HITACHI

Software Manual

CPMS General Description and Macro Specifications

S10VE

SEE-3-201 (A)

Software Manual

CPMS General Description and Macro Specifications

S10VE

First Edition, May 2020, SEE-3-201 (A)

Printed in Japan.

IC (FL-MW2007)

SAFETY PRECAUTIONS

- Before building a system, creating or program, or otherwise using this product, carefully read the contents of this manual to ensure that you have a sufficient understanding of the instructions and warnings herein. Incorrect operations might result in system failure.
- Keep this manual in a readily accessible place so that users can refer to it quickly as needed.
- If you are uncertain about any of the information contained in this manual or if any information is unclear, contact our nearest company location or your SE.
- Hitachi is not responsible for accidents or damage caused by incorrect customer use of the product.
- Hitachi is not responsible for accidents or damage caused by use of the product after the customer has modified the Hitachi-provided software.
- Hitachi is not responsible for product reliability in cases where software other than that provided by Hitachi is used.
- Include file backups as part of your daily operations. The contents of files might be lost because of a failure in a file device, a power failure that occurs while files are being accessed, incorrect operations, or other reasons. Back up files regularly as a safeguard against such situations.
- To ensure sufficient safety with respect to the system being used even in cases of product failure, incorrect operations, or program defects, design your system so that the protective and safety circuits are located externally, and implement sufficient safety measures with respect to injuries and major disasters.
- Configure the system so that the emergency stop circuit, interlock circuit, and other such circuits are outside the PLC. PLC failure might result in product failure or accidents.
- Sufficiently verify safety before performing such actions as modifications to a program that is operating, forced output, or a RUN or STOP instruction. Incorrect operations might result in product failure or accidents.
- Safety precautions in this manual are classified into four levels according to the severity of potential hazards: DANGER, WARNING, CAUTION, and NOTICE.

⚠ DANGER	: Indicates a hazardous situation which, if not avoided, will result in death or serious injury.
⚠ WARNING	: Indicates a hazardous situation which, if not avoided, could result in death or serious injury.
⚠ CAUTION	: Indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.
NOTICE	: Indicates a danger (resulting from incorrect use of the product) that can cause property damage or loss not related to personal injury if the safety precautions are not observed.

Failure to observe precautions marked with ⚠CAUTION or *NOTICE* could also lead to a serious consequence depending on the situation in which the product is used. Therefore, you must observe all of those precautions without fail.

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

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

Minor or moderate injury: Is an injury that does not require either hospitalization for medical treatment or long-term follow-up care. Examples of minor or moderate injuries include burns and electric-shock injuries.

Property damage or loss not related to personal injury: Is damage other than personal injury. Examples of property damage or loss not related to personal injury are as follows: damage or loss of personal property, failure or damage of the main unit of the product, and loss of data.

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

NOTICE

- If CPES or PCKS occurs in a system task (number 225 to 300), do not turn on the ULSUB_OUT_ABORTSUPRES bit in the output information. Doing so might result in unexpected operation, and the system might go down.
(See page 1-58.)

- Make sure that you initialize the ECB allocated in the GLB to `0` before use.
(See page 2-11.)

- The `asusp` macro locks the CPU, but this does not guarantee the locking of other resources.
  That is, if a conflict occurs with another task when locking resources by using the `rserv` macro, a deadlock occurs. After the `asusp` macro is issued, do not issue any process or macro that could cause a conflict for resources.
  If you do not minimize the time for which the `asusp` macro is active, system operation might be adversely affected. Do not issue any other macros after issuing the `asusp` macro until the `arsum` macro is issued.
(See page 2-15.)

- Depending on the specified address, the `wrtmem` macro might destroy the program. The CPMS cannot prevent the macro from destroying the program.
(See page 2-20.)

- Accessing MRAM repeatedly might cause the CPU to stop working properly. When copying MRAM, make sure you use `MRAMmemcpy()` instead of the standard `memcpy()`.
(See page 2-23.)

- Do not issue the delay macro while shared resources are locked, or system operation might be adversely affected.
(See page 2-27.)

NOTICE

- The time at which to generate timer events registered by using the `timer` macro is sometimes changed. The following table shows when this occurs.

Timer event type	When time is delayed	When time is advanced	Remarks
Duration-based and duration-cycle-based	The time of the timer event is not affected.	The time of the timer event is not affected.	When the duration specified by the `timer` macro has elapsed, the timer event is generated.
Time-based and time-cycle-based	When the time at which to generate the timer event is delayed by 24 hours or more, the time is registered as the same time on the following day.	If the first scheduled start time is skipped, making the task unable to start, the task starts when the time is changed. The scheduled start time is changed to the time the change was made or later, by adding the cycle time to the first scheduled start time.	

(See page 2-29.)

- When resources are locked by using the `prsrv` macro, no resource can be locked by using the `rserv` macro.

(See page 2-32.)

- During the return of control when using the `resume_env` macro to issue the `save_env` macro, if the BSS or GLB data related to control of the user stack or task differs from that when the `save_env` macro was issued, issuing the `resume_env` macro might not result in the same processing as the last time the environment was in operation.

(See page 2-46, 2-47.)

- The `resume_env` macro must be issued from the CPES built-in subroutine.
- If the `resume_env` macro is issued from somewhere other than the CPES built-in subroutine, the `resume_env` macro does nothing.
- If the `env` parameter is invalid, the CPU might go down.

(See page 2-47.)

NOTICE

- The time base is located in the device. In the future, the time base might have to be handled differently due to differences in the device or operating frequency.

(See page 2-48.)

- This exclusive control is effective only among the processes of the local processor, and is not used for exclusive control among other processors or I/O DMA.

(See page 2-50.)

- `fpsetsticky()` changes the values of all sticky flags' corresponding FPU exception flag fields.
  `fpsetmask()` changes all exception mask values' corresponding exception enable bits.
  The following modes are not available for rounding control by using `fpgetround()` and `fpsetround()`:
  • FP_RP: Negative values are truncated and positive values are rounded up (round to plus)
  • FP_RM: Positive values are truncated and negative values are rounded up (round to minus)

(See page 3-5.)

Revision History

Revision No.	History (revision details)	Issue date	Remarks
A	First edition	May 2020	

PREFACE

This manual focuses on the functions and macro call linkage specifications of the CPMS (Compact Process Monitor System), an operating system for real-time control of the S10VE system. Read this manual if you are designing or developing a real-time control program that uses the CPMS. Note that this manual is written for readers who are familiar with the basics of general operating systems.

< Organization of this manual >

PART 1 GENERAL DESCRIPTION

CHAPTER 1 OVERVIEW
This chapter explains the structure of the CPMS and the specifications of its basic functions.

CHAPTER 2 TASK MANAGEMENT
This chapter explains the functions of tasks needed for creating real-time control programs, including task structure and scheduling.

CHAPTER 3 MEMORY MANAGEMENT
This chapter explains memory management functions such as main memory allocation and protection.

CHAPTER 4 TIMER MANAGEMENT
This chapter explains the methods of managing times and durations.

CHAPTER 5 SHARED RESOURCE MANAGEMENT
This chapter explains the exclusive control of resources that are shared among tasks.

CHAPTER 6 I/O DEVICE MANAGEMENT
This chapter gives information such as how to identify I/O devices.

CHAPTER 7 SYSTEM MANAGEMENT
This chapter explains how to start up this system and perform HP scheduling.

CHAPTER 8 TASK ERROR HANDLING
This chapter gives information such as on built-in subroutines that are executed when a task error occurs.

CHAPTER 9 SYSTEM SERVICES
This chapter explains the functions for getting operating information about systems and tasks.

CHAPTER 10 NOTES ON MIGRATING FROM THE S10V
This chapter provides notes on performing migration from the S10V.

CHAPTER 11 OPTIONAL MODULES
This chapter explains the settings and error logs for optional modules.

CHAPTER 12 REMOTE I/O
This chapter explains the settings information and system operation when using remote I/O.

PART 2 MACRO SPECIFICATIONS

This part explains the macro call functions provided by the CPMS and their linkage specifications.

PART 3 LIBRARIES

This part explains the library functions such as arithmetic operations and their linkage specifications.

<Relevant manuals>

Manual title	Manual no.
S10VE User's Manual General Description	SEE-1-001
S10VE Software Manual Operation RPDP for Windows®	SEE-3-133
S10VE Software Manual Operation Ladder Diagram System for Windows®	SEE-3-131
S10VE Software Manual Programming Ladder Diagram System for Windows®	SEE-3-121
S10VE Software Manual Operation HI-FLOW for Windows®	SEE-3-132
S10VE Software Manual Programming HI-FLOW for Windows®	SEE-3-122

<Note for storage capacity calculations>

- Memory capacities and requirements, file sizes and storage requirements 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 equal signs).
  1 KB (kilobyte) = 1,024 bytes
  1 MB (megabyte) = 1,048,576 bytes
  1 GB (gigabyte) = 1,073,741,824 bytes
  1 TB (terabyte) = 1,099,511,627,776 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) = 1,000 bytes
  1 MB (megabyte) = $1{,}000^2$ bytes
  1 GB (gigabyte) = $1{,}000^3$ bytes
  1 TB (terabyte) = $1{,}000^4$ bytes

CONTENTS

PART 1 GENERAL DESCRIPTION

PART 3 LIBRARIES

APPENDIXES

FIGURES

TABLES

This Page Intentionally Left Blank

PART 1 GENERAL DESCRIPTION

CHAPTER 1 OVERVIEW

1.1 CPMS Functions

The CPMS (Compact Process Monitor System) is the nucleus of the real-time operating system.

The CPMS has the following functions:

- Task management
  This controls the multitasking of up to 300 tasks.
- Memory management
  This controls memory address conversion and memory protection.
- Timer management
  This controls times and durations in the system.
- Shared resource management
  This exclusively controls resources shared across tasks.
- I/O device management
  This manages a variety of I/O devices, and incorporates I/O drivers into the system.
- System management
  This controls system initialization as well as the state and configuration of the system.
- System services
  This provides information and services belonging to the system.

1.2 CPMS Specifications

Table 1-1 shows the CPMS specifications (system parameters).

Table 1-1 CPMS Specifications

Item	Value	Remarks
Number of tasks	Up to 300	Assign task numbers according to the following specifications: 1 to 224: User tasks 225 to 300: System tasks
Task priority	32 levels	Users: 4 to 27 System: 0 to 31
Number of timers	Number of tasks + 32	Used by the `TIMER` and `DELAY` macros
Number of resources that can be allocated concurrently	Up to 32	Used by the `RSERV` and `PRSRV` macros
DHP buffer	128 KB	12 to 32 bytes per case
Error log buffer	32 KB	1 KB per case
Built-in subroutine	10 points	4 entries per point

1.3 CPMS Structure

The CPMS consists of an exception processing program, dispatcher, and system tasks as shown in Figure 1-1.

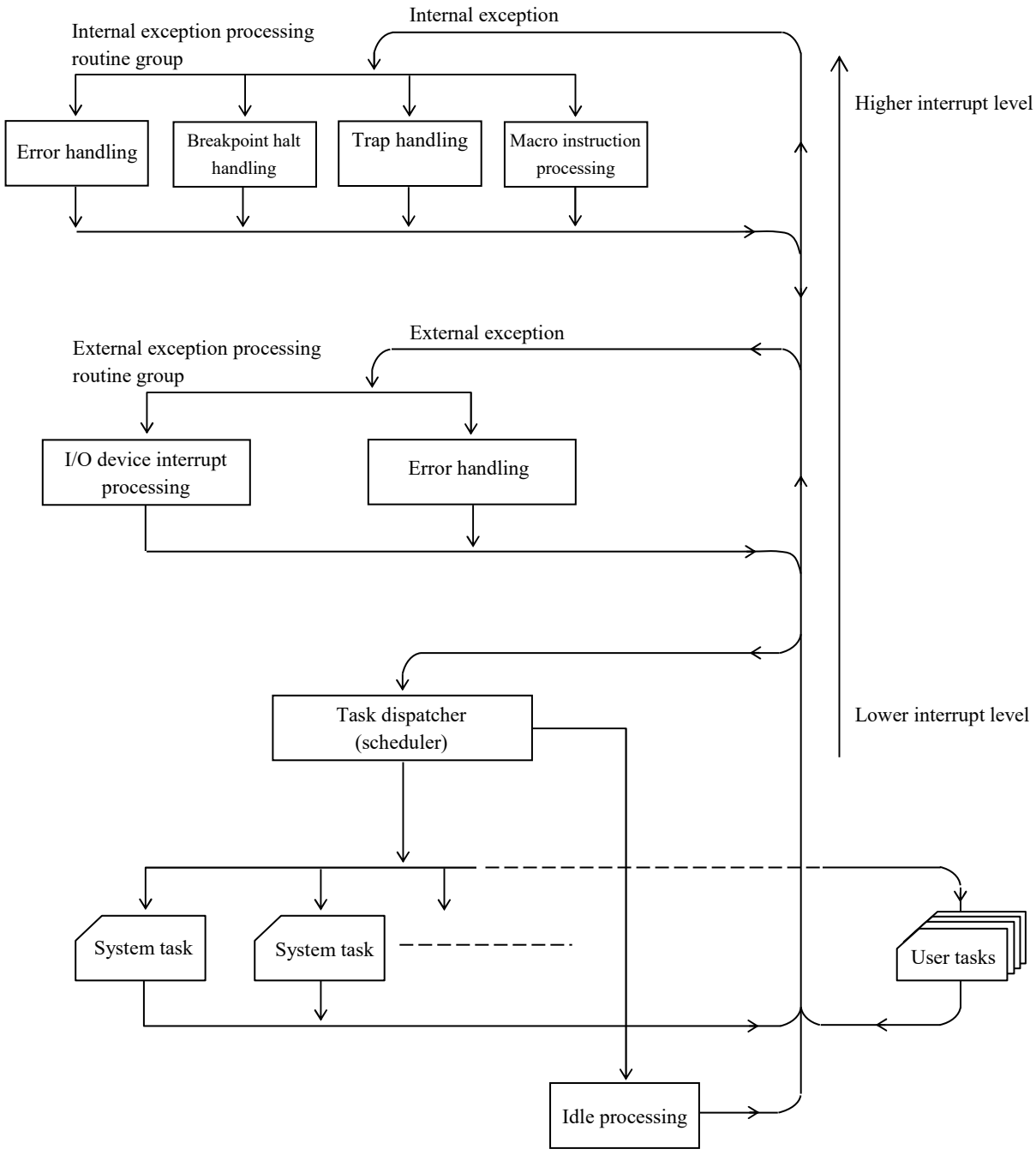


Figure 1-1 CPMS Structure

1.4 CPMS and Hardware

Figure 1-2 shows the relationships between the S10VE hardware configuration and the CPMS.

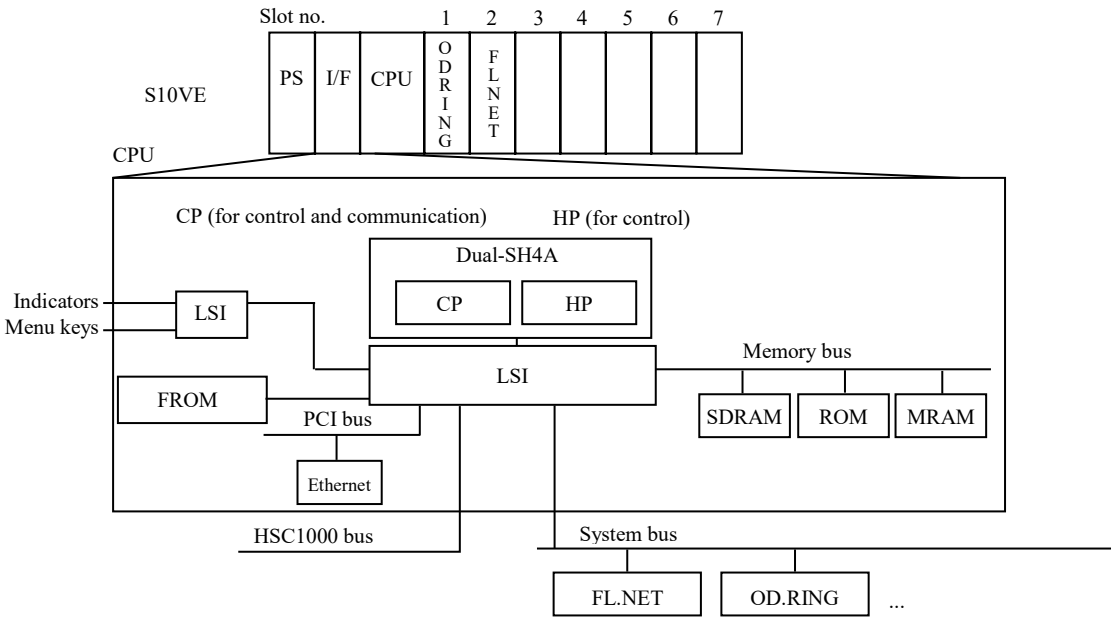


Figure 1-2 Relationship Between the Hardware Configuration and the CPMS

- Processor (CPU)

  The S10VE uses an SH4A dual processor as its CPU. Core 0 of the dual processor is used as a CP (communication processor) for control and communication purposes, and core 1 as an HP (high-speed processor). By using one core for control and communication processing and the other for control processing, and assigning control over communications to Core 0 (CP), the S10VE can distribute load across cores and improve the performance of control programs such as PI/O access running on Core 1 (HP).

  HP (High-speed Processor):

  The HP runs control programs. A control program is a program that periodically accesses PI/O using ladder programs and HI-FLOW programs and performs calculations and the like for control purposes. A program running on the HP can also be used to transmit and receive data. To do so, use the ladder Ether instructions. Because this might affect the control of ladders and HI-FLOW, do not register user tasks to the HP.

  CP (Communication Processor):

  The CP runs communication processing programs and control programs. Examples of communication processing programs include system tasks for communication processing purposes provided by subsystems, tool connection servers, and system tasks that execute ladder Ether instructions.

  A control program is a program that periodically accesses PI/O over the memory interface and performs calculations and the like for control purposes, for example, C-mode tasks for application purposes.

- LSI
  The LSI controls memory access and bus access from the processors.
- Memory bus, memory (SDRAM, ROM, MRAM (non-volatile memory), and FROM (flash memory))
  These include the main memory connected to the memory bus on the CPU (SDRAM), ROM, and MRAM. In addition, FROM is available for backing up SDRAM.
  SDRAM: SDRAM is the main memory of S10VE. The OS, programs, and data are stored here.
  ROM: Programs such as the OS startup control program are stored here.
  MRAM: MRAM is non-volatile memory that stores information such as settings. Its contents are retained even while the power is off.
  FROM: FROM is used to back up SDRAM. You can start the system even when data in SDRAM has been lost, by copying data from FROM to SDRAM.
  The CPMS information is downloaded through a tool and stored in FROM.
  Indicators: These display a variety of information.
  Menu keys: These are used for switching among content displayed on the indicators.
- I/O buses
  The S10VE has a system bus and PCI bus, which serve as LSI-controlled I/O buses on the CPU module.
  System bus: This bus is for connecting to optional modules such as the OD.RING and FL.NET modules.
  PCI bus: This bus is for connecting to Ethernet built into the CPU.
  HSC1000 bus: This bus is for controlling RI/O.

1.5 Interface Between the CPMS and Users

The interface between the CPMS and users facilitates interaction with operations from the RPDP (Real-time Program Development Package), macro instructions from user tasks, built-in subroutines, and more.

The RPDP provides an environment for creating tasks and built-in subroutines for the CP in the CPU.

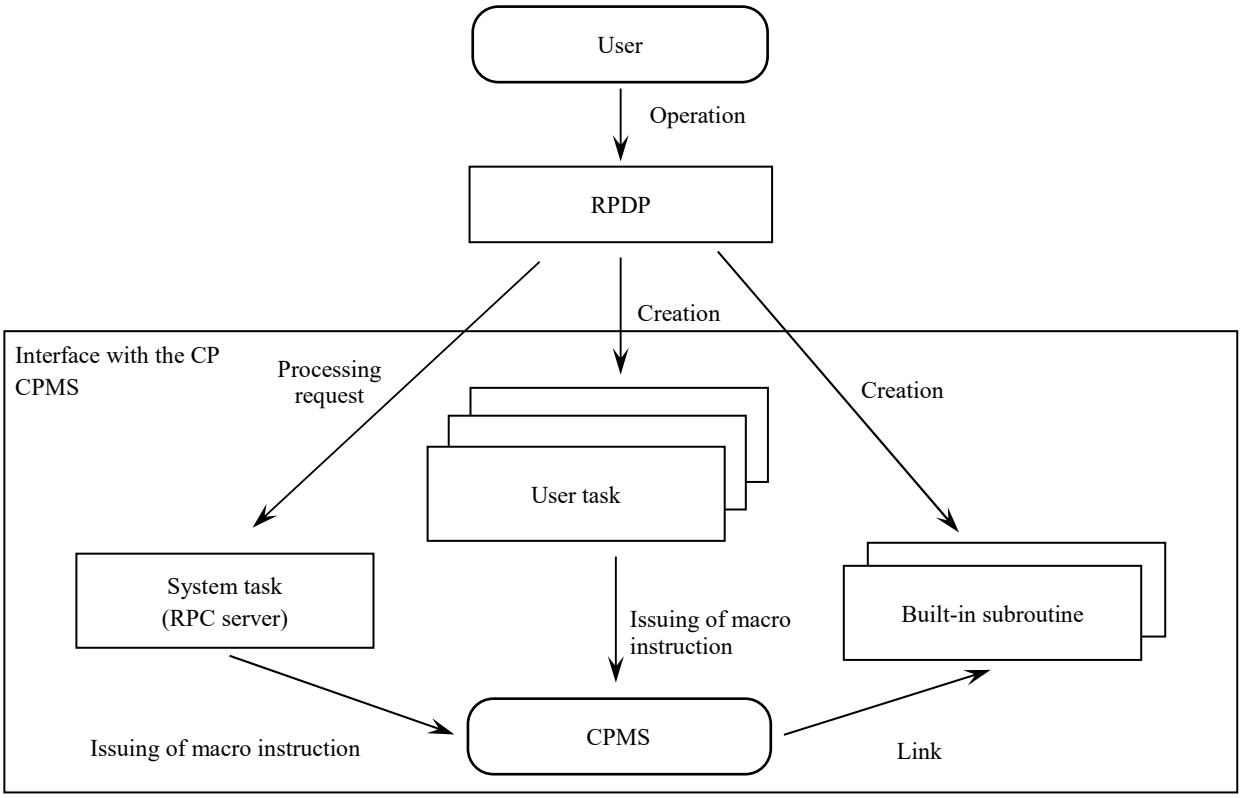


Figure 1-3 Interface Between the CPMS and Users

CHAPTER 2 TASK MANAGEMENT

2.1 Tasks

A task is a unit of work done by the system when executing a program. The CPMS manages program execution and resource allocation on a per-task basis.

(1) Task numbers

A task number (TN) is used to identify a task. The CPMS can manage up to 300 tasks. For this purpose, the CPMS has a task management table that can accommodate 300 tasks. Users can assign task numbers 1 to 224 to user tasks. Task numbers 225 to 300 are assigned to system tasks.

Upon startup, the CPMS starts task number 1 as the initial start task.

(2) Task structure

A task consists of following areas: TEXT, DATA, BSS, STACK, and OS work.

TEXT: This is the program execution area. This area is write-protected.

DATA: This is a data area with default values. This area is write-protected.

BSS: This is a data area with no default values.

STACK: This is the work data area used for program execution, and its addresses are used in descending order.

OS work: This is a work data area used when the CPMS executes macros.

Figure 1-4 Task Structure

You can create multitasks that share TEXT, DATA, and BSS areas. In a multitask, each task has its own STACK area. However, note that the BSS area is shared.

(3) Task types

There are two types of tasks: user tasks created by users, and system tasks provided by the system. Task numbers 225 to 300 are reserved for system tasks. Task numbers 1 to 224 are assigned to user tasks.

(4) Initial start task

Task number 1 is the user initial start task (UIST). The CPMS starts the user initial start task. For other user tasks, the user must create the user initial start task so that it starts the other user tasks.

Task number 255 is the system initial start task (SIST). The CPMS starts the system initial start task before starting the user system initial start task. The purpose of the system initial start task is to start the other system tasks.

The CPMS passes a start factor as an initiation factor for the initial start task. Load the initiation factor for the initial start task by using the `GFACT` macro. For information about the start factor, see 7.2 INS Built-in Subroutine and the Initial Start Task.

