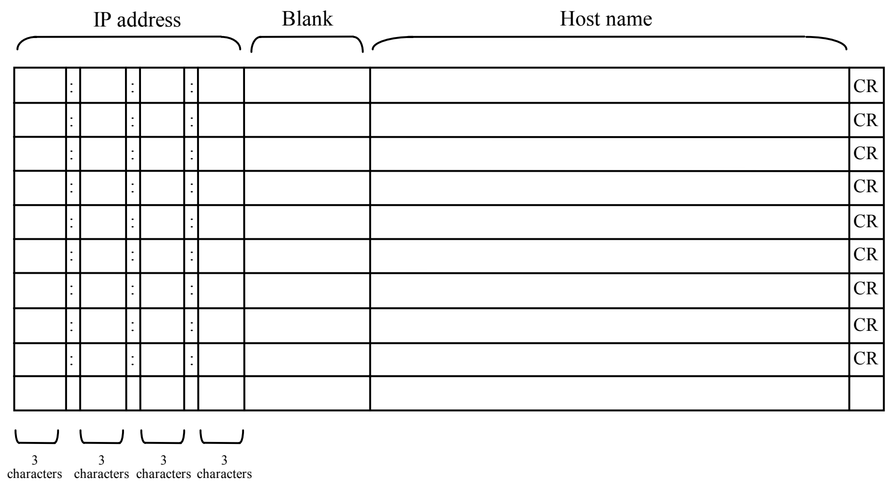
To perform communication using the Ethernet, the personal computer must be ready for connection to the Ethernet. The following sections describe settings required for the RPDP/S10 and PCs.

H.1.1 Setting the S10Hosts File

The RPDP/S10 identifies PCs with host names. The following file is used to make host names correspond to IP addresses:

C:\HITACHI\FODU\S10Hosts

In the S10Hosts file, set host names (PCs' names) and their corresponding IP addresses in the following format:



H.1.2 Configuring at MS-DOS Prompts

The RPDP/S10 uses environment variables to set up a method of communication. Set ETHER in the environment variable RSCOM. Also set the host name of the remote PC to be connected to the Ethernet in the environment variable RSHOST.

Example:

```
C:\>set RSCOM=ETHER
C:\>set RSHOST=pcs01
```

H.2 GP-IB based Communication

To enable GP-IB based communication, use the PCMCIA-GPIB card from National Instruments Corporation. Also install the supplied software to make the RPDP/S10 ready for GP-IB based communication. The following section describes settings required for the RPDP/S10 and PCs.

H.2.1 Configuring at an MS-DOS Prompt

The RPDP/S10 uses an environment variable to set up a method of communication. Set GPIB in the environment variable RSCOM.

Example:

```
C:\>set RSCOM=GPIB
```

H.3 RS-232C based Communication

No particular hardware or software is required for RS-232C based communication. Usually, use the COM1 port to connect to the PC.

H.3.1 Configuring at an MS-DOS Prompt

The RPDP/S10 uses an environment variable to set up a method of communication. Set RS232C in the environment variable RSCOM or delete RSCOM. If the environment variable RSCOM is undefined or it is not set to ETHER or GPIB, then the RPDP/S10 enables RS-232C-based communication.