(5) Task priority levels

When multiple tasks have requested to use a shared resource (CPU and memory) in the system, the task given the right to use the resource is determined based the tasks' processing priority. This processing priority is called the *priority level* or *level*. Each level is represented by a numerical value from 0 to 31, where the smaller the value, the higher the priority. Levels 4 to 27 are available to users. When a task is registered, its level is specified. This level is called the *original level* of the task. Usually, when a task is started, this original level becomes its level during operation (current execution level). The order of resource allocation is determined according to this current execution level. A priority level is specified for a task when the task is registered. Figure 1-5 shows the relationship between levels that can be assigned to system tasks and those that can be assigned to user tasks.

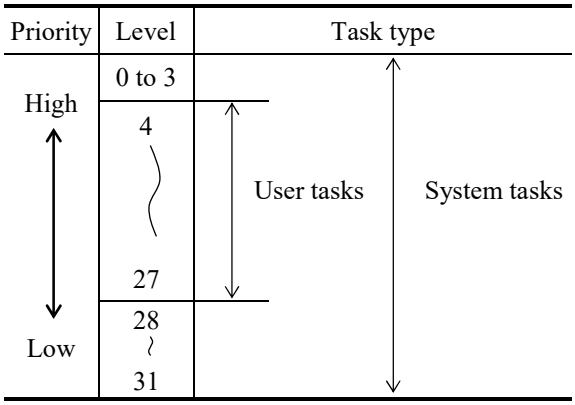
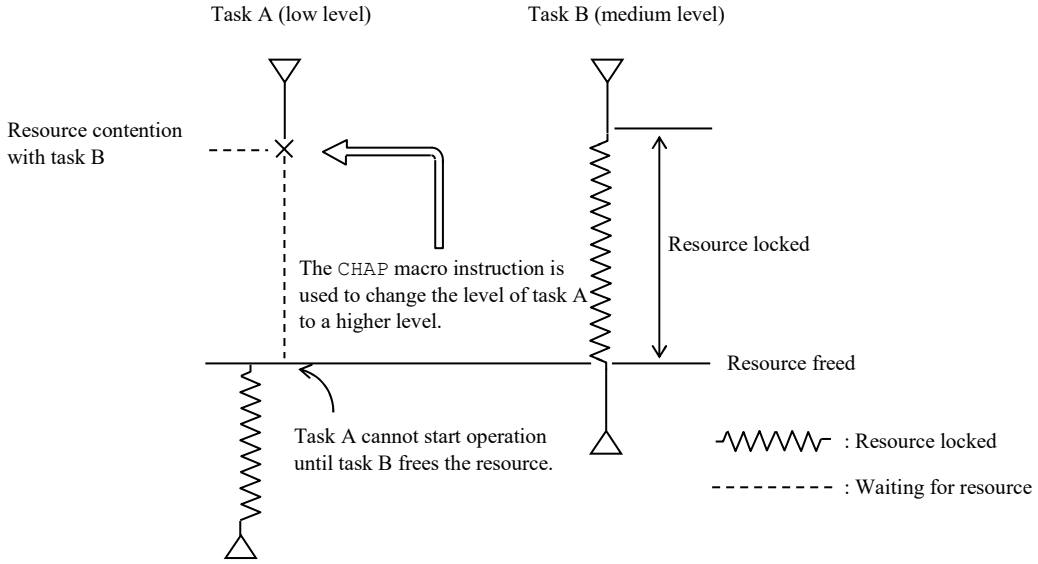


Figure 1-5 Relationship Between Task Levels and Types

(6) Changing priority level

You can use the CHAP macro instruction to change the level of a task while the task is being executed. The CHAP macro instruction takes effect from the start to the end of the target task's operation. When the task ends, its original level is restored. If you use the CHAP macro instruction to change a task's level before the task starts operation, the new level takes effect as the operation's priority level from the start of operation. However, if task operation is aborted at any time between the level change and the start of operation, the macro instruction loses its effect.

The CHAP macro instruction changes the priority level used as a criterion for allocating various resources. A resource already allocated to a task is not forcibly assigned to a task changed to have a high priority level. This is shown in Figure 1-6.



Note: When task A has a resource contention with task B and is put in the WAIT state, the resource locked by task B is not allocated to task A even if the CHAP macro instruction is used to raise task A's priority level.

Figure 1-6 Resource Allocation when Priority Level is Changed

2.2 Task Scheduling

(1) Scheduling algorithm

When start requests are issued by multiple tasks in a running system, multiple tasks contend for the right to use the CPU.

There is only one CPU in the system. Therefore, only one task can ever be serviced by the CPU at a time. The selection of one task among many contenders to use the CPU is called a *dispatch*. The method of dispatching tasks is called *task scheduling*.

While there are various different scheduling algorithms, the CPMS adopts a fixed priority scheduling method that dispatches tasks in order of priority. Among tasks of the same level, this method employs the FCFS (first come first served) algorithm.

In FCFS, tasks that have transmitted a start request are linked to the CPU queue in the order the start requests were received. As shown in Figure 1-7, blocks of memory for managing tasks called TCBs (task control blocks) are linked to the CPU queue. One TCB is assigned per task.

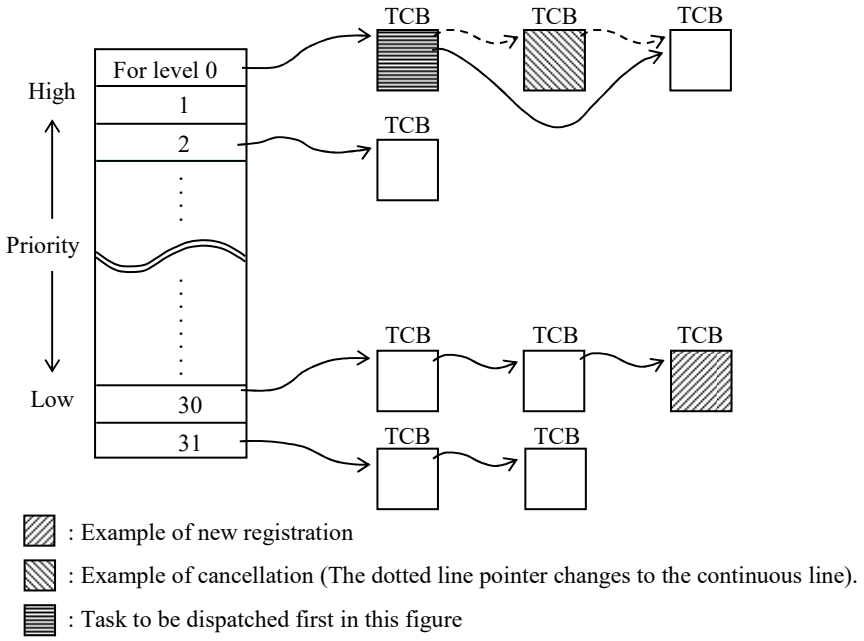
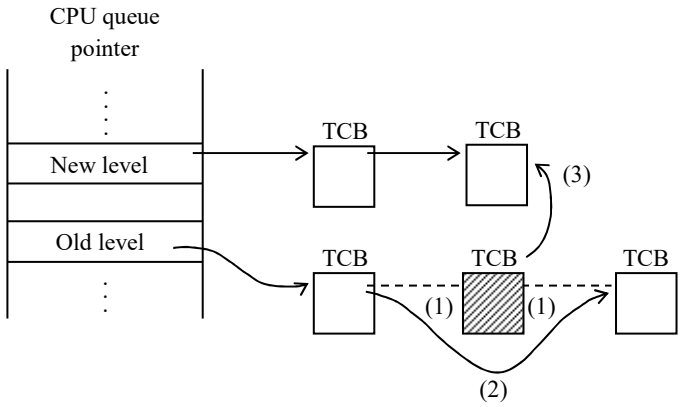


Figure 1-7 CPU Queue

A task is released from the CPU queue in the following cases:

- When the task issues an `EXIT` macro instruction
- When the task is specified by another task's `ABORT` macro instruction (prohibited)
- When an error occurs in the task (for example, when a task tries to access data without permission, causing a protection error). When an error occurs in a task, the CPMS aborts the task.

(2) Behavior of the CPU queue when priority level changes
Figure 1-8 shows how the TCB of a target task is handled in the CPU queue, when a CHAP macro instruction is issued for that task.



Procedure
(1) The specified TCB is released from the old level queue.
(2) The old level queue links to another TCB.
(3) The released TCB is linked to the end of the queue for its specified level.

Note: When a level is changed by using a CHAP macro instruction, the specified TCB is linked to the end of the queue for its specified level according to the FCFS algorithm.

Figure 1-8 Changing a Level

(3) Multitasking
In task management, processing is performed to maximize utilization of the CPU. For example, when an in-progress task is suspended for some reason, the next task in the CPU queue is dispatched immediately. The dispatched task then starts operation. If the suspended task becomes able to resume operation during execution of this new task, the dispatcher resumes the suspended task. Figure 1-9 shows an example of this. Although only one task is ever operating at one time, from a macro perspective, it appears as if tasks A, B, C, and so on are operating simultaneously. This is how multiple tasks are generally processed. This way of processing tasks is called *parallel processing*. Parallel processing allows the CPU to be used more efficiently.

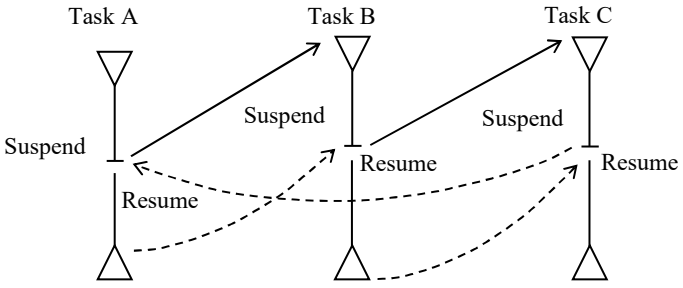


Figure 1-9 Concurrent Task Processing (Multitasking)

2.3 Task Operation

Tasks generally have a life cycle. That is, tasks are generated, started, executed, suspended, resumed, ended, and ultimately eliminated. However, for real-time tasks, the overhead from when a start request is issued until the task is executed is an important factor in determining response speed. Accordingly, it is necessary to minimize incidences of generation and destruction. Therefore, instead of generating a task when a start request is received, tasks need to be created and registered to the CPMS in advance. That is, when a real-time task is started, the target task does not need to be generated anew. Only a start request (`QUEUE` macro instruction) needs to be issued. Also note that the task is not eliminated after its operation ends.

Table 1-2 lists events (factors) that trigger the starting of a task.

Table 1-3 lists task execution conditions (initial activation). After a task is started, it is executed when all conditions shown in Table 1-3 are met.

Tasks executed in this way continue operation until some reason or interruption prevents further processing and a higher-level task must be run. When all necessary processing is completed (program execution ends), operation of the task ends. This is called *task suspension* and *ending of a task* (abortion).

Table 1-4 shows the conditions for suspending a task. A suspended task resumes operation when the cause of the suspension is eliminated, and no higher-level tasks or same-level tasks that started earlier can be operated. This is called *task resumption*.

Table 1-5 shows the conditions for resuming a task, and Table 1-6 shows the conditions for ending a task.

Table 1-2 Task Initiation Factors

Event		Description
Internal factor	A `QUEUE` macro instruction is issued.	When a task issues a `QUEUE` instruction, the task specified in the instruction parameters is started.
	A certain duration elapses, or a certain time is reached.	If a `TIMER` macro instruction has been issued, the task specified in the instruction parameters is started after the specified duration or at the specified time.
External factor	An attention interrupt occurs from an I/O device.	A task that is registered in a built-in subroutine is started by an attention interrupt from an I/O device.

Table 1-3 Conditions for Executing a Task (Start of Initialization)

Condition	Description
All higher-level tasks and same-level tasks that started earlier cannot be operated.	When a higher-priority task is operable, it is executed.
The task's main program is loaded on the main memory.	Unless the program is loaded on the main memory, it cannot be operated.
Execution of the task itself is not suppressed.	When execution is suppressed by the `SUSP` or `ASUSP` macro instruction, the task is not executed.

When all the conditions shown in Table 1-4 are met, the task is executed.

Table 1-4 Conditions for Suspending a Task

Condition	Description
A higher-priority task is started.	When a higher-priority task is started by an interrupt (process interrupt, timer) and is operable, control is transferred to that task.
The suppression of execution of a higher-priority task is canceled.	When a higher-priority task (whose execution had been suppressed) becomes operable, control is transferred to that task.
Execution is suspended by the task itself.	When execution is suspended by the task itself, such as for synchronization, control is transferred to another task.

When any one of the conditions shown in this table is met, the task is suspended.

Table 1-5 Conditions for Resuming a Task

Condition	Description
The suppression of execution by another task is canceled.	The suppression of execution by the SUSP or ASUSP macro instruction is canceled.
An event the system has been waiting for occurs.	An event that eliminates the cause of the suspension (DELAY or WAIT) occurs.
A higher-level task or same-level task that started earlier ends or is suspended.	CPU service cannot be rendered to this task as long as a higher-level task or same-level task that started earlier is operable.

Table 1-6 Conditions for Ending a Task

Condition	Description
The EXIT macro instruction is issued.	Usually, task processing is ended by using the EXIT macro instruction.
The task is targeted by an ABORT macro instruction.	Processing is aborted by using the ABORT macro instruction.
Processing cannot continue due to a program error or similar event.	The CPMS automatically aborts the task in which an error occurred.

When any one of the conditions shown in Table 1-6 is met, the task ends operation.

2.4 Task State Transition

In the CPMS system, multiple real-time tasks are interlinked and operated to perform the functions of the entire system. Individual tasks continue operation in close coordination with each other, starting, suspending, restarting, and ending each other repeatedly as described in section 2.3.

Data is exchanged between tasks by using the GLB (global data area), a data area common to all tasks.

Control exchange between tasks is performed by using macro instructions provided by task management.

Task management macro instructions control task operation by effecting task state transitions (changes). Design systems and programs with a correct understanding of how to effect task state transitions to ensure efficient and correct system operation, and of what macro instructions cause state transitions.

Table 1-7 shows the task states. Figure 1-10 shows the relationship between macro instructions that control task execution and states, and task states.

Note that in the state transitions shown in Figure 1-10, the RUNNING state includes those times when the task is suspended. The target task states of these macro instructions are only examples, and do not represent all cases.

Table 1-7 Task States

State	Designation	Description
The task is currently being executed.	RUNNING	The CPU is locked for task execution.
The task is awaiting execution.	RUNNABLE	The task is waiting for the CPU to be unlocked.
Execution is suppressed.	SUSPENDED	Execution of the task is suppressed.
The task is awaiting an event.	WAIT	The task is waiting for an event.
The task is awaiting startup.	IDLE	The task is waiting to be started.
Startup is suppressed.	DORMANT	Task startup is suppressed.
The task is unregistered.	NON-EXISTENT	The task is not registered in the CPMS.

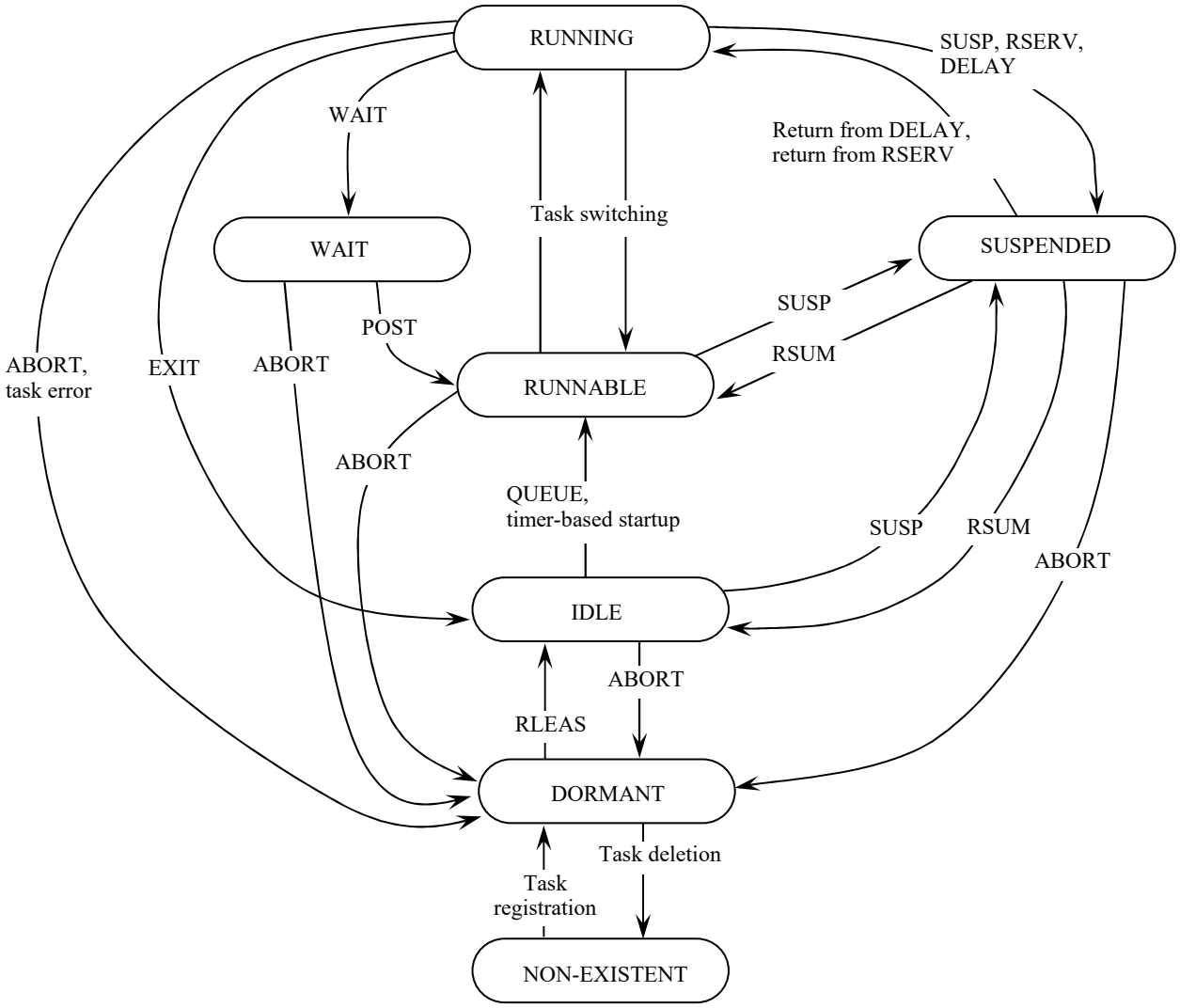


Figure 1-10 Task State Transitions

Note: If a `SUSP` or `RSUM` macro is issued on a non-queued task that is in the IDLE state, execution is suppressed or suppression of execution is canceled accordingly. However, the task remains in the IDLE state, and information on the suppression of execution is stored in the tc-flag field of the task management table TCB.

2.5 Task Control

The following subsections use examples to explain the task control method.

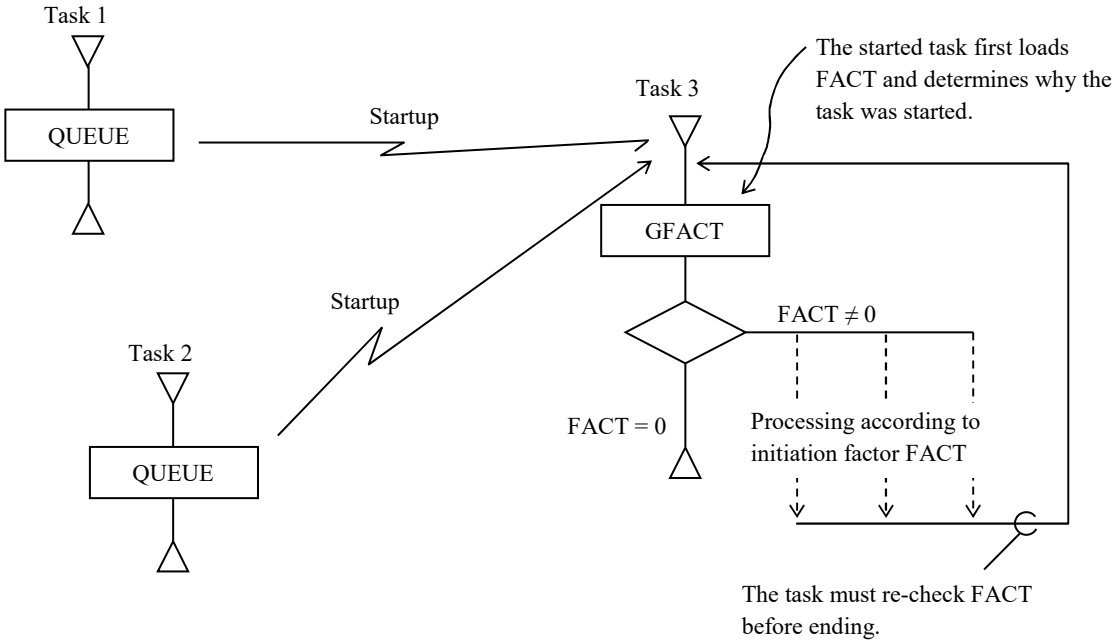
2.5.1 Initial state

When the system starts (when the power is on and the processing unit starts to operate), all user tasks except the initial start task are in the DORMANT state.
The initial start task is automatically started by the CPMS when the system starts. The initial start task puts tasks required for job execution into the IDLE state by using the `RLEAS` macro instruction. (This is called *releasing the task*.) This state means the task is ready to receive a start request.

2.5.2 Starting tasks

- `QUEUE` macro instruction

  Tasks are started by using the `QUEUE` macro instruction. A started task loads the initiation factor (FACT) by using the `GFACT` macro instruction in order to determine what factor caused the task to start. Figure 1-11 shows an example of this.



Task 3 can use the GFACT macro instruction to determine what task (task 1 or task 2) started task 3. That is, if a different FACT is specified for when task 1 starts task 3 from when task 2 starts task 3, it is possible to tell which task activated task 3.

Figure 1-11 Starting Tasks

In Figure 1-11, the GFACT macro instruction loads initiation factors one by one. For example, supposing that the four initiation factors (integers 1 to 32) 1, 5, 10, and 11 are set, the GFACT macro instruction loads the integers in sequence starting from the smallest number. The first time the GFACT macro instruction is issued, FACT = 1 is loaded. The second time the GFACT macro instruction is issued, FACT = 5 is loaded. Once a FACT is loaded, it is cleared to 0 by the GFACT macro instruction. Accordingly, after FACT = 1 is loaded, it is not loaded again even if the GFACT macro instruction is issued again.
Such FACTs can also be set by using the SFACT macro instruction. Figure 1-12 shows an example of this.

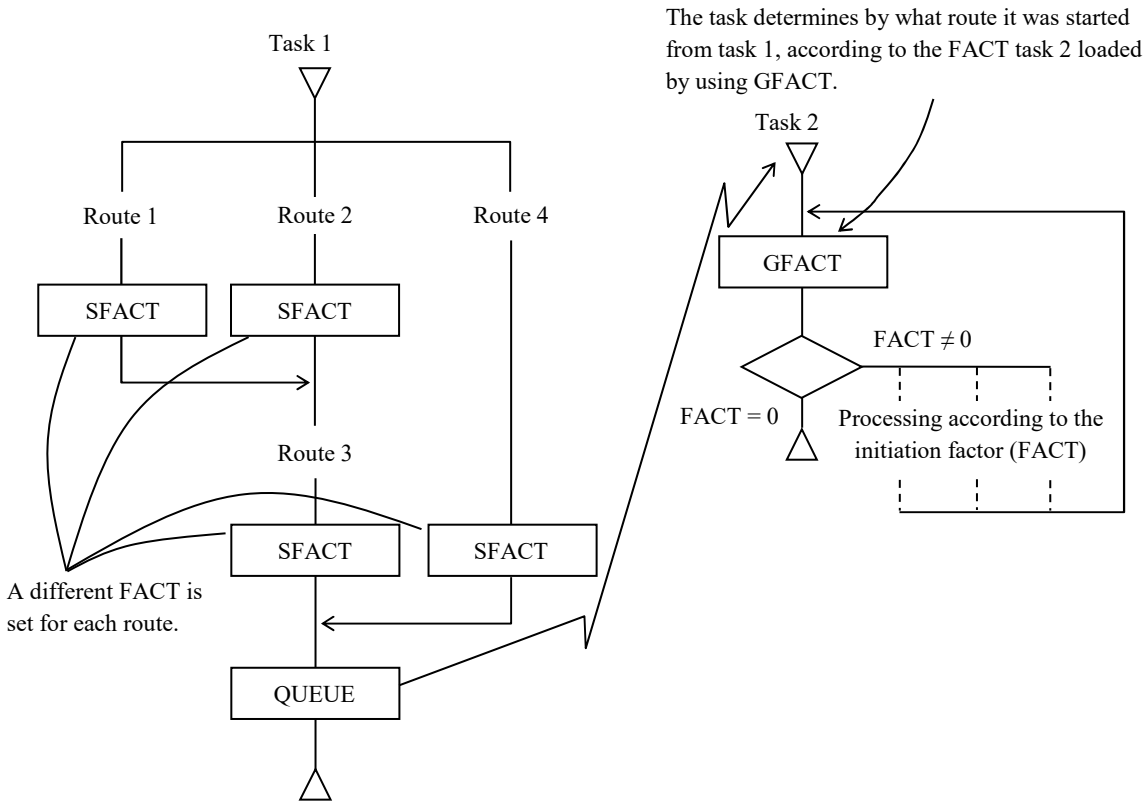
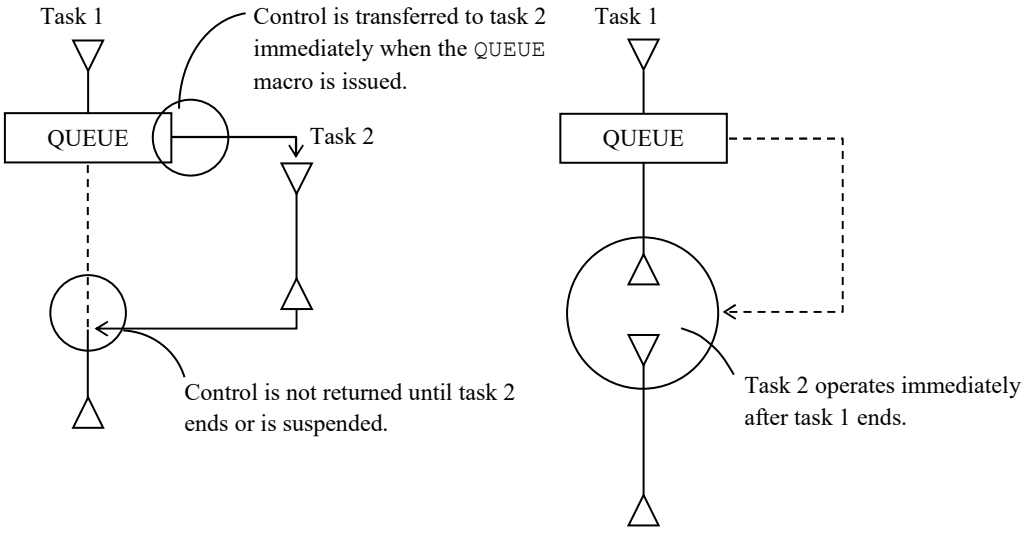


Figure 1-12 SFACT Macro Instruction

As explained earlier, the *first come first served* rule is applied to same-level tasks in the CPMS in order to perform efficient real-time control. Accordingly, the order in which tasks are executed depends on the relationship between task levels during startup as shown in Figure 1-13. This is an important key to understanding task scheduling in the CPMS.



When task 2 has a higher priority level than task 1

When task 1 has the same priority level as task 2, or a higher priority level than task 2

When task 1 starts task 2 by using the QUEUE macro instruction, the flow of program execution control differs depending on the relationship between the task levels.

Figure 1-13 QUEUE Macro Instruction and Task Execution Order

- TIMER macro instruction

  As seen in Figure 1-13, task startup by using the QUEUE macro instruction is usually performed immediately. In some cases, however, a task must be started after a certain duration has elapsed, or at a certain time. In these cases, use the TIMER macro instruction. This macro instruction can be used to start a specified task at a time or after a duration specified in the parameters. At this time, the initiation factor (FACT) is transferred to the started task in the exact same way as FACTs transferred by using the QUEUE macro instruction.

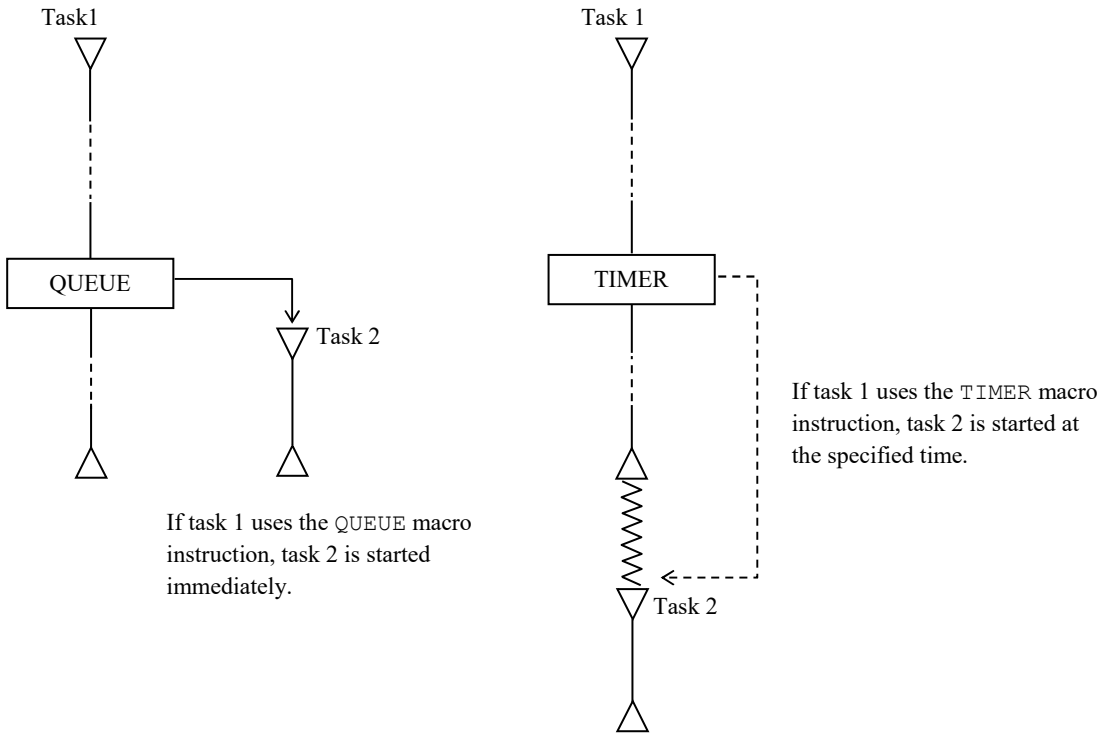


Figure 1-14 Difference Between Task Startup by Using the QUEUE and TIMER Macro Instructions

2.5.3 Ending tasks

A task ends itself by issuing an `EXIT` macro instruction. In the CPMS, tasks are allowed to issue the `EXIT` macro instruction even when returning from the main routine.

2.5.4 Suppressing task execution

- DELAY macro instruction

  The TIMER macro instruction is used mainly to start another task after a certain duration has elapsed. You can also have a task issue itself a TIMER macro instruction to operate after a certain duration. If the DELAY macro instruction is used, the environment (for example, BSS and STACK value) used at the time the DELAY macro instruction was issued can be saved when control is returned to the task itself after the duration specified in the parameters has elapsed. If the TIMER macro instruction is used, operation starts from the beginning of the task. However, the environment is not saved. For this reason, to resume operation after a certain duration of suspension, use the DELAY macro instruction.

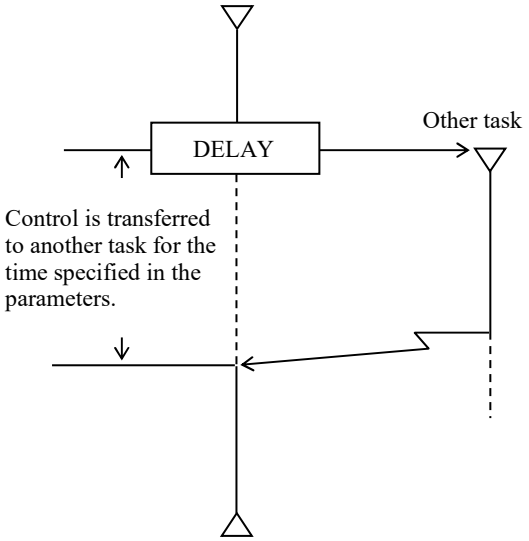


Figure 1-15 DELAY Macro Instruction

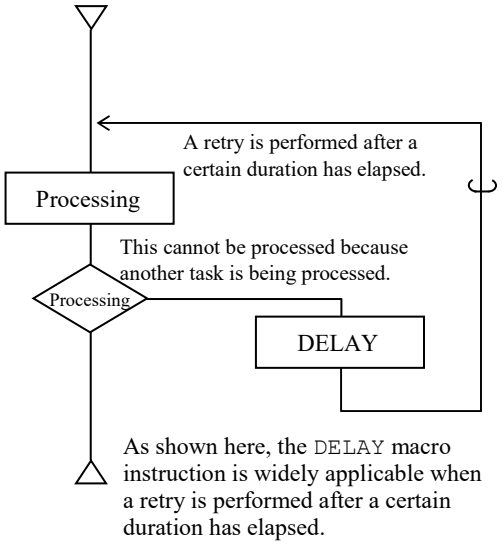


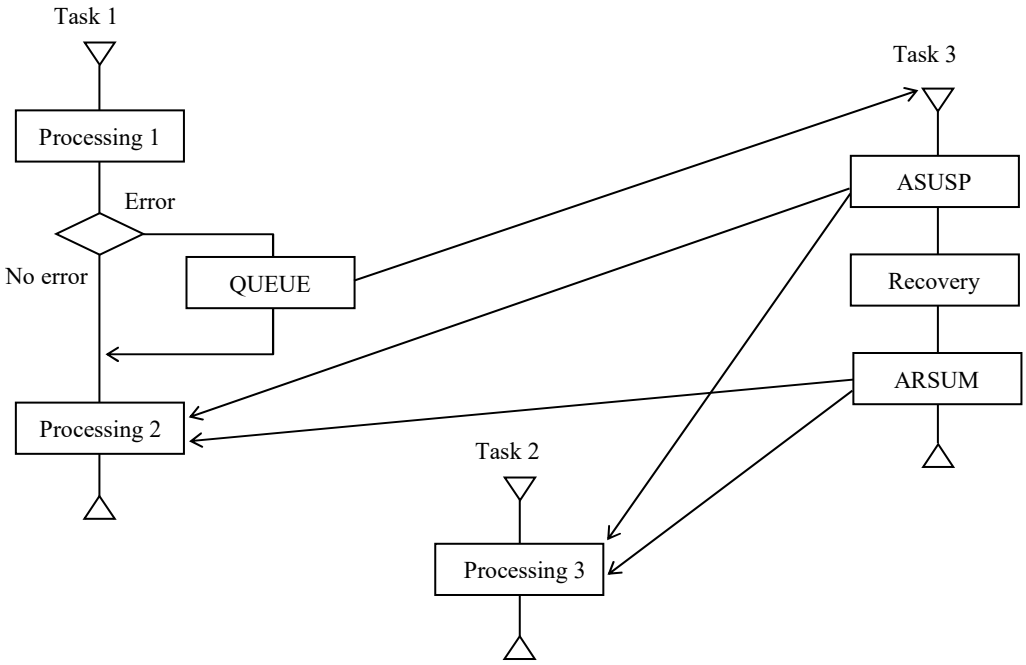
Figure 1-16 Application of the DELAY Macro Instruction

- ASUSP macro instruction

  The ASUSP macro instruction is used to suppress the execution of all other tasks, including higher-priority tasks.

  Tasks whose execution is suppressed by using the ASUSP macro instruction can have their suppressed state canceled by using the ARSUM macro instruction. However, because these instructions are used to suppress the execution of other tasks, a deadlock might occur if their use is not limited.

  To avoid such deadlocks, any processing that requires system resources must not be performed between the time the ASUSP macro instruction is issued and the time the ARSUM macro instruction is issued.



Task 3, which manages processing 1, 2, and 3, suppresses the execution of tasks 1 and 2 by using the `ASUSP` macro instruction when an error occurs in task 1. After suppression, task 3 performs recovery processing so that processing 2 and 3 can run normally, and then uses the `ARSUM` macro instruction to cancel the suppression of execution of tasks 1 and 2. Task 3 uses the `ASUSP` macro instruction to make processing 2 and 3 wait until recovery processing ends.

Figure 1-17 Suppressing Execution by Using the ASUSP Macro Instruction

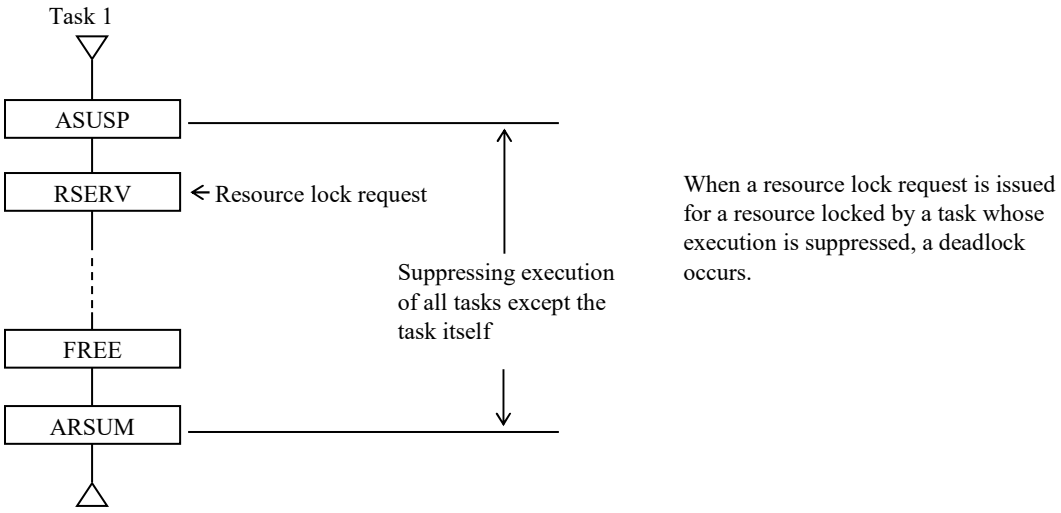


Figure 1-18 Example of a Deadlock Caused by the ASUSP Macro Instruction

2.5.5 Aborting tasks

- ABORT macro instruction

  To abort execution of a task and prohibit its execution, use the ABORT macro instruction. The ABORT macro instruction aborts a task in execution (or in the WAIT state), forcibly frees resources locked by the task, and puts the task into the DORMANT state.

2.5.6 Synchronization between tasks

The WAIT and POST macro instructions are available for synchronizing multiple tasks (to perform the processing of another task after that of another task ends). This synchronization is controlled by a concept called the *event*. To synchronize with another task, a task notifies an area called the ECB (event control block) that it is waiting for an event to occur, and enters the WAIT state. This ECB is defined for each event. A task sending notification of an event occurrence references the ECB, checks which task is waiting for an event to occur, informs the waiting task of an event occurrence, and releases the task from the WAIT state. This processing is performed by the WAIT and POST macro instructions respectively. Figure 1-19 illustrates this scenario.

One ECB is assigned per event. The same ECB must not be shared among multiple events, and multiple tasks must not share the same ECB. Through the ECB, detailed event information can be exchanged between tasks. This information is called the POST code.

In the WAIT and POST macro instructions, there are no limitations relating to issuing order. This is shown in Figure 1-20.

To prevent deadlocks, the ASUSP macro instruction loses its effect if the WAIT macro instruction is issued after the ASUSP macro instruction is issued.

Figure 1-22 shows an ECB state transition.

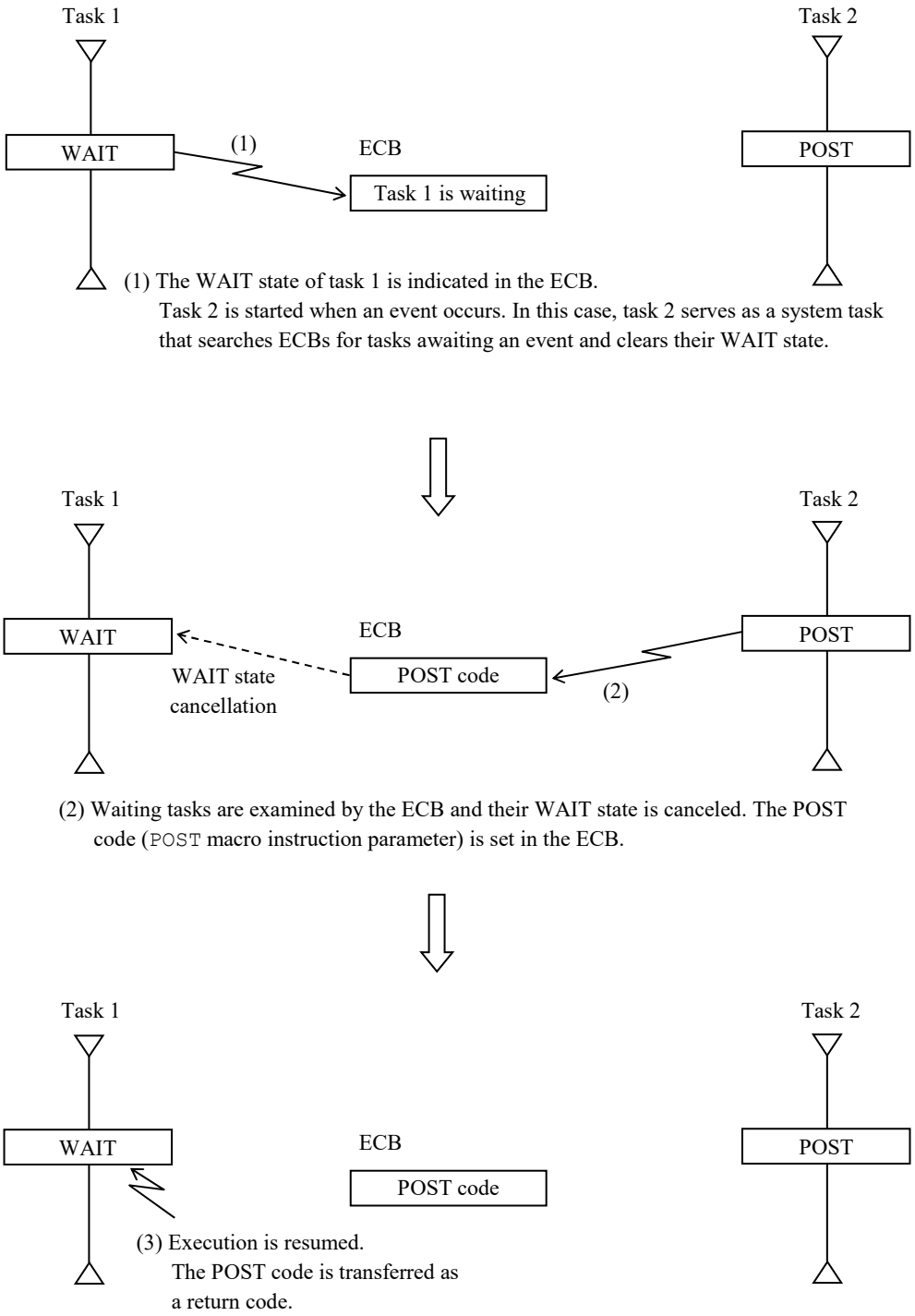
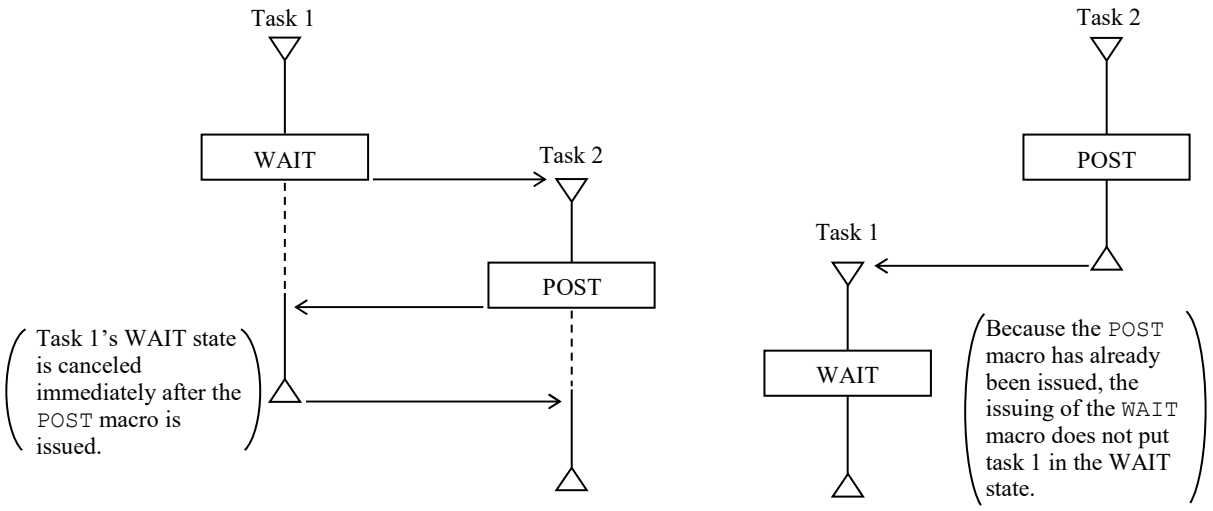
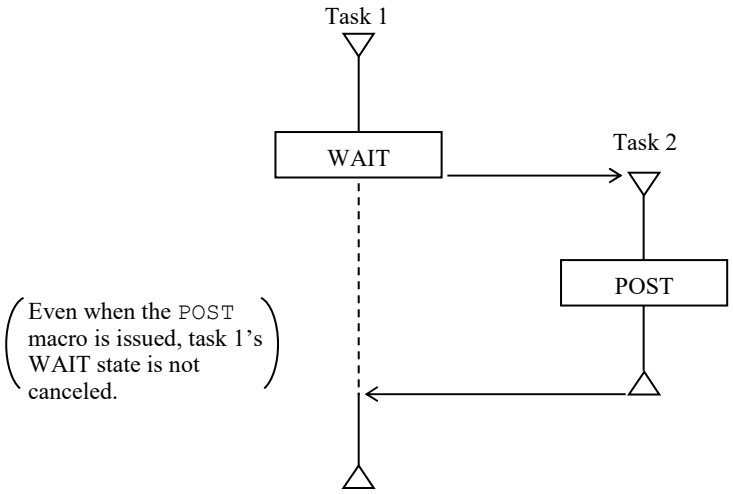


Figure 1-19 Synchronizing Tasks by Using the WAIT and POST Macro Instructions



When the WAIT macro is issued first, and task 1 is of higher level.

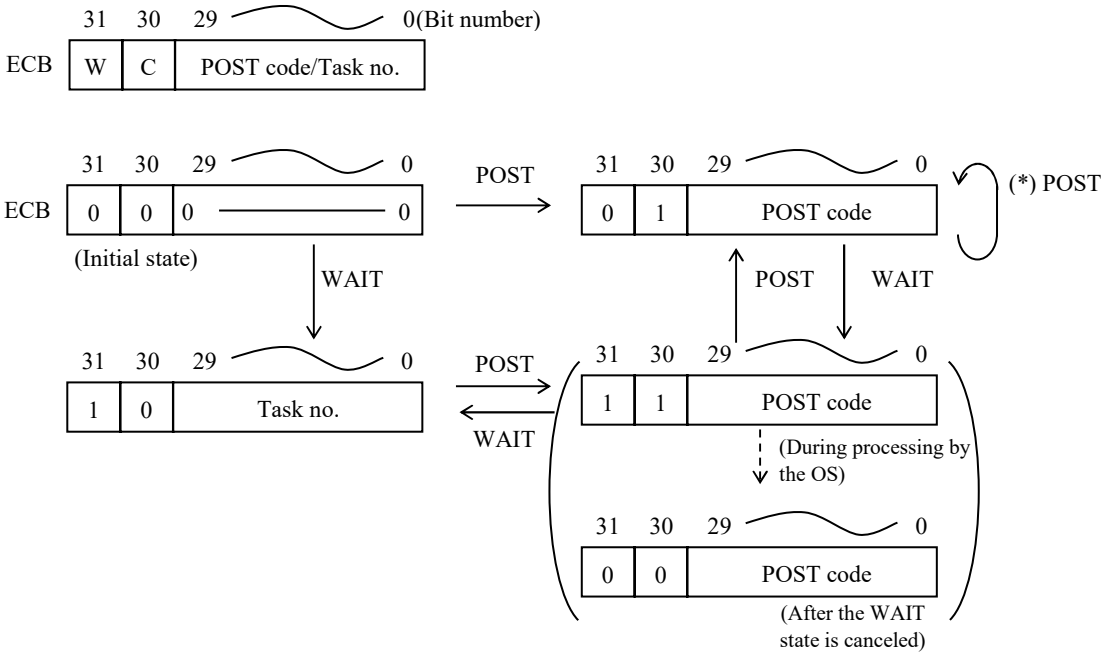
When the POST macro is issued first, and task 2 is of higher level.