Example:

```
C:\>set RSCOM=RS232C
```

or

```
C:\>set RSCOM=
```

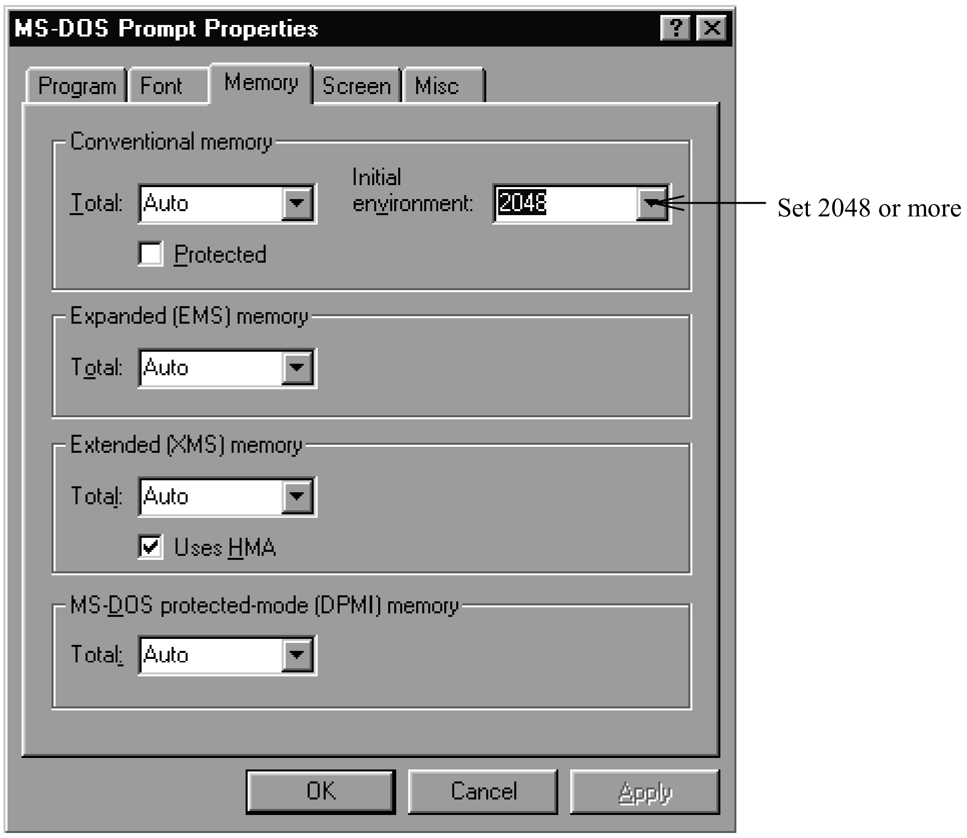
APPENDIX I C LANGUAGE PROGRAM DEVELOPMENT ENVIRONMENT AND SYSTEM EXECUTION ENVIRONMENT

(1) Setting MS-DOS prompt properties

The RPDP/S10 uses many environment variables. For this reason, if the MS-DOS prompt is activated without changing properties to start the RPDPE or RPDP command, the message "Out of environment space" may appear. If this happens, set the value 2048 or greater as the initial size of the environment variable in the memory tab for MS-DOS prompt properties.

In the program tab for MS-DOS prompt properties, specify a batch file. Then, just activating the MS-DOS prompt enables the desired environment to be automatically set up. Where shortcuts are created for multiple MS-DOS prompts to set up different environments, the user can get the desired environment with ease.

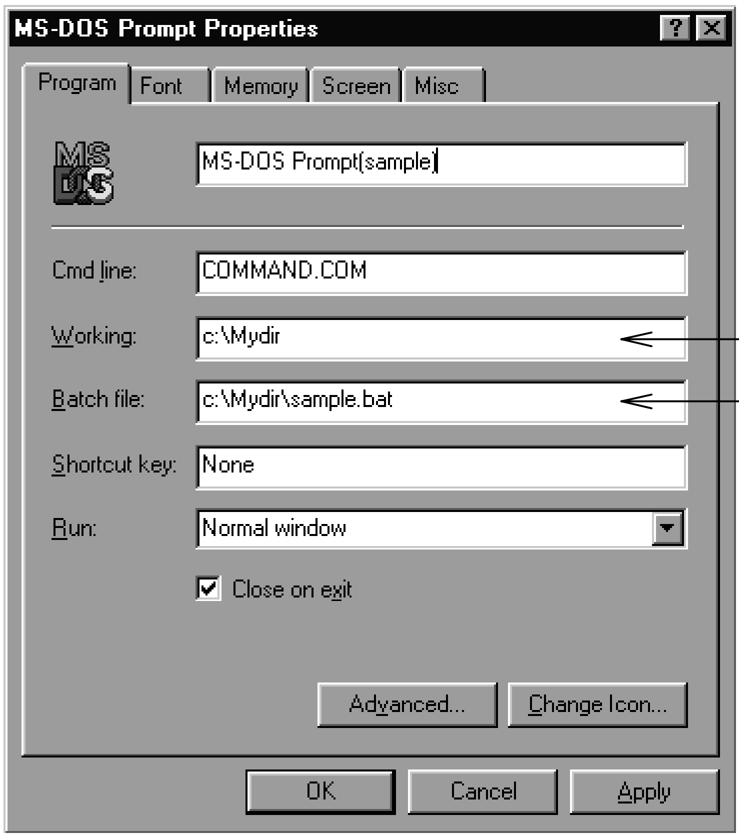
<Changing the initial memory size set in an environment variable>