The WAIT macro is issued first, and task 2 is of higher level.

The control flow between tasks varies depending on difference in level between tasks, and the order in which the WAIT and POST macros were issued.
The dotted line indicates that task execution during that time is suppressed (in the WAIT state).

Figure 1-20 Control Flow Using the WAIT and POST Macro Instructions



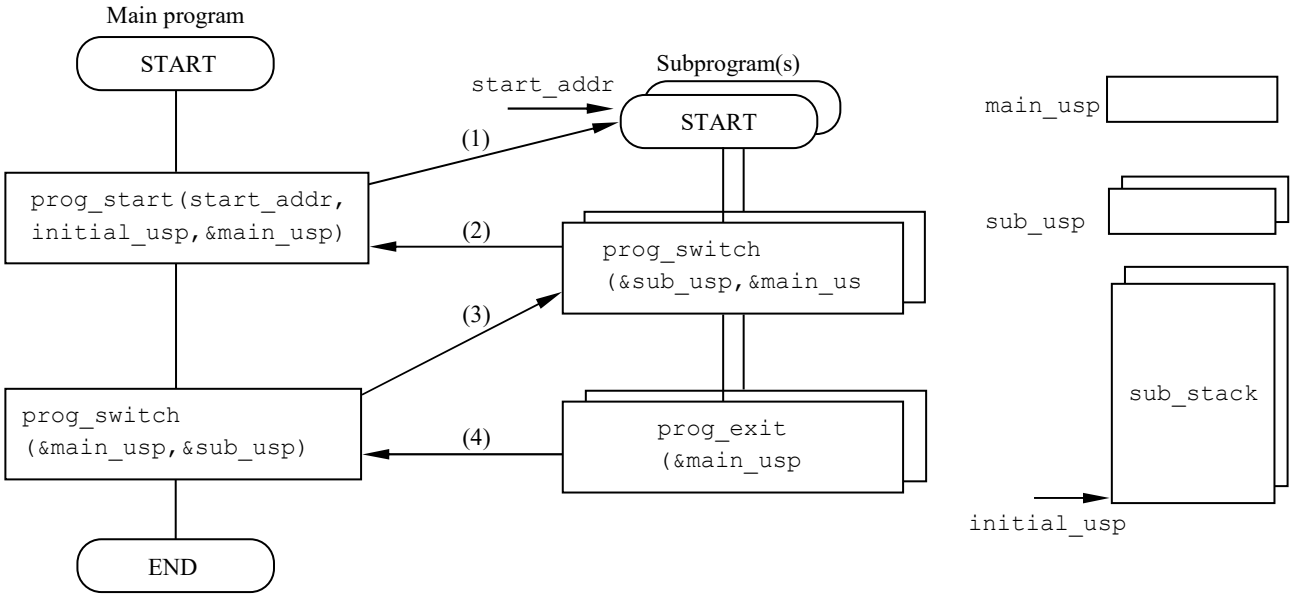
ECB bit numbers 31 and 30 are called the W (wait) bit and the C (complete) bit.
POST codes marked with an asterisk (*) are overwritten.

Figure 1-21 ECB State Transitions

2.6 Subtask Execution Control

- Subtask execution control

  Support is provided for executing programs in user tasks as shown in Figure 1-22.



(1) The main program performs an initial start on the subprogram.
(2) The subprogram is put in the WAIT state, and control is transferred to the main program.
(3) Control is transferred from the main program to the waiting subprogram.
(4) The subprogram ends, and control is transferred to the main program.

If `sub_stack` and `sub_usp` are prepared for each subprogram, the main program can run multiple subprograms. The CPMS treats the main program and subprograms as one task. As a result, scheduling is performed so that no pre-emptions occur between the main program and subprograms.

Figure 1-22 Subtask Execution Control

- Sharing a stack among subtasks
  Each subtask has its own stack. If there are 128 subtasks that each have a 10 KB stack, 1280 KB memory is required in total. However, each individual subtask only needs to save its stack contents for `prog_switch()`. If we assume 2 KB of the aforementioned 10 KB is required for this, the rest of the stack (8 KB) can be shared among all subtasks. This means the required memory is just 128 x 2 + 8 = 264 KB.
  To share a stack among subtasks, have each subtask use the shared stack when it calls a subprogram.
  Do not call `prog_switch()` in a subprogram that uses a shared stack.
  Do not share a stack among subprograms that are called from different main tasks.

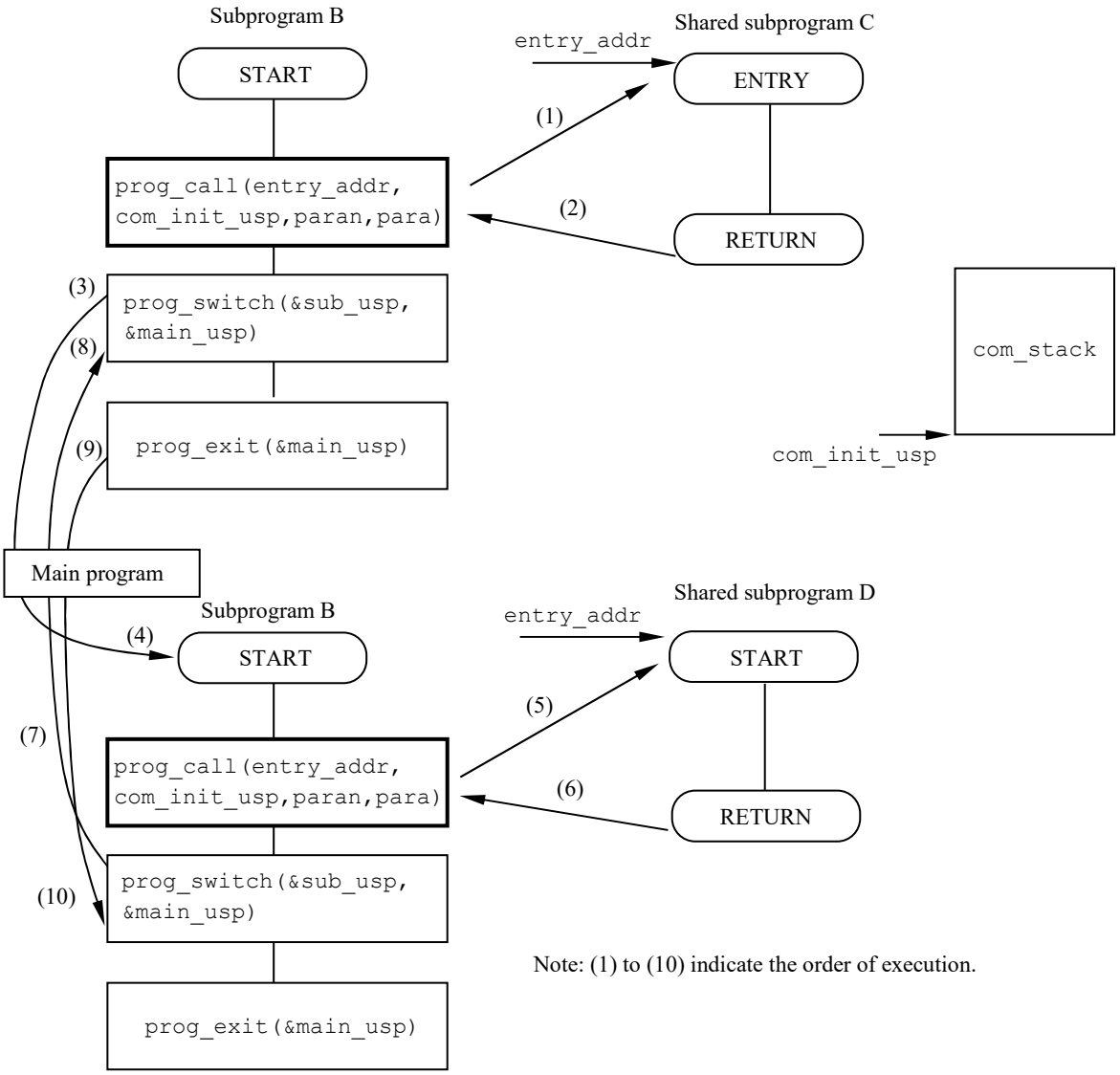
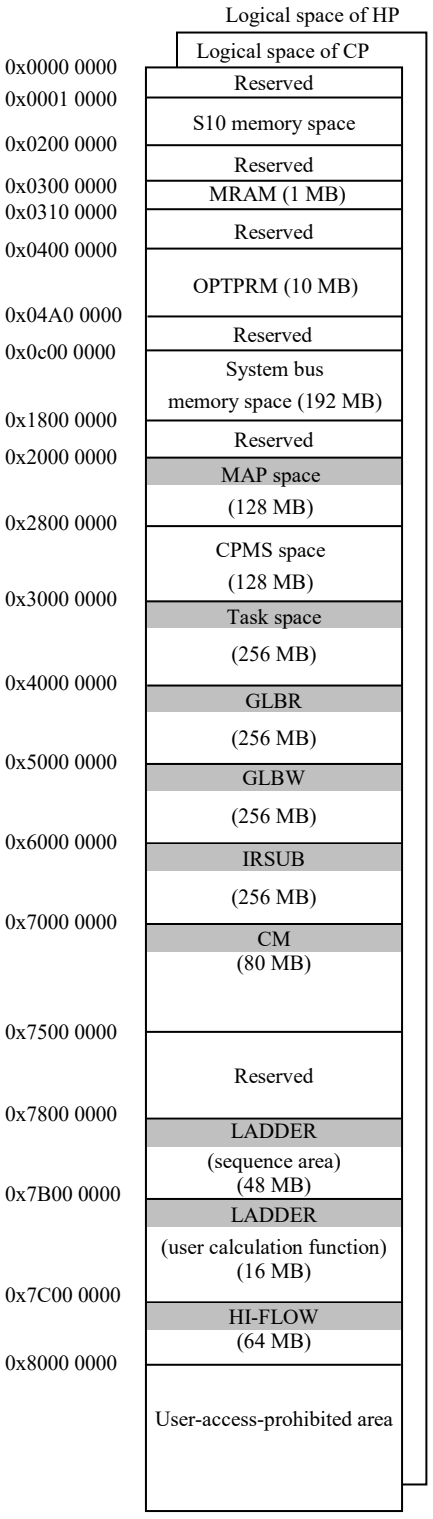


Figure 1-23 Sharing a Stack Among Subtasks

CHAPTER 3 MEMORY MANAGEMENT

3.1 Logical Space

The CPMS operates all tasks in a single logical space. The CPMS also manages conversion between logical addresses and physical addresses.



Reserved: This area is currently unused. It is reserved for future use.

S10 memory space:
The I/O and memory for S10 are assigned to this area.

MRAM: This is the MRAM for users.

OPTPRM: This area stores the setting parameters of optional modules. This area cannot be accessed by user tasks, and is mapped only to the CP side.

System bus memory space:
The I/O and memory for the system bus are assigned to this area.

MAP space: The tasks and IRSUB used by the CPMS, and the built-in subroutine management table are placed in this area.

CPMS space: This area is dedicated to the CPMS.

Task space: The task areas TEXT, DATA, BSS, STACK, and OS work are assigned to this area.

GLBR: The (read-only) memory shared among tasks in the PU is assigned to this area.

GLBW: The (read/write) memory shared among tasks in the PU is assigned to this area.

IRSUB: The indirect link subprogram shared among tasks is assigned to this area.

CM: The memory shared among PUs is assigned to this area.

LADDER: This is the area for storing ladder programs. This area is mapped only to the HP side.

HI-FLOW: This is the area for storing HI-FLOW programs. This area is mapped only to the HP side.

User-access-prohibited area:
Tasks cannot access addresses from 0x80000000 onwards. Attempts to access this area result in a program error.

indicates sub-areas that have a one-to-one correspondence to physical memory. Each sub-area corresponds to a pre-assigned amount of physical memory starting from the starting address of the parent area. Tasks cannot access any location in this parent area other than this sub-area. Any attempts to do so result in a program error. The assigned size is fixed, and cannot be changed by users.

The following table shows the assigned size per space.

Name of logical space	Assigned size (MB)	
	CP	HP
Task space	12	4
GLBR	4	2
GLBW	17	4
IRSUB	4	2

Figure 1-24 Logical Address Map

3.2 CM

CM (common memory) is memory that is shared among CPUs (CP and HP) connected by the system bus.

A logical address is mapped to the same physical address in the main memory.

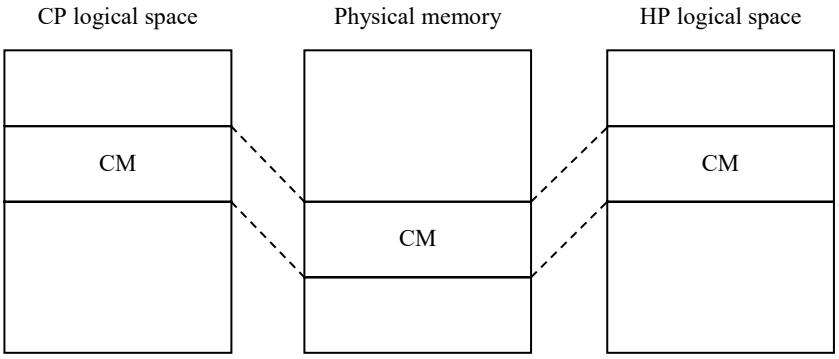


Figure 1-25 Mapping of CM

3.3 Memory Protection

The CPMS manages memory write protection in units of 4-KB pages.
Table 1-8 lists the memory access rights.
Tasks can write to the following memory spaces. The other spaces are write-protected.

- BSS and STACK for the local task (Note that when a multitask is used, BSS is shared.)
- Areas in the S10 memory space, GLBW, CM, and MRAM where the logical space and physical memory are mapped
- PI/O and cyclic transfer memory in the system bus memory space

The CPMS provides the `wrtmem` macro, which allows user programming tasks to rewrite programs and protected data. This macro can be used to write to write-protected main memory.

Table 1-8 Memory Access Rights

Space type	Model	S10VE		Remarks
	Accessed by	CPMS	Task	
	Access mode	System	User	
Task space (in the user space)				
	Text for the local task	R-X	R-X	
	Data for the local task	R-X	R-X	
	Stack for the local task	RWX	RWX	
	BSS for the local task	RWX	RWX	
	Text for other tasks	R-X	R-X	
	Data for other tasks	R-X	R-X	
	Stack for other tasks	R-X	R-X	
	BSS for other tasks	R-X	R-X	RWX when a multitask is used
User space (except the task space)				
	CM	RWX	RWX	
	LADDER (ladder sequence area)	R-X	R-X	Can only be accessed by the HP
	LADDER (user calculation function)	R-X	R-X	Can only be accessed by the HP
	HI-FLOW	RWX	RWX	Can only be accessed by the HP
	GLBW	RWX	RWX	
	GLBR	R-X	R-X	
	IRSUB	R-X	R-X	
	MAP	R-X	R-X	
	System bus memory space (for users)	RWX	RWX	For memory interface I/O (e.g., PI/O, transfer memory)
	System bus memory space (for the system)	R-X	R-X	For the OS subsystem (driver)
	CPMS space (read-only)	R-X	R-X	
	CPMS space (read/write)	RWX	RWX	
	S10 memory space (for users)	RWX	RWX	
	S10 memory space (for the system)	R-X	R-X	
	MRAM	RWX	RWX	Can only be accessed by the CP
	OPTRAM	RWX	RWX	
Kernel space				
	Space with V = R specified in the main memory	RWX	–	Includes the text and data of the CPMS
	I/O register space	RWX	–	Can only be accessed by the kernel and I/O driver
	KROM space	R-X	–	

R: Readable
W: Writable
X: Executable
–: Inaccessible (Tasks that attempt to access such memory are aborted.)

3.4 Error Handling During Memory Access

- Memory error
  If a multi-bit error occurs in memory that has the ECC feature, the system stops. (If a multi-bit error occurs, both the CP and HP stop.)
- Single-bit memory error
  Single-bit errors in memory that has the ECC feature are corrected, and the data is read correctly. Therefore, single-bit errors are not handled as errors. If a single-bit error is encountered during patrol scrubbing, the data is rewritten to correct the error. If the single-bit error persists, it is handled as a solid failure and an alarm report is recorded in the error log.
- System bus access error
  An attempt to access a non-mapped address results in a program error. Also, even if an address has been mapped, an attempt to access it in the event of a hardware failure might result in a system bus error. Such errors are handled as target abort errors, not program errors. The following describes system behavior in the event of a target abort error:
  - If the target abort error is detected during a read access, data where all the bits are `1` is read.
  - If the target abort error is detected during a write access, the program continues operating as if the data was written.
  - In response to the target abort error, an interrupt to the PU is generated, resulting in a module error.
- Write-protect error
  A write might occur at a write-protected address due to a software failure. This results in a program error, and the task is aborted.

3.5 Procedure for Accessing the System Bus

The system bus memory space is accessed directly from user programs as the bus memory. To detect failures during access to bus memory, you need to perform the following procedure: Have the user program accessing the bus memory issue a `CHKBMEM` macro to check whether the bus memory in the specified slot is accessible. The `CHKBMEM` macro returns information that indicates whether: (1) the bus memory is mounted in the specified slot, (2) the specified slot is in the CARD OFF state, and (3) a target abort error has occurred due to a system bus error. If the `CHKBMEM` macro detects an error, do not access the bus memory in the slot where the error was detected.

After accessing the bus memory, use the `CHKTAER` macro to check whether a target abort occurred. This step is necessary because tasks continue processing as usual even if a target abort occurs. Note, however, that PI/O readout processing reads out all bits of data as `1` if a hardware error has occurred. This means that if not all the bits of data you want to read out are `1`, you can tell if a hardware error has occurred without using the `CHKTAER` macro.

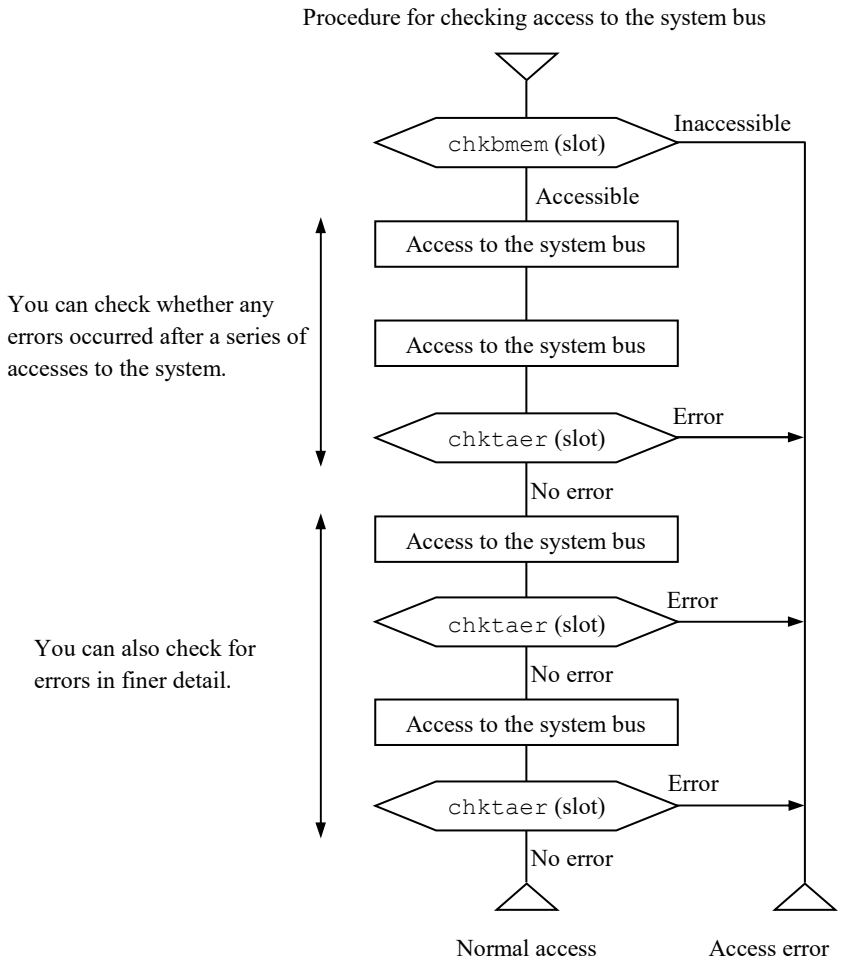


Figure 1-26 Procedure for Checking Access to the System Bus

CHAPTER 4 TIMER MANAGEMENT

4.1 Duration and Time

The CPMS manages durations and times in the PU. Times are expressed in years, months, days, and milliseconds since 12 a.m. Valid years are from 1970 to 2069. Durations are expressed in milliseconds.

Tasks can use the GTIME macro to get times managed by the CPMS. The STIME macro can also be used to set times managed by the CPMS.

The CPU is equipped with a battery-powered clock (RTC: real-time clock) that can operate even during power failures. When the CPMS is started, it reads the year, month, day, hours, minutes, and seconds from the RTC and sets the result as the starting point from which the time is measured. During operation, the CPMS manages durations and times by using an internal timer based on the clock supplied with the processor. Because the RTC and internal timer operate based on different clocks, disparities might occur after long periods of time. Once a day, the CPMS sets the time read from the internal timer to the RTC to correct any disparities.

4.2 Duration- and Time-Based Task Control

Tasks can use the DELAY macro to suppress their own execution for a specified duration. The TIMER macro can also be used to create a timer that starts a task at a specified time or after a specified duration, and then restarts the task in cycles (at fixed intervals). This timer can be deleted by using the CTIME macro. Timers created by the TIMER macro only accept times within 24 hours of the time the TIMER macro was issued.

4.3 Effect of Changing the Time on Timer Operation

Changing the time by using the STIME macro affects the operation of timers set to start tasks at a certain time by using the TIMER macro. If the scheduled time is skipped due to the time being advanced, the task might lose its chance to start because its first scheduled start time has already passed. In this case, the task is started when the time is changed. In the case of a time-cycle-based timer, the scheduled start time is changed to the time the change was made or later, by adding the cycle time to the first scheduled start time.

Timers that have already started a task at the scheduled time do not re-register to start the task at the scheduled time even if the time is delayed (moved backward).

For duration-based timers, the start time is not changed even if the time is changed.

4.4 Clock Synchronization in the CPU (Between CP and HP)

In the CPU, the CP is equipped with an RTC, but the HP is not. Therefore, the time in the CP is set as the current time in the HP on the following occasions (1) to (5):

(1) When CPU (CP or HP) starts
(2) When an `STIME` macro is issued by the CPU (CP)
(3) When the CPU (CP) time is synchronized (00:00:30)
(4) When the system register (SW200 to SW2F0) is used to set the CPU (CP) time
(5) When the CPU (CP) time is set from BASE SYSTEM/S10VE

Note that the time in the CPU (HP) is disrupted when the time in the CPU (CP) is copied to the CPU (HP) as the current time on the preceding occasions (1) to (5).

CHAPTER 5 SHARED RESOURCE MANAGEMENT

5.1 Shared Resources

Resources that are shared among tasks include the main memory unit, CPU, I/O, and data area (GLB). Among these, the main memory unit, CPU, and I/O are exclusively controlled on the system side. However, GLB and the like must be exclusively controlled on the user side. Figure 1-27 shows the necessity of this exclusive control. Figure 1-28 shows how problems caused by resource contention are prevented through exclusive control.

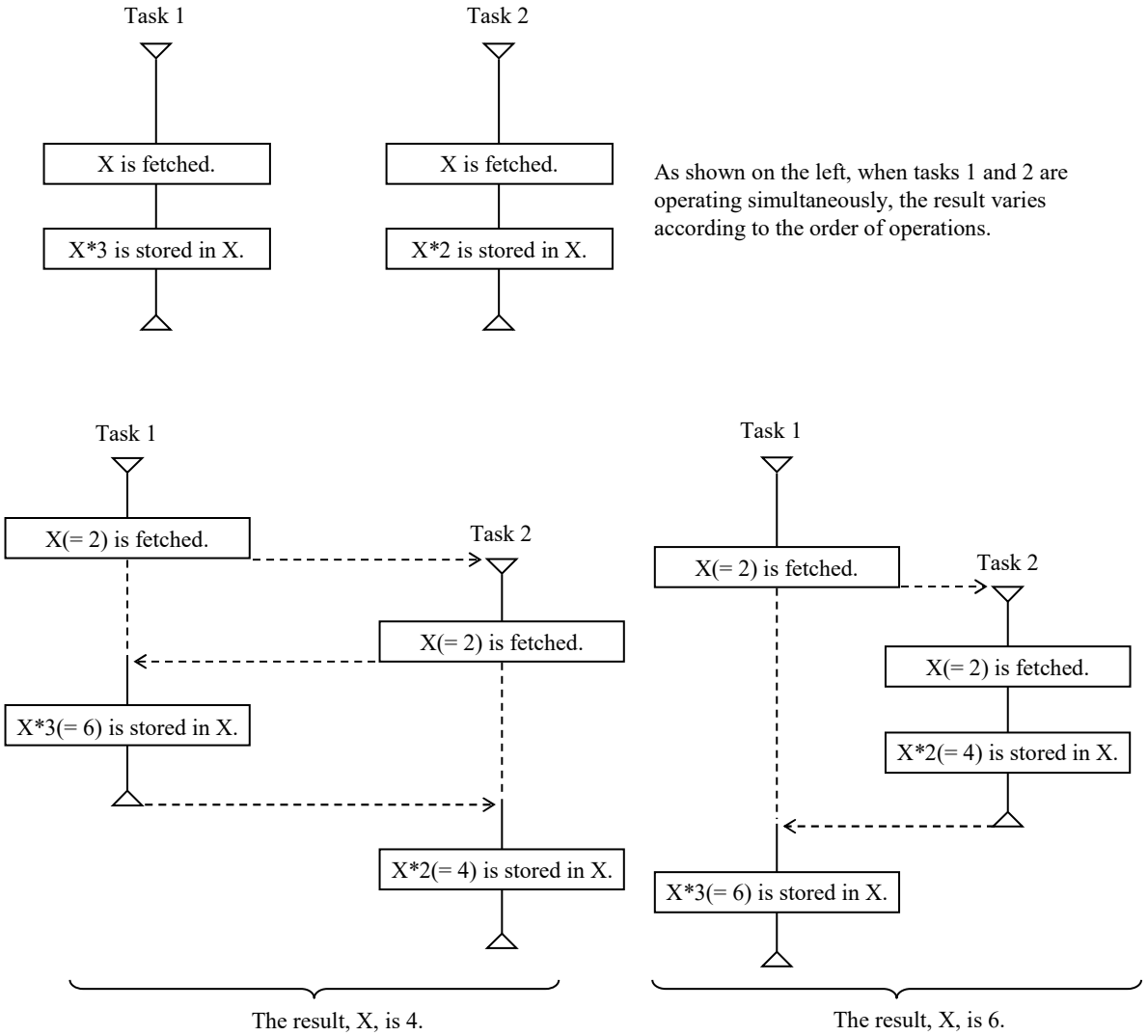
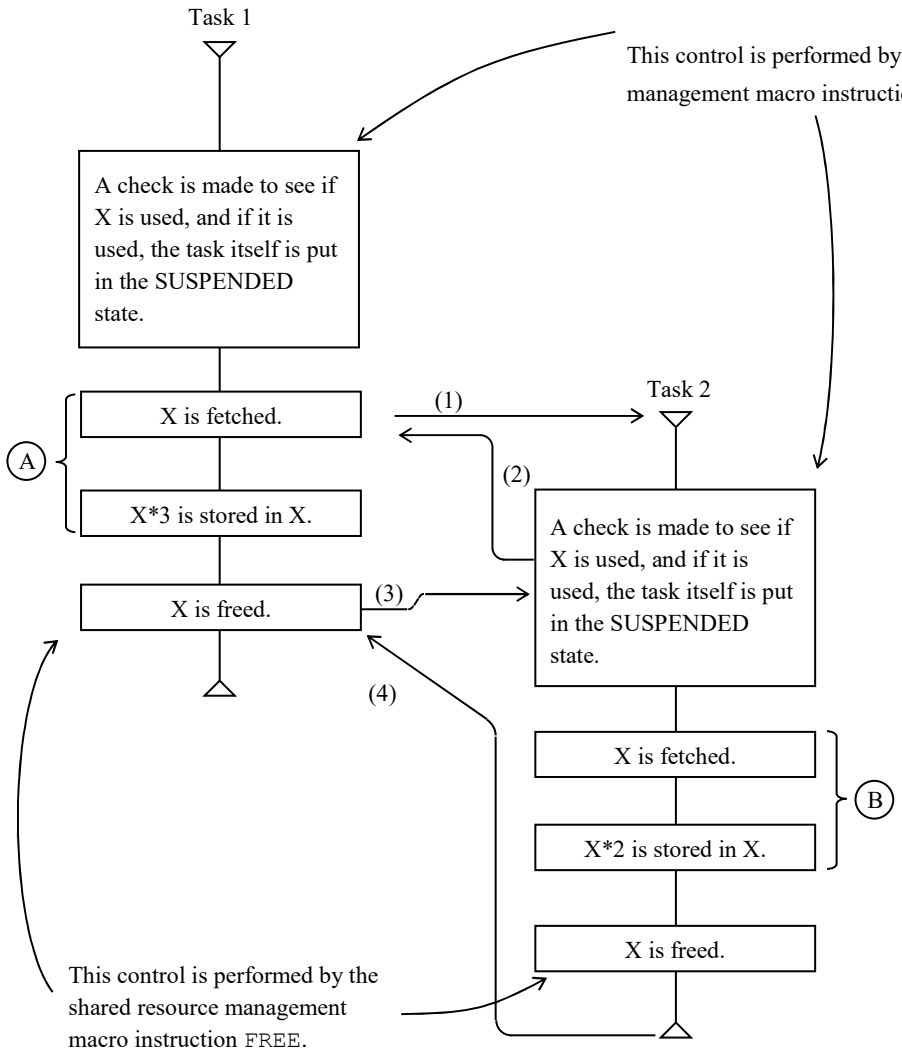


Figure 1-27 Problem that Occurs when Exclusive Control is Not Performed

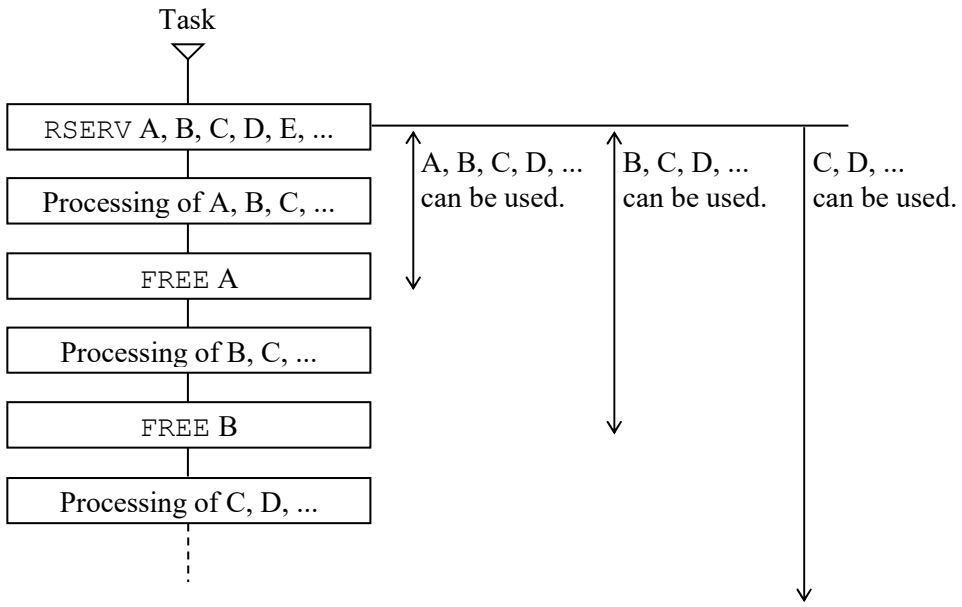


Control flows in the order (1), (2), (3), and (4) to prevent both A and B from being operated at the same time.

Figure 1-28 Exclusive Control by Shared Resource Management Macro Instructions

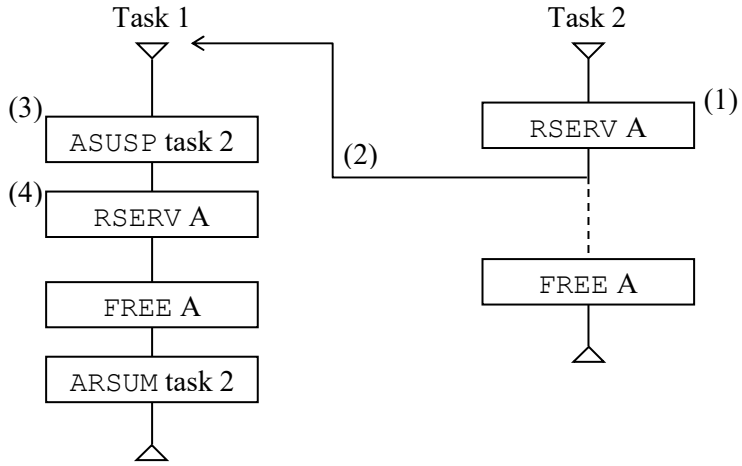
5.2 Method of Managing Shared Resources

For GLB, a resource that is shared among tasks, the physical resource itself can be locked. That is, each time the GLB address and size are registered in the system table that manages shared resources and a lock request is sent by using the RSERV macro instruction, this system table is referenced and a check is made to see whether the target GLB is already locked. If the target GLB is locked, the requesting task is put in the SUSPENDED state by using the RSERV macro instruction until the resource is freed. The SUSPENDED state of this task is canceled when the requesting resource is freed and becomes usable.
When multiple tasks are waiting for a resource to be freed, the resource is allocated to the highest-level task among them. However, this rule does not apply when the highest-level task cannot be operated for some other reason.
As a rule, all resources required by a task need to be locked at one time (in one batch) to avoid deadlocks. Therefore, multiple issuance of the RSERV macro instruction is not allowed, meaning that a task that has already locked a requested resource cannot issue this RSERV macro instruction again. As shown in Figure 1-29, a task locks all necessary resources from the beginning of processing, and resources it has finished using are immediately freed by using the FREE macro instruction.
Figure 1-30 shows an example of a deadlock. As shown in this example, the RSERV macro instruction must not be issued after a macro instruction that suppresses execution of another task, for example the SUSP macro instruction.



- All the resources used by a task are locked at the same time, and each resource is freed by using the FREE macro instruction as soon as the task has finished using it.
- A single FREE macro instruction can free multiple resources at a time.

Figure 1-29 Using RSERV and FREE



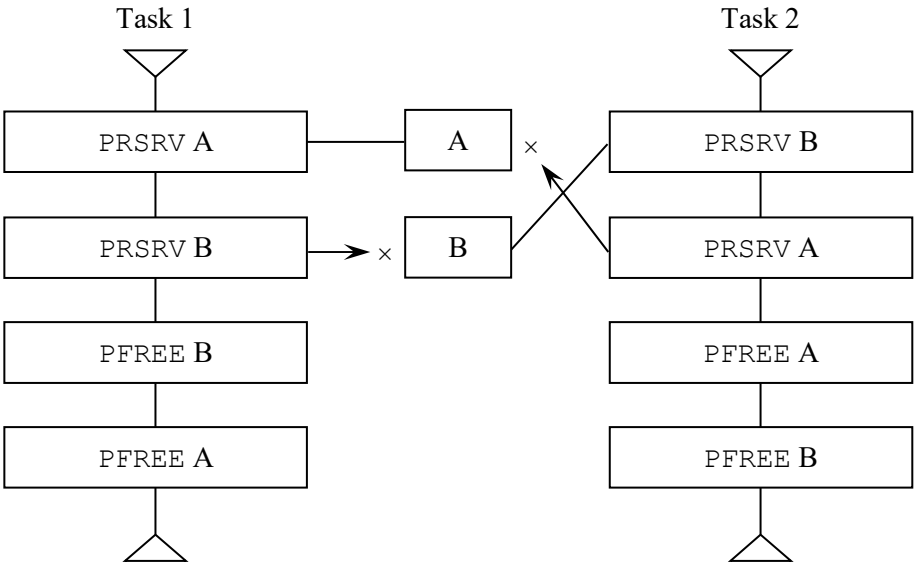
(1) Task 2 locks resource A.
(2) Control is transferred from task 2 to task 1 before task 2 frees resource A.
(3) Task 1 performs SUSP processing for task 2 (making task 2 inoperable and unable to free resource A).
(4) Task 1 attempts to lock resource A, but this resource has already been locked by task 2. Task 1 is put in the SUSPENDED state, and cannot perform RSUM processing for task 2.

This results in both tasks 1 and 2 becoming unexecutable.

Figure 1-30 Example of a Deadlock

5.3 Exclusive Control of Shared Resources by Using the PRSRV and PFREE Macros

The PRSRV and PFREE macros can be used for exclusive control of shared resources among tasks, with more precision than by using the RSERV and FREE macros.
To start locking a resource: Issue a PRSRV macro with SAREA in GLB and the locked range specified.
To stop locking a resource: Issue a PFREE macro with SAREA in GLB and the locked range specified.
When a specified GLB area cannot be locked, control is not returned to the task that issued a PRSRV macro until the shared resource is freed.
Tasks can issue the PRSRV macro as many times as necessary. Because a task can use more than one PRSRV macro to gradually lock more and more of the multiple shared resources rather than locking all the resources at one time, this reduces the number of times tasks must wait to lock resources. However, the order in which the shared resources are locked must be clarified to prevent deadlocks.



In the preceding example, task 1 is locking resource A, and task 2 is locking resource B. Task 1 waits for resource B to be freed, while task 2 waits for resource A to be freed. Each task is waiting for the other to free the locked resource, making neither task executable.
A solution to this problem is to have the same resources locked in the same order.

Figure 1-31 Example of a Deadlock Caused by the PRSRV Macro

CHAPTER 6 I/O DEVICE MANAGEMENT

6.1 Structure of the I/O Device Management Function

The CPMS provides subsystems (I/O drivers) equipped with the basic functions of I/O device management, which are responsible for controlling I/O devices. Users must perform I/O operations by using the interfaces provided by individual subsystems.

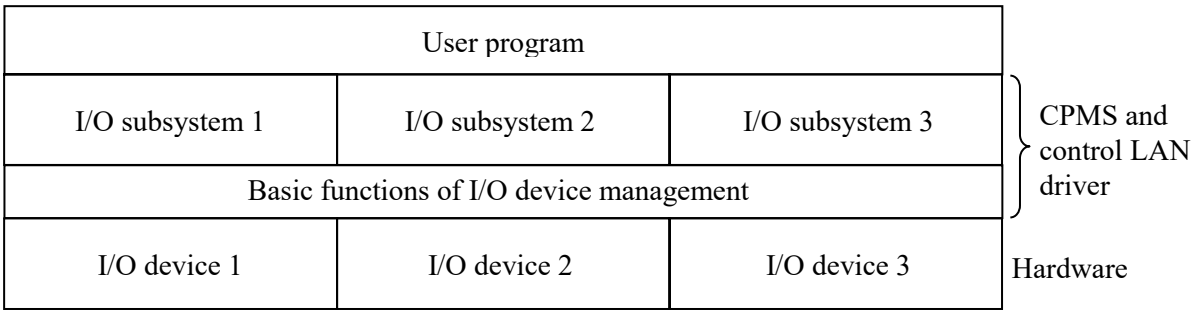


Figure 1-32 Structure of the I/O Device Management Function

6.2 I/O Unit Number

The CPMS identifies the system bus connection I/O as the I/O target (device) by unit number (abbreviated to *UNO*). The assigned unit number is the connected slot number plus 4.

6.3 Device Number

The device number identifies a logical device and the driver that controls the logical device. The logical device is used to define the purpose of a device. More than one logical device can be defined for a single device.

The device number consists of a major number and minor number. The major number identifies the subsystem that controls the device. The minor number specifies the location where the device is connected and its purpose. The device-dependent field is defined individually for each subsystem.

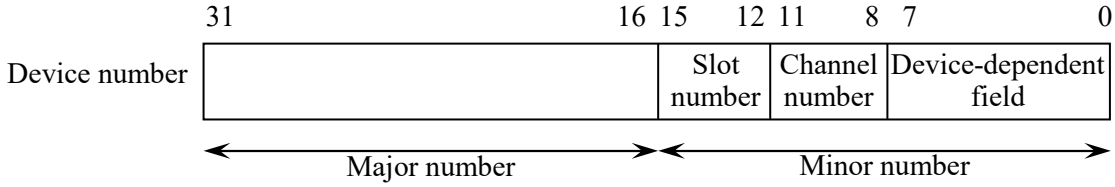


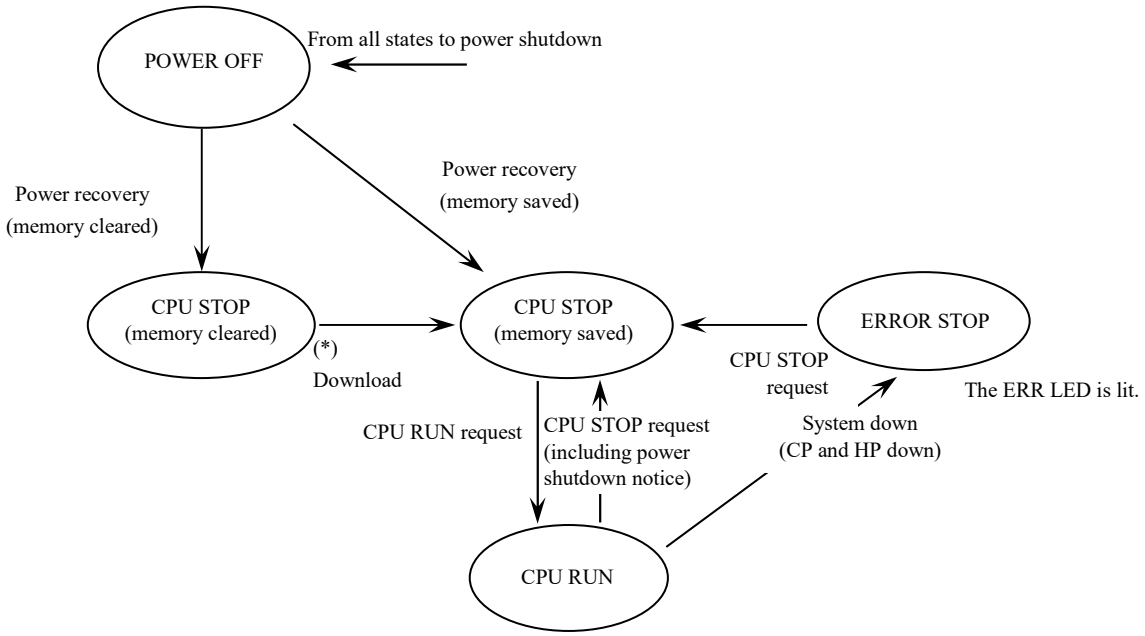
Figure 1-33 Device Number

CHAPTER 7 SYSTEM MANAGEMENT

7.1 Starting and Stopping the CPMS

7.1.1 State changes when starting and stopping the CPMS

Figure 1-34 shows how the state changes when the CPMS is started and stopped. Table 1-9 describes each state, while Table 1-10 describes the events.
The CPMS states in Figure 1-34 correspond to the states of both processors (CP and HP) in the CPU. The CP and HP are paired processors in the CPU, and therefore the states of the CPMS on both processors (CP and HP) are the same and change in sync.



(*) If FROM contains memory backup data, the data is downloaded to the memory from FROM. If FROM does not contain memory backup data, the development machine uses BASE SYSTEM/S10VE to perform *CPMS download* and download the data to the memory.

Figure 1-34 State Changes when the CPMS is Started and Stopped

Table 1-9 Startup and Stop States

State	Description
POWER OFF	The power is off.
CPU STOP (memory cleared)	The downloaded memory contents are not saved, so the CPU is stopped.
CPU STOP (memory saved)	Any valid contents of the downloaded memory are saved, and the HP and CP system programs are stopped.
ERROR STOP	The system programs of both the HP and CP are stopped due to an error.
CPU RUN	The system programs of the HP and CP are in execution. The system programs of the HP have stopped due to an error, and some system programs of the CP might be in execution.

Table 1-10 Startup and Stop Events

Event	Description
Power recovery	The power is turned on.
Power shutdown	The power is turned off.
Download	System programs and user programs are stored in the memory.
CPU RUN request	One of the following events occurs: • The power is recovered when the CPU RUN/STOP switch is in the RUN position. • The CPU RUN/STOP switch is switched from the STOP to the RUN position. • When the CPU RUN/STOP switch is in the RUN position and the CPU is stopped due to a remote STOP request from BASE SYSTEM/S10VE, BASE SYSTEM/S10VE sends a remote RUN request.
CPU STOP request	One of the following events occurs: • During the CPU RUN state, the power supply sends a power shutdown notice. • Power is recovered when the CPU RUN/STOP switch is in the STOP position. • The CPU RUN/STOP switch is switched from RUN to STOP. • The BASE SYSTEM/S10VE sends a remote STOP request when the CPU RUN/STOP switch is in the RUN position.
System down	The system program stops due to an error.

7.1.2 Startup operation

The first hardware state is the POWER OFF (memory erased) state. When the power is recovered from this state, the state transitions to CPU STOP (memory cleared). After that, when CPMS download is performed by BASE SYSTEM/S10VE, the state transitions to CPU STOP (memory saved). When this download is performed, the data is saved in FROM of the CPU module. From then onward, if a reset or power cycle occurs, the downloaded data is copied from FROM to memory, and the state transitions to CPU STOP (memory saved). Then, the CPMS is started by using a CPU RUN request as described in Table 1-10, and the state transitions to CPU RUN.

7.1.3 Stop operation

The CPMS is stopped by using a CPU STOP request, and the state transitions to CPU STOP (memory saved). Even if the power shuts down during CPU RUN, a CPU STOP request is received due to a power shutdown notice (POP signal) from the power supply.
In addition, if a major system failure is detected, the system goes down and the state transitions to ERROR STOP (memory saved).

7.2 INS Built-in Subroutine and the Initial Start Task

At the end of processing during OS startup, the CPMS carries out the following procedure. After that, the CPMS transitions to the application STOP state of the task RUN state.

(1) Link to the INS built-in subroutine.
(2) Start the system initial start task (SIST: task number 255).
(3) Start the user initial start task (UIST: task number 1).
(4) Initialize the settings in the CP-side UIST. From the CP-side UIST, you can initialize the settings for ladder programs and the settings for before remote I/O is started.

The CPMS passes numbers corresponding to the initiation factors listed in Table 1-11, as parameters of the INS built-in subroutine and initiation factors of the initial start task.

Table 1-11 Initiation Factors

No.	Initiation factor	Description
1	IPL start	The initial start task is started after the OS, TASK, IRSUB, GLB, and CM have been downloaded. This factor is passed after BASE SYSTEM/S10VE performs the CPMS download, and after RPDP is used to execute the `svrpl` command.
5	Reset fast restart	When the state has transitioned from CPU RUN to CPU STOP due to a CPU STOP request, the initial start task is started by using a switch or remote RUN request without performing any download. Part of the OS initialization processing is skipped to make the restart process faster. The initial start task is started from the saved state of the GLB and CM.
6	Reset start	When the CPU is down due to a serious error during the CPU RUN state, the initial start task is started by using a switch or remote RUN request without any download. Data containing the initial values of the OS is restored to its post-IPL state, and the controller starts from the post-IPL processing. The initial start task is started from the saved state of the GLB and CM.
7	FROM start	The initial start task is started after OS, TASK, IRSUB, GLB, and CM are copied from FROM. This factor is passed when power is recovered after the power shuts down.