Conventional memory
Total: Auto
Initial environment: 2048
Protected
Expanded (EMS) memory
Total: Auto
Extended (XMS) memory
Total: Auto
Uses HMA
MS-DOS protected-mode (DPMI) memory
Total: Auto

Set 2048 or more

<Example of automatically setting environment variables>



MS-DOS Prompt Properties

MS-DOS Prompt(sample)

Cmd line: COMMAND.COM

Working: c:\Mydir

Batch file: c:\Mydir\sample.bat

Shortcut key: None

Run: Normal window

☑ Close on exit

Work directory

Configuration program (The “sample.bat” file is automatically opened.)

<Sample “C:\Mydir\sample.bat” file>

```
call rpdpe
set RSCOM=ETHER
set RSHOST=pcs01
set RSSITE=site01
```

Do not change the “rpdpe.bat” and “rpdp.bat” files as much as possible. Create a batch file that opens the “rpdpe.bat” or “rpdp.bat” file as shown on the left. Change environment variables in the batch file, as necessary.

(2) Operation from multiple MS-DOS prompts

Activate multiple MS-DOS prompts and specify different sites at them. Then, the user can perform programming for multiple sites. However, the same site cannot be subjected to simultaneous programming.

APPENDIX J SAMPLE OPERATION

C:\>rpdpe or rpdp (configuration)

rpdp:	for the H-S10/2α
rpdpe:	for the H-S10/2αE, 2αH, and 2αHf

C:\>sgen (generation)

site:	uf3
type:	S10/2A
size:	384
addr:	0x140000
tsize:	192
ssize:	64
irsmax:	1024
grsize:	64
grwsize:	64
irgmax:	1024

0x140000 0x150000 0x160000 0x190000

garea	os	sub	glbr		task	glbrw		ems		
area	rpdp	a3	a5		al a2	a6		a10		a11
sarea	Not managed by RPDP	a4	s1	s2		s3	s4	s10	s11	

C:\>ssi uf3

C:\>sdfa task/a1 32768 -p

C:\>demo (compilation and task creation)

C:\>srpl (loading into the actual machine)

Contents of the file

DEMO.BAT

```
mcc68k -c -f -s -Fsm demo.c (-f is set if floating-point operations are supported.)
asm68k -l -f “case, -t” > demo.lst demo.src
sdtask demo
sdload demo +p
sload demo -a al -f cmddemo -w 1024 +p
sctask demo demo -t 2 -v 3
```

CMDDEMO

```
load c:\test\demo\demo.obj
load c:\hitachi\fodu\lib\cpms.lib
load c:\hitachi\fodu\lib\irad.lib
```

<Accessing PI/O units>

- Defining VALs

```
C:\>sgen
C:\>ssi uf3
C:\>sdfa task/al 32768-p
C:\>pio (Defines a VAL.)
C:\>demo
C:\>srpl
```

- Deleting VALs

```
C:\>dpio (Deletes a VAL.)
```

Contents of the file

PIO.BAT

```
sdfv XW 0xE0000 ┐
                ├-------------- Defines a VAL.
sdfv FW 0xE2000 ┘
```

DEMO.C

```
extern short XW_v [0x100];
extern short FW_v [0x100];
main()
{
   short w;
   w=XW_v [0]+XW_v [1];
   FW_v [0]=w;
}
```

DPIO.BAT

```
sdlv XW ┐
        ├-------------- Deletes a VAL.
sdlv FW ┘
```