7.3 State Transition During RUN, and LED Output

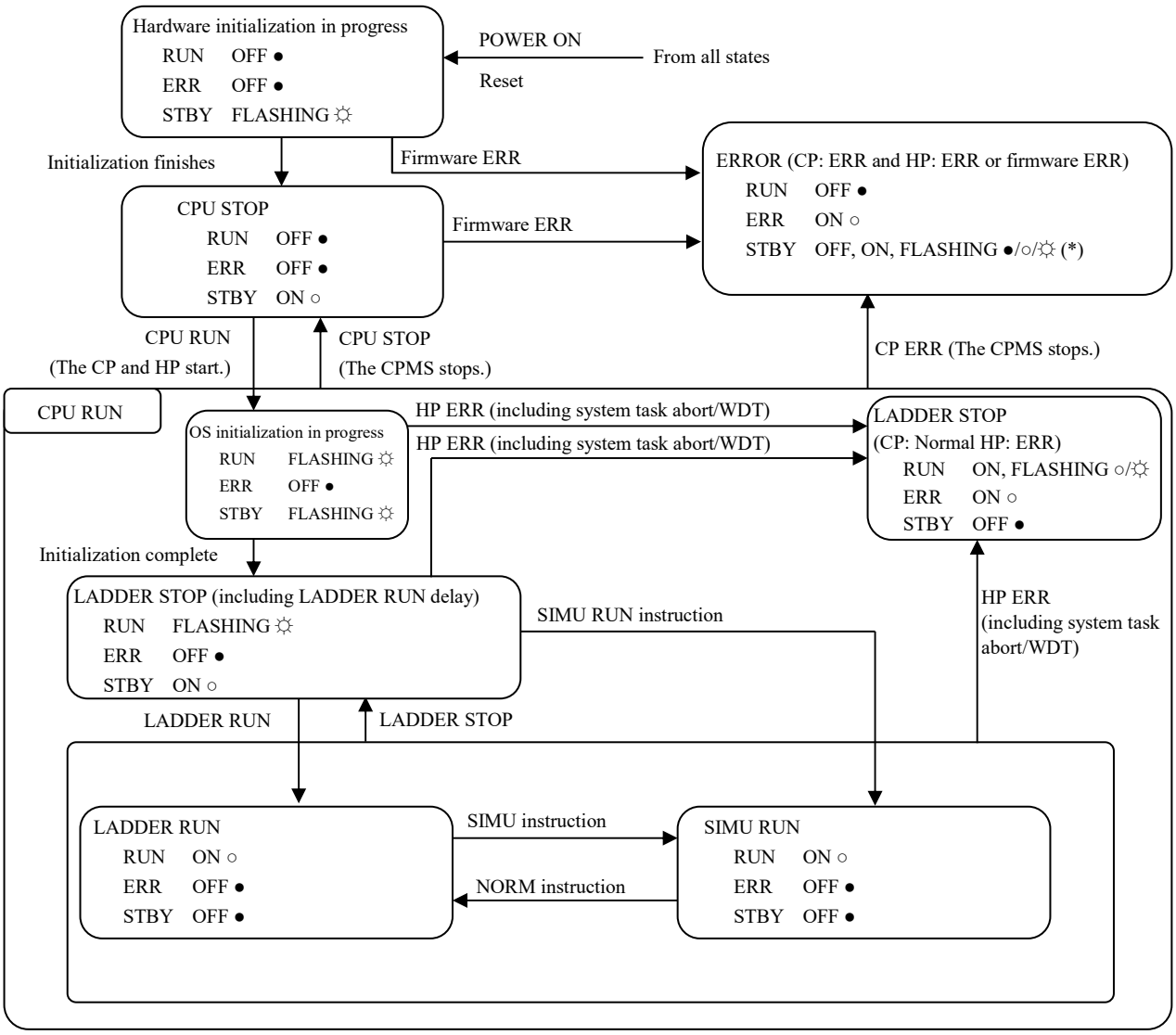
(1) States of the CPMS during RUN, and LADDER RUN state

During RUN, the CPMS has the following three states: LADDER STOP, LADDER RUN, and SIMU RUN.

Each of these states is indicated by a combination of ON, FLASHING, and OFF states of the CPU state indicator LEDs RUN and STBY.

Figure 1-35 shows the LED state transitions and LED output. Table 1-12 shows the details of each state.

The ERR LED indicates an error state. The ERR LED is on when a hardware or system error has occurred. A system watchdog timer timeout also causes the ERR LED to turn on.



(*) The LED is off if a firmware ERR is detected. If the CPMS has stopped, the LED state set by the CPMS is retained.

Figure 1-35 State Transitions During RUN

Table 1-12 CPMS States

No.	State	Description
1	OS initialization in progress	CPMS is being initialized.
2	LADDER STOP	The ladder program is stopped. The CPMS enters this state if any of the following conditions is met: • The LADDER RUN/STOP switch is in the STOP position. • The contact for the IF module's STOP/RUN contact input is on. • In BASE SYSTEMS/S10VE, **LADDER STATUS** is changed to `STOP` by going to **Online**, and operating the **Display PCs STATUS and change PCs STATUS** menu.
3	LADDER STOP (LADDER RUN delay)	The condition that caused LADDER STOP (the preceding No. 2) has been canceled, and remote I/O operation has started, but the ladder RUN operation is delayed. When the delay time has elapsed, the ladder RUN operation starts. The delay time setting (*n* times the sequence cycle time) is set in LADDER DIAGRAM SYSTEM/S10VE.
4	LADDER STOP (CP: Normal, HP: ERR)	The HP side has stopped due to an error.
5	LADDER RUN	A ladder RUN operation is in progress.
6	SIMU RUN	A simulation is running. This state occurs when in BASE SYSTEM/S10VE, **LADDER MODE** is changed from `NORM` to `SIMU` by going to **Online**, and operating the **Display PCs STATUS and change PCs STATUS** menu.

(2) State of the processor (CP and HP) in each state

The LED output of the CPU module indicates the state of the system. The processors CP and HP are paired in the controller system. As a result, the state of CPMS in both processors (CP and HP) is the same and changes in sync with the states in Figure 1-35.

- CPU STOP: The CPMS stops in the CP and HP.
- OS initialization in progress: The CPMS starts at the same time in the CP and HP, and each initial start task is started.

The following shows the correspondence between the CPMS operation in the CP and HP, and the system state:

- CP: This is a communication and control processor whose CPMS state matches that of the RUN LED. When the CP's CPMS is in the CPU STOP state, the HP's CPMS also enters the CPU STOP state, the ERR LED turns on, and the RUN LED turns off.
- HP: This is a control processor. CP continues operating even if the HP has stopped. Accordingly, the RUN LED stays on even if the HP's CPMS is in the STOP state.

7.4 PCsOK Output Control

The CPU has a contact used for indicating to the outside that the ladder STOP conditions have been canceled and remote I/O operation has started. The CPMS first turns the PCsOK output on (starts remote I/O operation), and then starts ladder execution. The PCsOK signal is used to send out the control start state to devices connected to the controller.
If the CPMS state is not LADDER STOP (LADDER RUN delay) or LADDER RUN, the PCsOK output is off.
Note that even if the PCsOK output is on, remote I/O does not operate if the contact for the RIO STOP contact input of the IF module is on.

Table 1-13 PCsOK Output by CPU State

State / Output signal name	PCsOK output
LADDER STOP	OFF
LADDER STOP (LADDER RUN delay)	ON
LADDER RUN	ON
SIMU RUN	OFF

7.5 Watchdog Timer (WDT)

7.5.1 WDT Functions

CPMS uses a watchdog timer (WDT) to monitor whether a task, ladder program, or HI-FLOW program is stuck in an infinite loop. A WDT can detect when the execution of a task, ladder program, or HI-FLOW program takes too long, making it too late to execute plant control. If the WDT (*) times out, the hardware turns on the ERR LED, and PCsOK turns off. In the CP, the built-in subroutine WDTES is linked (called) by a WDT timeout interrupt. The user can register an error handling program to WDTES. In addition, a CPU STOP instruction can be issued based on the return value of WDTES.

If WDTES is not registered, the CPMS does not abort the task or stop the CPU even if the WDT times out.

In the HP, WDT control is managed by the CPMS, and consequently, an error handling program cannot be registered to WDTES.

(*) If the WDT in the CPU (HP) times out, the ERR LED turns on, and PCsOK turns off. The LEDs and PCsOK do not change even if the WDT in the CPU (CP) times out.

7.5.2 Using the WDT

If you want to use a WDT in the CP, one task monitoring the execution time must issue the WDT control macro (WDTSET) at intervals shorter than the time set in the WDT. If the execution time of the macro-issuing task or other, higher-priority tasks exceed the expected value, the WDT setting is not updated by this task, and the WDT times out.

On startup of the initial start task, the WDT is not yet started. The WDT starts when the user program first issues a WDTSET macro. If the time is set to `0` by using the WDTSET macro, the WDT stops without timing out.

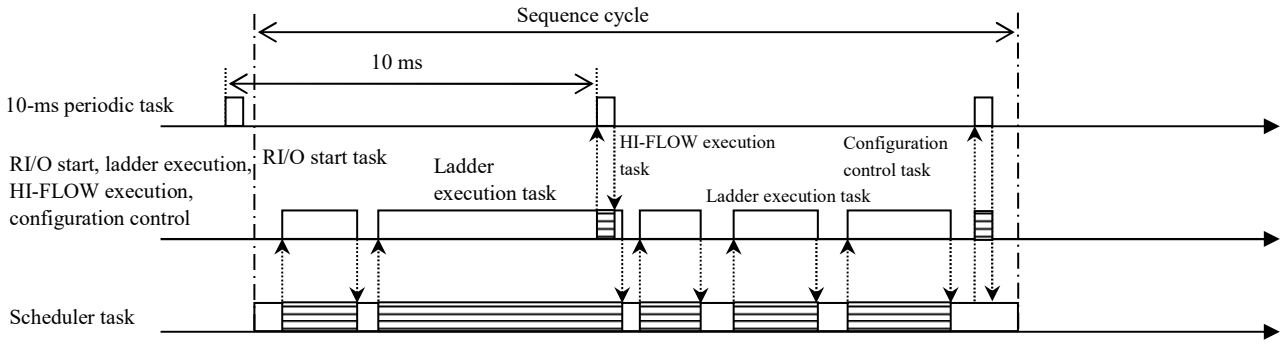
The CPMS only uses one WDT, so only one task at a time can undergo execution time monitoring by the WDT. Therefore, set one task to monitor tasks involved in plant control, and have WDT monitor this monitoring task.

To use the WDT in the HP, set it by using LADDER DIAGRAM SYSTEM/S10VE, and perform either a reset or a power cycle. The WDT in the HP monitors the sequence cycles of the ladder diagram, and as a result, monitors the sum of the execution times of the ladder program and HI-FLOW program.

7.6 Scheduler

On the HP, the tasks are managed by the 10-ms periodic task, and the schedules for non-periodic tasks such as ladder and HI-FLOW are managed by the scheduler task. The following gives an overview of how this is done:

(1) The system initial start task (SIST) of the HP starts the 10-ms periodic task and scheduler task as periodic start tasks of timer.
(2) The 10-ms periodic task executes the 10-ms, 100-ms, and 1-s processing.
(3) The scheduler task starts the WDT and executes remote I/O start, ladder, HI-FLOW, and tasks for ladder execution and configuration control.
(4) The remote I/O start task executes remote I/O processing.
(5) The ladder execution task executes ladder processing.
(6) The HI-FLOW execution task executes HI-FLOW.
(7) The second ladder execution task executes ladder processing related to remote I/O.
(8) The configuration control task configures the system register.



7.6.1 Ladder execution task

This task executes a ladder program created in LADDER DIAGRAM SYSTEM/S10VE. For details about ladder programs, see the *S10VE Software Manual Programming Ladder Diagram System for Windows®* (manual number SEE-3-121) and *S10VE Software Manual Operation Ladder Diagram for Windows®* (manual number SEE-3-131).

7.6.2 HI-FLOW execution task

This task executes a HI-FLOW program created in HI-FLOW SYSTEM/S10VE. For details about the HI-FLOW program, see the *S10VE Software Manual Programming HI-FLOW for Windows®* (manual number SEE-3-122) and *S10VE Software Manual Operation HI-FLOW for Windows®* (manual number SEE-3-132).

7.6.3 Configuration control task

This task performs configuration control. For details, see 7.7 Configuration Control.

7.6.4 RI/O start task

This task performs remote I/O communication. For details, see CHAPTER 12 REMOTE I/O in PART 1.

7.7 Configuration Control

Configuration control is performed in system tasks, and applies the control state to the system register.

(1) System register for configuration control

To reference the state of configuration control, use this system register.

- CPU status register (S0BF0 to S0BFF)

  This register applies the CPU status.

Table 1-14 CPU Status Register

Bit register	Description	ON (1)	OFF (0)	Read/Write (*1)
S0BF0	CPU state	LADDER STOP	LADDER RUN	R
S0BF1	Simulation state	Simulation	RUN	R
S0BF2	For future use	–	–	R
S0BF3	State of protection switch	ON state	OFF state	R
S0BF4	Remote I/O operation	In operation	Stopped	R
S0BF5	For future use	–	–	R
S0BF6	Ladder reprogram	Reprogram in progress	Reprogram completed	R
S0BF7	For future use	–	–	R
S0BF8	Primary battery state	Low voltage	Normal	R
S0BF9	Station timeout	Yes	No	R
S0BFA	Blown fuse detected	Yes	No	R
S0BFB	Optional module error (*2)	Yes	No	R
S0BFC	For future use	–	–	R
S0BFD	Cleared to 0 at power recovery and reset	–	–	R/W
S0BFE	Scheduler operation state	Stopped	Normal operation	R
S0BFF	CP operation state	CP stopped	CP normal operation	R

(*1) R: You can only read the register. Do not write to the register.
R/W: You can both read from and write to the register.

(*2) This means that a parity error occurred when the CPU attempted to access the memory in an optional module.

CHAPTER 8 TASK ERROR HANDLING

The following lists the basic concepts behind error handling during task execution:
- When an error is detected in a task, execution of the task is canceled. However, you can continue task execution by returning the task to its recovery point (see *8.5 Recovering from program errors*).
- In the case of a hardware error that cannot affect task execution, the task continues its execution. The task can obtain hardware error information and perform error handling.
- A built-in subroutine handles task errors. The task number of the task where the error occurred is returned with the input parameters of the built-in subroutine.

8.1 Repertoire of Built-in Subroutines

The CPMS imposes some rules on built-in subroutines so that the user can create part of the system processing.

A total of four entries are allowed for each built-in subroutine: two for middleware, two for the OS, and two for the user.

Entry numbers 1 and 2 are assigned for middleware and the OS, while numbers 3 and 4 are for the user. Entries are linked in ascending order of entry numbers: 1, 2, 3, 4.

Table 1-15 Repertoire of Built-in Subroutines

Built-in subroutine name	When link is made	Input information	Output information	Can issue macros?	Number of entries
CPES	When a program error occurs	PRGEB	Available	Yes	4
IES	When an I/O error occurs	IOERB	Available	Yes	4
EAS	When an error is logged	ADB	Available	Yes	4
INS	Before IST starts	Start factor	None	No	4
EXS	When exiting a task	Task number	None	Yes	4
ABS	When aborting a task	Task number	None	Yes	4
PCKS	When a macro parameter error occurs	SVCEB	Available	Yes	4
MODES	When a module error occurs	HARDEB	Available	Yes	4
WDTES	When a WDT timeout occurs	None	Available	Yes	4
XEAS	When an error occurs in another PU or XPU in the CPU	ADB	None	Yes	4

8.2 Execution Environment of Built-in Subroutines

The CPMS executes built-in subroutines in a system mode that prohibits interrupts. The execution priorities of all built-in subroutines are higher than those of any tasks.
The following restrictions are imposed on the execution environment of built-in subroutines:

- As a guideline, keep the size of the stack area used by built-in subroutines within 1 KB. When the stack area overflows, the CPU stops.
- Built-in subroutines are intended for event logging, accessing the GLB and CM, and starting and stopping other tasks. Do not perform processing where built-in subroutines being executed are made to wait or are stopped.
- To limit the time for which interrupts are prohibited, make sure that built-in subroutines are executed for one millisecond or less.
- Only the RLEAS, QUEUE, and ABORT macros can be called by built-in subroutines.
- Floating-point arithmetic operations cannot be used in built-in subroutines. Attempts to do so will stop the CPU.
- A programming error in a built-in subroutine stops the CPU.

8.3 Processing to Link Built-in Subroutines

Figure 1-36 shows how the built-in subroutines INS, ABS, EXS, CPES, PCKS, and WDTES are linked to EAS.

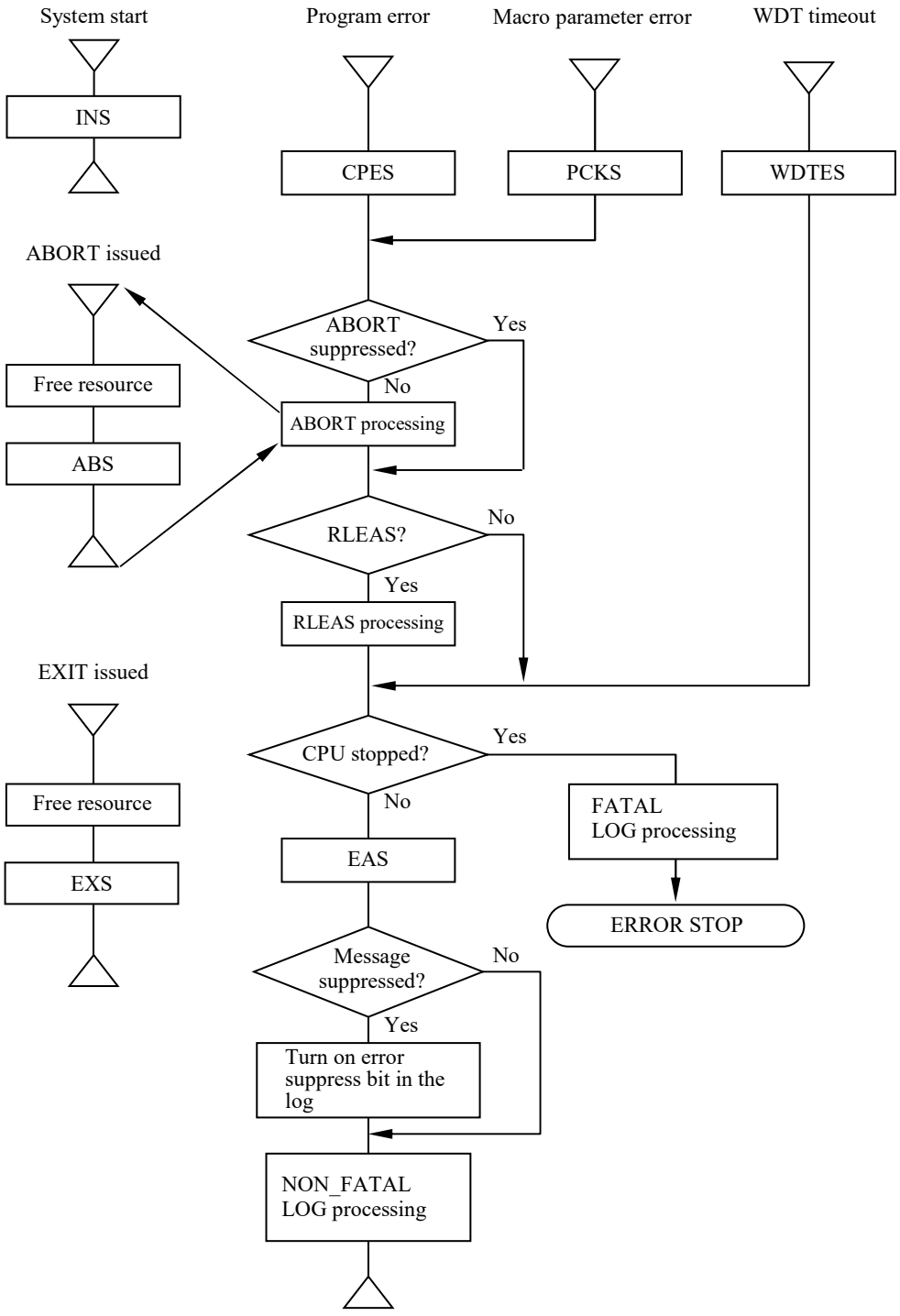


Figure 1-36 Processing to Link Built-in Subroutines (1)

Figure 1-37 shows how the built-in subroutines IES and MODES are linked to EAS.

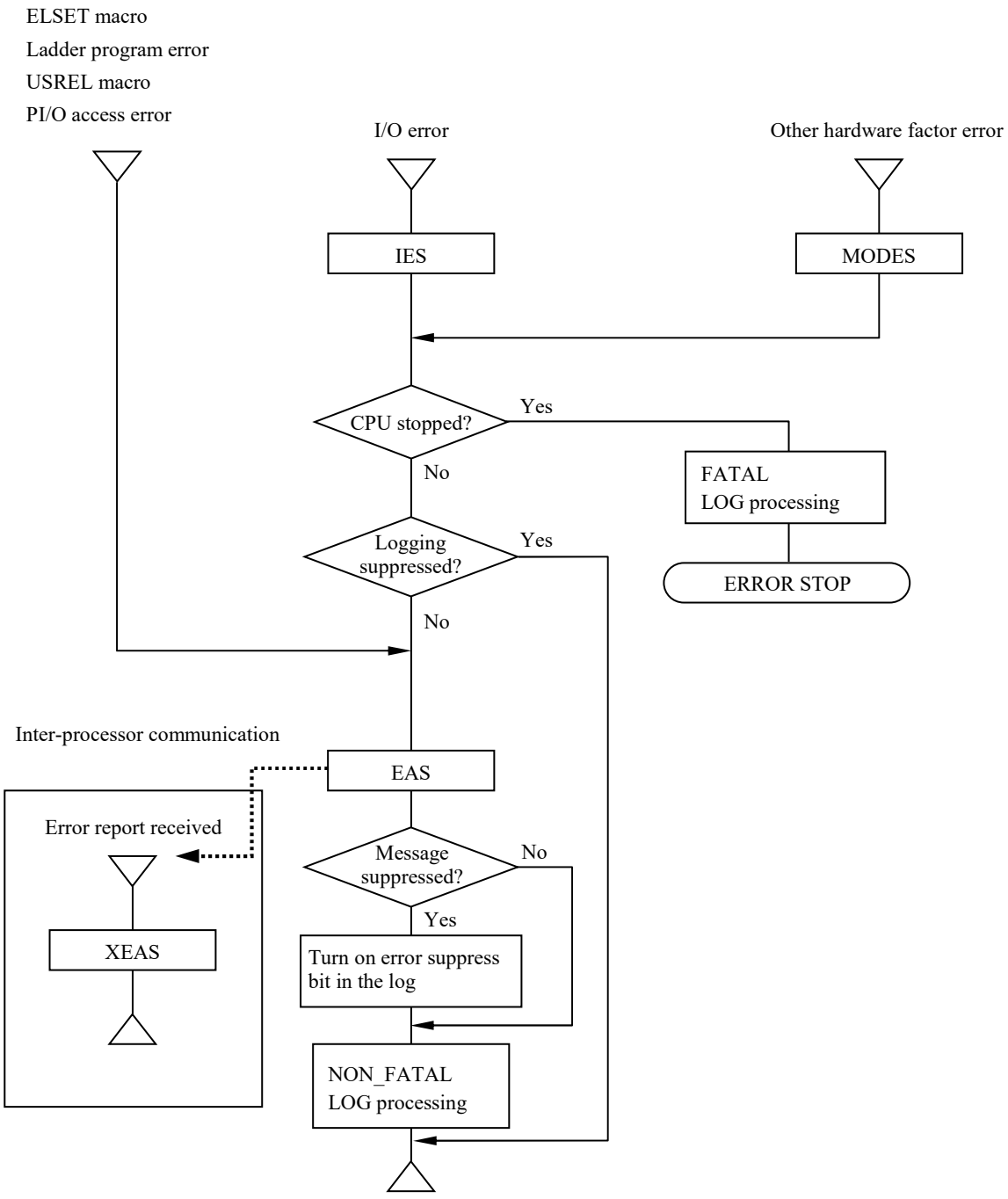


Figure 1-37 Processing to Link Built-in Routines (2)

8.4 Linkage of Built-in Subroutines

```
#include <cpms_ulsub.h>
```

- CPES - CPU Error Subroutine
```
int cpes(prgeb)
struct PRGEB *prgeb;  /* Program Error Block */
```
- IES - I/O Error Subroutine
```
int ies(ioerb)
struct IOERB *ioerb;  /* I/O Error Block */
```
- EAS - Error Alert Subroutine
```
int eas(adb)
struct ADB *adb; /* Alert Data Block */
```
- INS - Initial Start Subroutine
```
int ins(reset)
long reset; /* System start factor */
```
See *Table 1-11 Initiation Factors*.
- EXS - Exit Subroutine
```
int exs(tn)
long tn; /* Task Number */
```
- ABS - Abort Subroutine
```
int abs(tn)
long tn; /* Task Number */
```
- PCKS - Parameter Check Subroutine
```
int pcks(svceb)
struct SVCEB *svceb; /* SVC Error Block */
```
- MODES - Module Error Subroutine
```
int modes(hardeb)
struct HARDEB *hardeb;
```
- WDTES - WDT Error Subroutine
```
int wdtes( )
```
- XEAS - XPU Error Alert Subroutine
```
int xeas(adb)
struct ADB *adb;
```

Note 1: When CPES or PCKS is called, the task is aborted by default.
To suppress abortion of a task, turn on the ULSUB_OUT_ABORTSUPRES bit in the output information (however, do not use this bit for system tasks number 225 to 300).

Note 2: WDTES is called when a timeout error occurs in the system watchdog timer.
WDTES is not intended for monitoring termination of each task.

Note 3: For a system task, no link is made to either EXS or ABS.

NOTICE
If CPES or PCKS occurs in a system task (number 225 to 300), do not turn on the ULSUB_OUT_ABORTSUPRES bit in the output information. Doing so might result in unexpected operation, and the system might go down.

For input information, see APPENDIX D INPUT DATA FOR BUILT-IN SUBROUTINES.

Output information (return values)

All output information (returned information) from built-in subroutines is in the common format, and its meaning is determined by bits. Because there is more than one entry point, output information for each subroutine is ORed.

```
#define ULSUB_OUT_LOGSUPRES    0x00000010  /* Error logging is suppressed.   */
#define ULSUB_OUT_MSGSUPRES    0x00000020  /* Error messages are suppressed.*/
#define ULSUB_OUT_RLEAS        0x00000040  /* The task is released.     */
#define ULSUB_OUT_ABORTSUPRES 0x00000080  /* Task abortion is suppressed. */
#define ULSUB_OUT_CPUDOWN      0x00000100  /* The CPU goes down.        */
```

Table 1-16 shows which bits are enabled and disabled according to the type of built-in subroutine.

Bits that are enabled in the event that there is output information are indicated by a circle (○).

Table 1-16 List of Output Information from Built-in Subroutines

	CPES	IES	EAS	INS	EXS	ABS	PCKS	MODES	WDTES	XEAS
Output information available?	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	No
ULSUB_OUT_ABORTSUPRES	√	×	×	×	×	×	√	×	×	×
ULSUB_OUT_RLEAS	√	×	×	×	×	×	√	×	×	×
ULSUB_OUT_LOGSUPRES	×	√	×	×	×	×	×	√	×	×
ULSUB_OUT_MSGSUPRES	×	×	√	×	×	×	×	×	×	×
ULSUB_OUT_CPUDOWN	√	√	×	×	×	×	√	√	√	×

√: Enabled, ×: Disabled

(1) ULSUB_OUT_ABORTSUPRES

This bit suppresses abortion of the task specified by an argument. This bit is enabled with the CPES and PCKS built-in subroutines. When the ULSUB_OUT_CPUDOWN bit is on but the ULSUB_OUT_ABORTSUPRES bit is off, the task is aborted, and then the CPU is stopped.

(2) ULSUB_OUT_RLEAS

This bit releases the task specified by an argument. This bit is enabled with the CPES and PCKS built-in subroutines.

To abort or release a task, turn on the ULSUB_OUT_RLEAS bit, and turn off the ULSUB_OUT_ABORTSUPLES bit.

(3) ULSUB_OUT_LOGSUPRES

This bit is enabled with the IES and MODES built-in subroutines. When a PI/O or I/O access has been processed normally, turn this bit on. The on/off state of this bit is determined by the PI/O or I/O driver. When this bit is on, error logging is skipped.

(4) ULSUB_OUT_MSGSUPRES

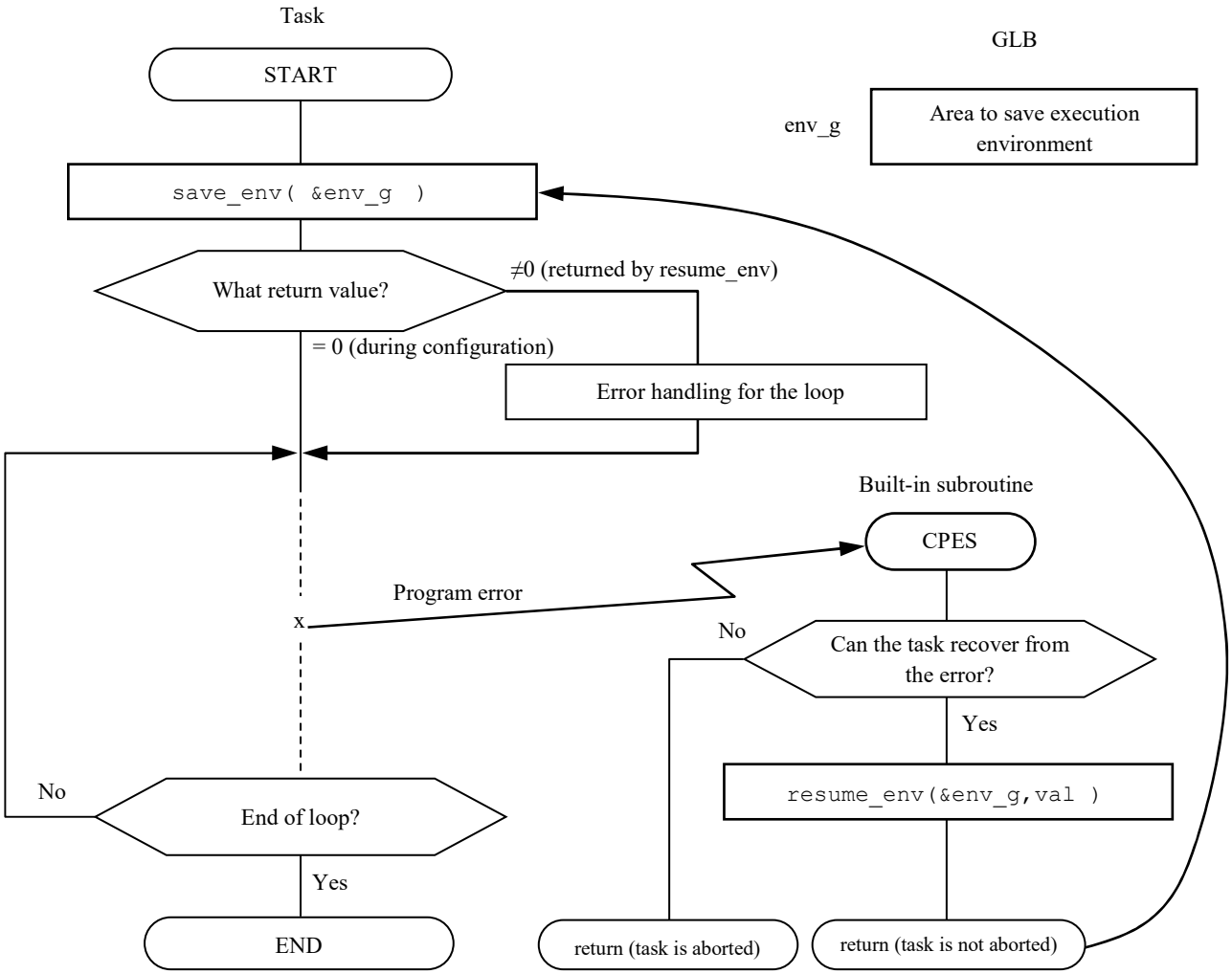
This bit is enabled only with the EAS built-in subroutine. When this bit is on, the message suppress flag in the error information is set to `1`. The actual message suppression processing is entrusted to the display program. Therefore, make sure that the display program performs processing according to the message suppress flag in the error information. This bit does not skip error logging.

(5) ULSUB_OUT_CPUDOWN

This bit stops the CPU (ERROR STOP).

8.5 Recovering from Program Errors

To prepare for any program errors in a task, set a recovery point in advance so that the task can return to this point to continue execution. Note that the recovery point is effective for program errors in the routine containing the recovery point.



• Call Courier to save the execution environment data at the recovery point in the GLB.

• Call Courier from the CPES built-in subroutine to return control to the recovery point after CPES is executed.

Figure 1-38 Recovering from Program Errors

(1) Method of use

- Call `save_env(&env_g)` to save the execution environment at the error recovery point in the `env_g` assigned to the GLB. At this time, `save_env` returns `0`.
- If a program error occurs in a task, link that task to the built-in subroutine CPES.
- If the task where the error occurred needs to be recovered, in the user built-in subroutine registered in the built-in subroutine CPES, use `resume_env (&env_g, val)` to specify to resume the task from the error recovery point, and make sure that the task is not aborted.
  Set `val` to any value other than 0.
- If the task returns from the built-in subroutine without being aborted, the task resumes from the recovery point. At this time, `save_env` returns val.
- From the code returned from `save_env`, the task can determine that control has been returned from CPES, and can execute postprocessing for the error.

(2) Notes

- When an error occurs, it is assumed that the stack contents remain the same as when the recovery point was set. The task cannot recover from a program error caused by a stack crash or program crash.
- When the task is recovered from an error by `resume_env`, external variables, static variables, and auto variables are not recovered. The loop where the error occurred can be determined, for example by the number of loops shown in Figure 1-38 remaining unchanged. Conversely, when these variables are corrupted by an error, processing cannot be continued correctly even after the task returns to the error recovery point.
- When CPES attempts recovery from an unrecoverable program error, an infinite loop might occur, repeating the program errors and recovery processing. Have CPES check whether the task is a recoverable task, and limit the program error conditions from which the task needs to be recovered.
- In `resume_env`, make sure that you specify the area (`env_g`) where the task where the error occurred used `save_env` to save the execution environment. If this is not done, the task cannot return to the recovery point correctly.

CHAPTER 9 SYSTEM SERVICES

9.1 DHP

Every time the CPMS passes a predetermined processing point, that fact is recorded in a buffer in the main memory. This record is called the *debugging helper* (DHP). The DHP buffer is located in the kernel work area. The CPMS supports DHP processing.

(1) Recording point

DHPs are recorded at the following points:

- As a rule, a DHP is recorded when all CPMS macros are issued, after the parameters are read.
  Note that when macro processing is complete, the processing result (for example, the initiation factor returned by GFACT to the user) might be recorded as a DHP.
- Before and after task switch processing
- Processing to start or end a task
- I/O start processing and end interrupt processing
- Task error handling
- OS/hardware error handling
- Issuing of a `usrdhp` macro instruction (recording of user information)

(2) Data to be recorded

At each DHP point, the following data is recorded:

- Code representing the DHP point (4 bytes)
- DHP recording time (4 bytes)
- Task number and task priority (2 bytes each)
- Data necessary for analysis (variable length from 0 to 20 bytes)

(3) Recording mode

By default, the system is always making records. You can control the stopping and resuming of DHP by using either the **RAS** menu of BASE SYSTEM/S10VE or the RPDP `svdhp` command.

(4) DHP buffer

The DHP buffer uses 128 KB of main memory by default.
A 128-KB area is allocated to the CP by default.
A 128-KB area is allocated to the HP by default.

(5) Output of record data

- The RPDP `svdhp` command can be used to load the current DHP data.
- Errors are logged together with the most recent DHP data.

9.2 PU Load Ratio

The user can measure the PU load ratio.

The SYS_IDLE function of the getsysinfo macro can be used to get the cumulative PU idle time. The idle time is accumulated continuously. Therefore, obtain the idle time from the difference between the cumulative idle time when SYS_IDLE was last issued, and the current cumulative idle time.

When measuring this difference between idle times, do not change the clock of the PU. If you do so, you cannot obtain the correct difference between the two idle times.

The following is the formula for calculating the PU load ratio:

PU load ratio = (Measurement time - difference between current and previous cumulative idle times)/measurement time

CHAPTER 10 NOTES ON MIGRATING FROM THE S10V

Note the following when migrating programs created in the C language and operated in S10V CMU to an S10VE CP:

- Macro specifications
  There are some differences in the specifications for macros supported by the S10VE, and those supported by S10V CMU.
  You must check the parameters and return code information, and make any necessary corrections.
  See APPENDIX B DIFFERENCES IN MACRO SPECIFICATIONS FROM S10V CMU.
  For details about the specifications, see PART 2 MACRO SPECIFICATIONS.

CHAPTER 11 OPTIONAL MODULES

11.1 Optional Modules

You can set up and operate optional modules by using the setup tool. For details about the specifications and information on how to use an optional module, see its corresponding manual.

11.2 Automatic Setup Function for Optional Module Parameters

The CPMS allows modules to be changed without having to perform setup again by using the setup tool. Therefore, setting parameters for optional modules are stored in the CPU. On startup, these settings are compared with the parameters set in the optional module. If the settings do not match, the parameters stored in the CPU are written to the optional module automatically.

After parameters are written to an optional module automatically, either the optional module alone is reset, or the entire unit is reset.

When the automatic parameter setup function is activated, execution of user applications such as LADDER is delayed until setup finishes (the system might take a few minutes on startup, depending on the parameter write size).

(1) System registers for automatic parameter setup for optional modules

The following table lists system registers that store information relating to the automatic parameter setup function for optional modules.

- Optional module parameter validity register (S0F00 to S0F0F)

  This register indicates the validity of specified optional module parameters.

Table 1-17 Parameter Validity Register for Optional Modules

Bit register	Description	ON (1)	OFF (0)	Read/Write (*)
S0F00	State of parameter 1	Valid	Invalid	R
S0F01	State of parameter 2			
S0F02	State of parameter 3			
S0F03	State of parameter 4			
S0F04	State of parameter 5			
S0F05	State of parameter 6			
S0F06	State of parameter 7			
S0F07	State of parameter 8			
S0F08	State of parameter 9			
S0F09	State of parameter 10			
S0F0A to S0F0F	For future use	–	–	–

(*) R: You can only read the register. Do not write to the register.

- Error register for writing optional module parameter settings (S0F10 to S0F1F)
  This register indicates whether the result of writing parameter settings to the optional module and restarting the optional module was successful (ended normally or with an error).

Table 1-18 Error Register for Writing Optional Module Parameter Settings

Bit register	Description	ON (1)	OFF (0)	Read/Write (*)
S0F10	Result of writing parameter 1	Write error occurred	Write ended normally	R
S0F11	Result of writing parameter 2			
S0F12	Result of writing parameter 3			
S0F13	Result of writing parameter 4			
S0F14	Result of writing parameter 5			
S0F15	Result of writing parameter 6			
S0F16	Result of writing parameter 7			
S0F17	Result of writing parameter 8			
S0F18	Result of writing parameter 9			
S0F19	Result of writing parameter 10			
S0F1A to S0F1F	For future use	–	–	–

(*) R: You can only read the register. Do not write to the register.

11.3 Function for Applying the Operation State of Optional Modules

The CPMS applies the operation/stop state of optional modules to the system register.
For information on an optional module that writes its operation/stop state directly to the system register, see the system register written in the corresponding manual of that optional module.
Table 1-19 lists the system registers assigned to apply the operation/stop state of optional modules.

Table 1-19 Registers for Optional Module Operation/Stop States

No.	Word register	Optional module
1	SW0D00	FL.NET (main module)
2	SW0D10	FL.NET (submodule)
3	SW0D20	OD.RING (main module)
4	SW0D30	OD.RING (submodule)
5	SW0D40 to SW0EF0	For future use

- Register for optional module operation/stop states (S0xx0 to S0xxF)
  Table 1-20 shows the bit assignments for the aforementioned word registers.

Table 1-20 Bit Assignments for Optional Module Operation/Stop State Registers

Bit register (*2)	Description	ON (1)	OFF (0)	Read/Write (*1)
S0*xx*0	Error stop state	Stopped with an error	Did not stop with an error	R
S0*xx*1 to S0*xx*F	For future use	–	–	–

(*1) R: You can only read the register. Do not write to the register.
(*2) *xx*: This stores the aforementioned word register assignments for the target optional module.
For example, in the case of FL.NET (main module), S0D00 indicates an error stop state.

11.4 Error Log Application Function for Optional Modules

Log information on errors detected in optional modules is stored in the CPMS error log.
For the stored error codes, see APPENDIX C LIST OF ERROR MESSAGES.

11.5 Function for Applying Mount States of Optional Modules

In the S10VE, the optional module mount state is applied to the system register.
Table 1-21 shows the system registers assigned to apply the mount states of optional modules.

Table 1-21 Optional Module Mount State Register

No.	Bit register	Optional module
1	SW3010	OD.RING
2	SW3020	FL.NET
3	SW3030	J.NET
4	SW3040	2ch D.NET
5	SW3050 to SW30D0	For future use
6	SW30E0	ET.NET
7	SW30F0 to SW3EF0	For future use

- Optional module mount state register (S0xx0 to S0xxF)
  Table 1-22 shows the bit assignments for the aforementioned word register.

Table 1-22 Bit Assignments for the Optional Module Mount State Register

Bit register (*2)	Description	ON (1)	OFF (0)	Read/Write (*1)
S3*xx*0	Module 0 or main module mount state	Mounted	Not mounted	R
S3*xx*1	Module 1 or submodule mount state	Mounted	Not mounted	R
S3*xx*2	Module 2 mount state	Mounted	Not mounted	R
S3*xx*3	Module 3 mount state	Mounted	Not mounted	R
S3*xx*4 to S3*xx*F	For future use	–	–	–

(*1) R: You can only read the register. Do not write to the register.
(*2) *xx*: This stores the aforementioned word register assignments for the target optional module. For example, in the case of FL.NET (main module), S3020 indicates a mount state.

CHAPTER 12 REMOTE I/O

12.1 Remote I/O

This chapter describes the remote I/O function built into the CPU module.
The remote I/O function supports connection to HSC-1000 and HSC-2100 PI/O.
Table 1-23 shows the specifications for the remote I/O function.

Table 1-23 List of Specifications for the Remote I/O Function

No.	Item	Specification
1	Number of channels	2 (1024 points per channel)
2	Number of transfer points	You can select from 64, 128, 256, 512, 1024, 1536, and 2048 points.
3	Transfer speed	768 kbps
4	Number of connected stations	Up to 12 per line
5	Connectable stations	Stations for HSC-2100 and HSC-1000
6	Function for turning ladder synchronization on and off	You can select whether to use ladder synchronization.
7	Function for selecting whether to connect an optical adapter	You can select whether to connect an optical adapter.
8	Analog and pulse counter module support	You can enable conversion between an XW/YW area and an EW area for AI/AO data and pulse counter data.
9	System register	The operation state of remote I/O is applied to the system register.

Remote I/O operates only when the PCsOK signal output is on and the remote I/O stop input signal is off. Table 1-24 lists combinations of the PCsOK signal output state, ladder operation, and remote I/O operation.

Table 1-24 List of Ladder Operation and Remote I/O Operation Combinations

No.	RI/O STOP input signal state for the I/F module	Ladder operation state	PCsOK signal output state	Remote I/O operation
1	OFF(RUN)	STOP	OFF	Stopped
2		RUN	ON	Running
3		SIMU RUN	OFF	Stopped
4	ON(STOP)	STOP	OFF	Stopped
5		RUN	ON	Stopped
6		SIMU RUN	OFF	Stopped

12.2 Function for Turning Ladder Synchronization On and Off

You can use the function for turning ladder synchronization on and off to select whether to synchronize remote I/O operation with one ladder cycle. The following describes remote I/O operation with and without ladder synchronization.

You can turn ladder synchronization on or off in LADDER DIAGRAM SYSTEM/S10VE by going to **PCs edition**, and then using the **PCs edition** menu.

For details about LADDER DIAGRAM SYSTEM/S10VE operation, see *S10VE Software Manual Operation Ladder Diagram System for Windows®* (manual number SEE-3-131).

(1) When not using ladder synchronization

Remote I/O operates without synchronizing with ladder operation. Remote I/O starts the next transfer immediately after transfer of the specified number of points is complete.

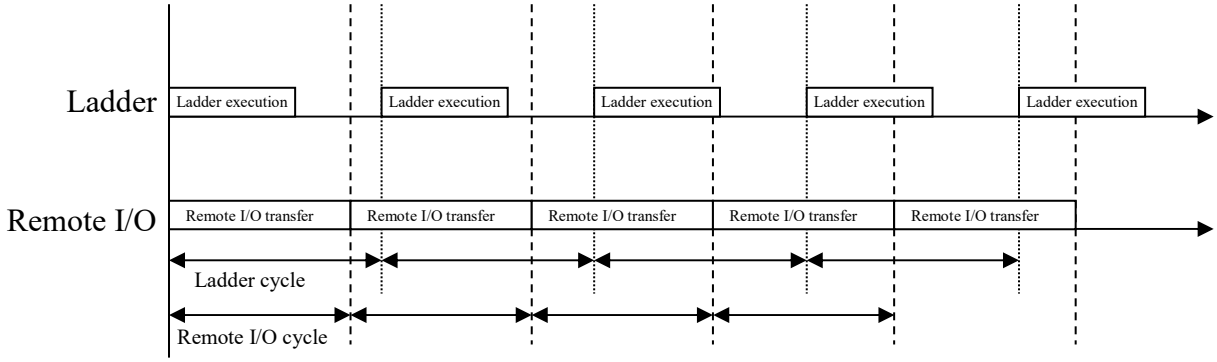


Figure 1-39 Timing Chart for when Ladder Synchronization is Off

(2) When using ladder synchronization
Remote I/O makes one transfer per ladder execution. The input area (XW area) for remote I/O does not change during ladder execution. Even if a value in the output area (YW area) changes, the output state of the output module (DO, AO, pulse counter) does not change during ladder operation. The input and output areas are updated when ladder execution and remote I/O transfer are complete.

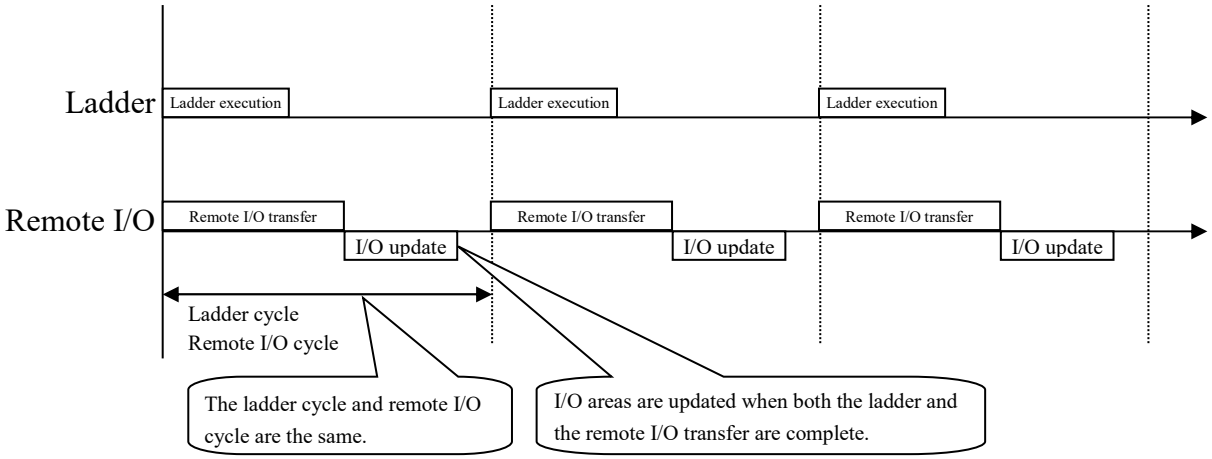


Figure 1-40 Timing Chart for when Ladder Synchronization is On

If the ladder sequence cycle is longer than the timeout detection time of the remote I/O station (*), the station detects a timeout at every sequence cycle. This means that if you want to enable the ladder synchronization mode, you must specify a sequence cycle shorter than the timeout detection time of the station.
(*) For information about the timeout detection time of a remote I/O station, see the manual of the station module.

12.3 Function for Selecting Whether to Connect an Optical Adapter

Use this setting to select whether to connect a remote I/O optical adapter (module model: LQZ410) to remote I/O. More specifically, the user must use a tool to select whether to use a remote I/O optical adapter (the default setting is not to use a remote I/O adapter).
The only difference in operation between using and not using a remote I/O optical adapter is in the timeout detection time. The timeout detection time is the time for which the system is monitored after a transfer finishes, until reception finishes normally.

Table 1-25 List of Timeout Detection Times

No.	Remote I/O optical adapter	Timeout detection time (hardware specification)
1	Not used	93.6 μs
2	Used	136.7 μs

Figure 1-41 shows a configuration that uses a remote I/O optical adapter.

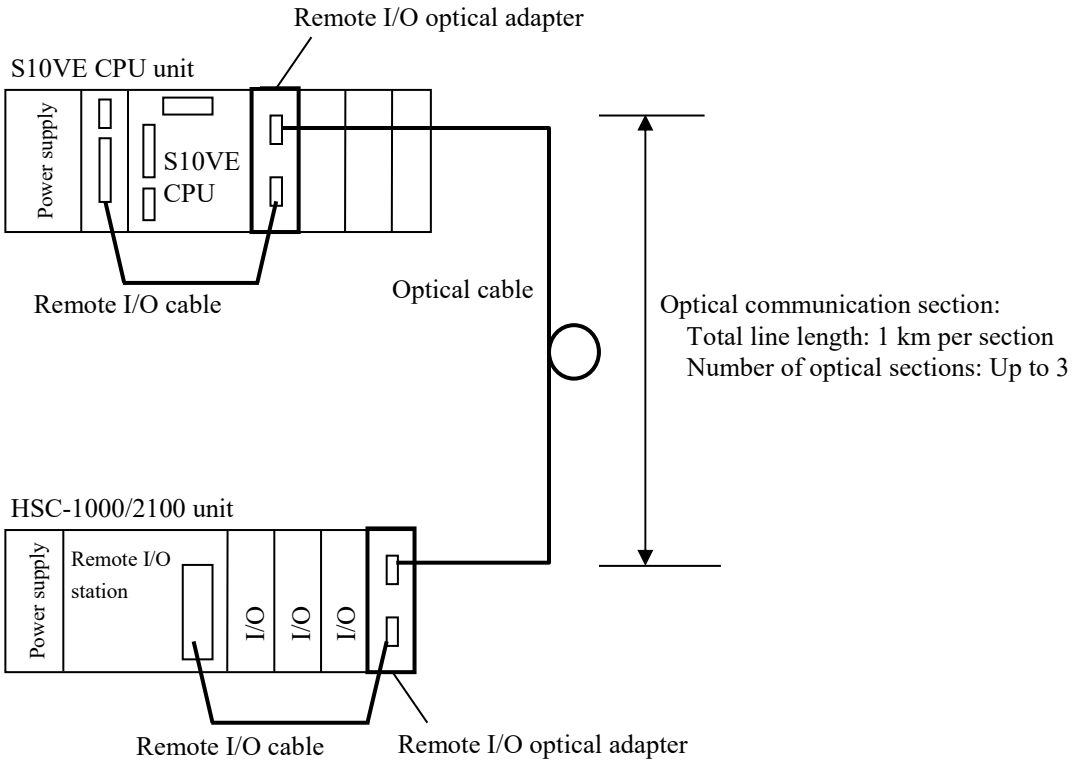


Figure 1-41 Example Configuration Using an Optical Adapter

12.4 Analog and Pulse Counter Support Function

This function converts 4ch AI/AO, 8ch AI data, and pulse counter data passing through remote I/O. Table 1-26 lists the supported functions.

Table 1-26 List of Analog and Pulse Counter Support Functions

No.	Supported module	Function
1	4ch AI (12-bit)	Data in an XW area is shifted to the right by 4 bits (sign extension) and copied to an EW area.
2	4ch AI (14-bit)	Data in an XW area is shifted to the right by 2 bits and copied to an EW area.
3	8ch AI (12-bit)	Data in an XW area is shifted to the right by 4 bits (sign extension) and copied to an EW area.
4	4ch AO (12-bit)	Data in an EW area is shifted to the left by 4 bits and copied to a YW area.
5	Pulse counter	The control code and count data in an EW area are concatenated and copied to a YW area. The data in an XW area is divided into a state code and counter data and copied to an EW area.

Table 1-27 shows the relationships between setting names (module names) in the CPMS support module and in the ladder diagram system.

Table 1-27 Relationships Between Setting Names in the Support Module and Ladder Diagram System

No.	Supported module	Setting in the ladder diagram system (module name)	Remarks
1	4ch AI (12-bit)	4ch AI (12-bit)	
2	4ch AI (14-bit)	4ch AI (14-bit)	
3	8ch AI (12-bit)	8ch AI (12-bit) (MODE2)	Eight channels' worth of data deployed to a continuous segment of an EW area
		8ch AI (12-bit) (MODE4) (ch0-ch3)	Eight channel's worth of data deployed to only four channels' worth of EW area (channels 0 to 3)
		8ch AI (12-bit) (MODE4) (ch4-ch7)	8 channels' worth of data deployed to only 4 channels' worth of EW area (channels 4 to 7)
4	4ch AO (12-bit)	4ch AO (12-bit)	
5	Pulse counter	PCT (MODE1)	No sign extension used when input count data is deployed
		PCT (MODE2)	Sign extension used when input count data is deployed

(1) 4ch AI data (12-bit)

Data input from remote I/O to an XW area is shifted 4 bits to the right (sign extension) and copied to the EW area corresponding to the channel number. Only one channel's worth of data is updated per remote I/O transfer. Consequently, it takes four remote I/O cycles to update the data of all channels.

XW area

15	14 ... 4	3	2	1 0
S	Analog/digital conversion data	E	0	Channel number

S: Sign, E: Enable bit (0 = Enabled, 1 = Disabled)

Conversion specifications

When the enable bit is enabled (= 1): Data in the XW area is shifted 4 bits to the right and copied to the EW area corresponding to the channel number.

When the enable bit is disabled (= 0): 0x8000 (data disabled) is set in the EW area.

EW area

	15	14	13	12	11	10 ... 0
Channel 0	S	S	S	S	S	Analog/digital conversion data
Channel 1	S	S	S	S	S	Analog/digital conversion data
Channel 2	S	S	S	S	S	Analog/digital conversion data
Channel 3	S	S	S	S	S	Analog/digital conversion data

S: Sign

(2) 4ch AI data (14-bit)

Data input from remote I/O to an XW area is shifted 2 bits to the right (sign extension) and copied to the EW area corresponding to the channel number. Only one channel's worth of data is updated per remote I/O transfer. Consequently, it takes four remote I/O cycles to update the data of all channels.

XW area

15	14 … 2	1 … 0
S	Analog/digital conversion data	Channel number

S: Sign

Conversion specifications

Input data ≠ 0x8000: Data in the XW area is shifted 2 bits to the right and copied to the EW area corresponding to the channel number.

Input data = 0x8000: `0x8000` (data disabled) is set in the EW area.

EW area

	15	14	13	12 … 10	… 0
Channel 0	S	S	S	Analog/digital conversion data	
Channel 1	S	S	S	Analog/digital conversion data	
Channel 2	S	S	S	Analog/digital conversion data	
Channel 3	S	S	S	Analog/digital conversion data	

S: Sign

(3) 8ch AI data (12-bit)

Data input from remote I/O to an XW area is shifted to the right by 4 bits (sign extension) and copied to the EW area corresponding to the channel number. Only one channel's worth of data is updated per remote I/O transfer. Consequently, it takes eight remote I/O cycles to update the data of all channels.

XW area

15	14 … 4	3	2 … 0
S	Analog/digital conversion data	E	Channel number

S: Sign, E: Enable bit (0: Enabled, 1: Disabled)

Conversion specifications

When the enable bit is enabled (= 1): Data in the XW area is shifted 4 bits to the right and copied to the EW area corresponding to the channel number.

When the enable bit is disabled (= 0): 0x8000 (data disabled) is set in the EW area.

EW area

	15	14	13	12	11	10 … 0
Channel 0	S	S	S	S	S	Analog/digital conversion data
Channel 1	S	S	S	S	S	Analog/digital conversion data
				:		
Channel 7	S	S	S	S	S	Analog/digital conversion data

S: Sign

(4) 4ch AO data (12-bit)

Data input to an EW area is shifted 4 bits to the left and copied to an EW area with a channel number added. Only one channel's worth of data is updated per remote I/O transfer. Consequently, it takes four remote I/O cycles to apply the data of all channels to the AO.

EW area

	15	14	13	12	11	10 … 0
Channel 0	S	S	S	S	S	Analog/digital conversion data
Channel 1	S	S	S	S	S	Analog/digital conversion data
Channel 2	S	S	S	S	S	Analog/digital conversion data
Channel 3	S	S	S	S	S	Analog/digital conversion data

S: Sign

Conversion specification

For one remote I/O cycle, one channel's worth of data in the EW area is shifted to the left by 4 bits and copied to the YW area with a channel number added.

For each completed remote I/O transfer, the channel number is incremented by one. If the channel number is 3, it is returned to 0.

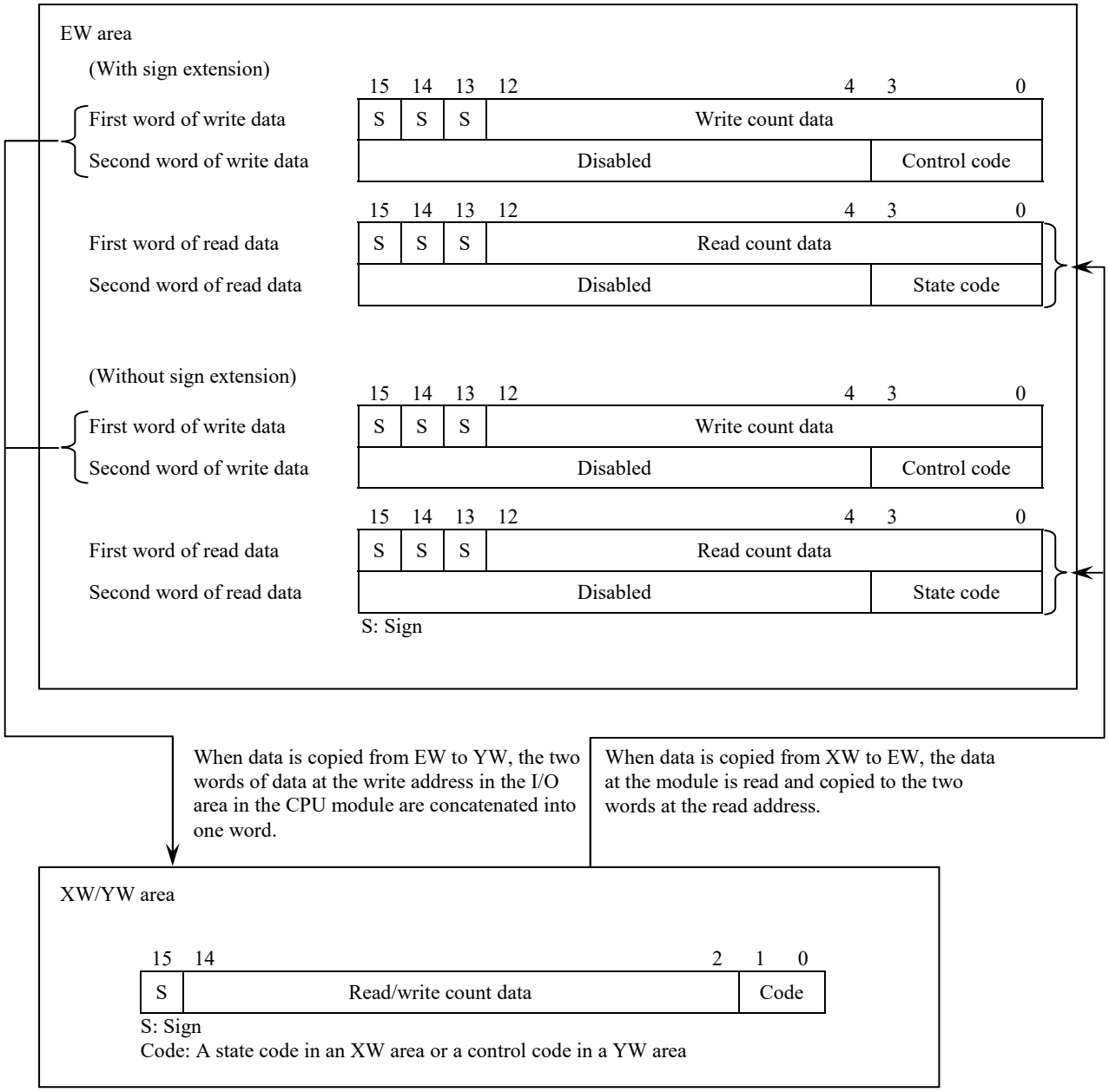
YW area

15	14 … 4	3	2	1 0
S	Analog/digital conversion data	0	0	Channel number

S: Sign

(5) Pulse counter data

The first word of the write data in an EW area is shifted to the left by 2 bits, the control code in the second word is converted to 2 bits and concatenated to the data, and the data is copied to a YW area.
The data in an XW area is shifted to the right by 2 bits and copied to the first word of the read data in an EW area. The code in the XW area is converted to a state code and copied to the second word of the read data in the EW area.



Conversion table between codes and control codes/state codes

No.	Code	Control code/State code
1	0	8
2	1	4
3	2	2
4	3	1

(6) EW area to be used

Up to 64 analog pulse counters can be registered.

The following lists the registration numbers in the ladder diagram system and their corresponding EW areas.

Registration no.	Corresponding EW area
1	EW0400
2	EW0480
3	EW0500
4	EW0580
5	EW0600
6	EW0680
:	:
24	EW0F80
25	EW1000
26	EW1080
27	EW1100
:	:
32	EW1380
33	EW1400
:	:
48	EW1B80
49	EW1C00
:	:
63	EW2300
64	EW2380

12.5 System Registers for Remote I/O

Table 1-28 shows the system registers used in remote I/O.

Table 1-28 System Registers for Remote I/O

System register	Name	Description
S0300 to S037F	Registered station	Bits corresponding to stations that have returned a normal response at least once after the CPU is reset or power-cycled are turned on. All bits are turned off when the CPU is reset or power-cycled.
S0380 to S03FF	Timed-out station	Bits corresponding to stations where hardware has reported a timeout are turned on. If a timeout is detected three times in a row at 100-ms intervals while not synchronized with remote I/O transfer cycles, the corresponding bit is turned on. If a recovery is detected at least once, the corresponding bit is turned off. All bits are turned off when the CPU is reset or power-cycled.
S0400 to S047F	Station with blown fuse	Bits corresponding to stations where hardware has reported a blown fuse are turned on. The timing for turning bits on or off is the same as for timed-out stations. All bits are turned off when the CPU is reset or power-cycled.

Table 1-29 lists the correspondences between remote I/O stations' I/O areas (X or Y numbers) and system registers.

Table 1-29 Correspondence List Between Remote I/O Stations and System Registers

X or Y number	Registered station	Timed-out station	Station with blown fuse
0000 to 000F	S0300	S0380	S0400
0010 to 001F	S0301	S0381	S0401
0020 to 002F	S0302	S0382	S0402
0030 to 003F	S0303	S0383	S0403
:	:	:	:
07C0 to 07CF	S037C	S03FC	S047C
07D0 to 07DF	S037D	S03FD	S047D
07E0 to 07EF	S037E	S03FE	S047E
07F0 to 07FF	S037F	S03FF	S047F

PART 2 MACRO SPECIFICATIONS

CHAPTER 1 OVERVIEW

1.1 Macro Instructions

Macro instructions are used to send requests from user programs (tasks) to the CPMS for processing. In a user program, macro instructions are written as subroutine calls. These subroutines are automatically expanded into trap instructions, which are CPMS calling instructions, by using the CPMS macro linkage library. When the program issues a macro instruction, that macro instruction is linked to the CPMS by using a trap instruction, and CPMS processing is performed.

1.2 CPMS Macro Linkage Library

The CPMS macro linkage library is a subroutine for expanding a macro instruction written in a user program to a trap instruction when a CPMS macro instruction is used. When the CPMS macro linkage library is called, it stores parameters (arguments) in the user stack in the specified order for each macro instruction, and then issues a trap instruction (see Figure 2-1).

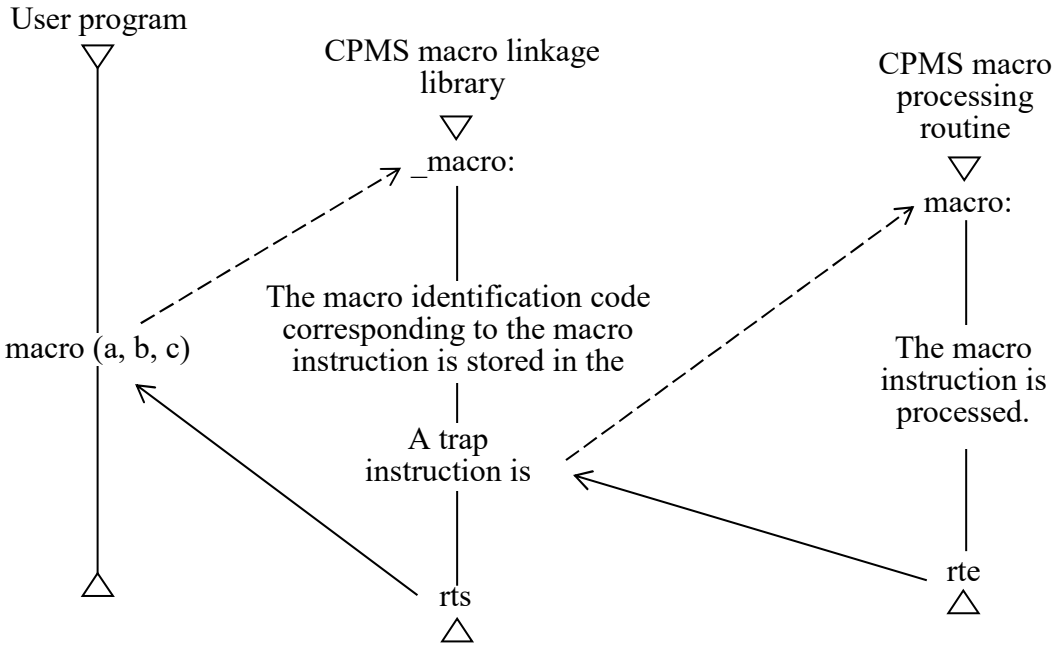


Figure 2-1 Workings of the CPMS Macro Linkage Library

1.3 General Rules for Macro Instructions

(1) Passing parameters

When the CPMS macro linkage library is used, parameters are passed in the form of addresses or their contents. For example, when creating a user program in the C language, make the following corrections:

```
long tn ;
tn = 100 ;
abort ( &tn ) ;
```

When the ABORT macro instruction is used, the address containing tn (= 100) is written in an argument (&tn indicates the pointer to tn or the address storing tn). This must not be written as abort (tn).

If you are using the C language, there are various other writing methods available. Code programs by using a method you find easy. The following are some coding examples:

- When the whole array is composed of parameters:

```
long  x[n] ;
macro  (x) ;
```

- When parameters are one of multiple elements in the array (the lower three are equivalent):

```
long  x[n] ; long  x[n] ;  long  *x[i] ;
x[i]=100 ;   x[i]=100 ;    *x[i]=100 ;
macro(&x[i]) ;      macro(x+i) ;  macro(x[i]) ;
```

- When parameters are simple variables (the lower two are equivalent):

```
long  x ;     long  *x ;
x=100 ;       * x=100 ;
macro(&x) ;   macro(x) ;
```

(2) Return code

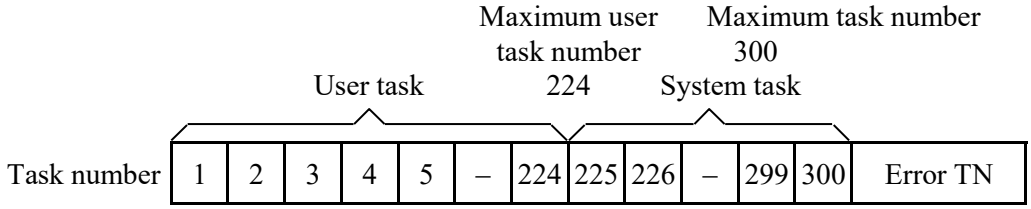
The result of executing a macro instruction is returned as a return code from the CPMS macro processing module. When a macro instruction is used as a function, the result of processing the macro instruction can be determined by the return code as shown in the following:

```
long macro( ) , rtn ;
long *x ;

rtn = macro(x) ;
if(rtn)
 {
   :
   :
   :
 }
```

Note: When a macro instruction is processed normally, the return code is usually 0. However, depending on the macro instruction, a value other than 0 might be returned even if the instruction is processed normally.

1.4 Checking Macro Instruction Parameters

Macro instructions are direct data exchanges between user programs and the CPMS. If a parameter is wrong, this might cause a system malfunction or system failure.
For CPMS macro instructions, software is used to make a rationality check on major parameters. If the check determines any parameters to be irrational, this is reported as a macro parameter error, and the task issuing the macro instruction is aborted.
Figure 2-2 shows the relationships among task numbers (TNs) when a parameter check is made.
In the parameter check for each macro, the maximum user task number is 224. A memory protect check is made in accordance with the information (accessibility/inaccessibility) set by the CPMS in each task.



Target TN / Issuing TN	1 to 224	225 to 300	301 onwards
1 to 224	Y	?	N
225 to 300	Y	Y	N

When a task that has an issuing TN issues a macro instruction to perform a parameter check on a task that has a target TN, Y, ?, and N indicate the following:
Y: Processing is performed normally.
?: Processing is performed, but the instruction must not be issued.
N: A parameter error is detected.
When the target TN is `0`, the CPMS does not regard it as a parameter error. No processing is performed, and a return code of `1` is returned.

Figure 2-2 Relationships Among TNs During Parameter Checks

1.5 CPMS Macros

(1) Task management
- rleas
- queue
- exit
- abort
- wait
- post
- susp
- rsum
- asusp
- arsum
- chap
- sfact
- gfact

(2) Memory management
- wrtmem
- chkbmem
- chktaer
- MRAMmemcpy

(3) Timer management
- timer
- ctime
- delay
- stime
- gtime

(4) Shared resource management
- rserv
- prsrv
- free
- pfree

(5) System management
- wdtset

(6) System services
- getsysinfo
- gettaskinfo
- gtkmem
- usrdhp
- usrel
- save_env
- resume_env
- gettimebase
- TimebaseToSecs
- atmswap
- atmand
- atmor
- atmxor
- atmadd
- atmtas
- atmcas
- prog_start
- prog_switch
- prog_exit
- prog_call

Notes on coding

The CPMS provides the following include files:

- cpms_types.h: This defines variable types used by macros.
- cpms_macro.h: This defines macro functions.
- cpms_errno.h: This defines macro return codes.
- cpms_table.h: This defines table structures in the CPMS.
- cpms_dhp.h: This defines the codes used in the DHP.
- cpms_elog.h: This defines the codes and structures used in the error log.
- cpms_ulsub.h: This defines the codes and structures used by built-in subroutines.

To use macros for system management or TimebaseToSecs, add the `-lsysctl` option to your command when performing loading (`svload`).

Name

rleas - Make a task wait to be started (put a task in the wait state).

C format

```
int rleas(&tn)
long tn;
```

Description

The `rleas` macro checks whether the task specified by the `tn` parameter is in the DORMANT state, and if so, puts the task in the IDLE state. If the target task is not in the DORMANT state, the macro has no effect.
Specify the task number of the target task in the `tn` parameter.

When a target task is aborted during execution of I/O processing, the task enters the DORMANT state, but is not actually aborted until I/O processing has finished. If the `rleas` macro is issued for a task in the DORMANT state that is currently executing I/O processing, `rleas` quickly ends normally with a return code of `0`. The target task enters the IDLE state after it finishes I/O processing and abort processing.

Diagnostics

When the macro ends processing normally, it returns a return code of `0`. In other cases, the following return codes are returned:
`1`: `tn` is `0`.
`3`: The task specified by `tn` is not in the DORMANT state.
`4`: The task specified by `tn` has not been registered.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.
• The task number specified by `tn` is within the range of `0` to the maximum task number.

Name

queue - Start a task.

C format

```
int queue(&tn, &fact)
long tn, fact;
```

Description

The `queue` macro makes the task specified by the `tn` parameter wait to be executed by the CPU when the task is in the IDLE state. Making a task wait to be executed by the CPU is called *starting the task*.

Tasks waiting to be executed by the CPU are dispatched in order of priority. If a specified task has a higher priority than the task that issued the `queue` macro, the specified task is dispatched. If a specified task has a lower priority than the task that issued the `queue` macro, the macro that issued the `queue` macro is executed.

Specify the task number in the `tn` parameter.

When a task is made to wait to be executed by the CPU, the value set in the `fact` parameter is set as the initiation factor. Up to 32 initiation factors can be set. Note that the same factor cannot be set more than once. When an initiation factor is loaded by the `gfact` macro, the factor is eliminated.

If the value of the specified initiation factor does not fall in the range from 1 to 32, the task is processed assuming that no initiation factor has been specified.

If another `queue` macro is issued when the specified task is waiting to be executed by the CPU, after the specified task ends, the task waits again to be executed by the CPU. Up to two task executions can be stored. That is, a task waiting to be executed by the CPU can have one more execution specified, and a task that is not waiting to be executed by the CPU can be queued for execution up to two times. Note that if the execution of a task is aborted, any second queued execution is not performed either.

Diagnostics

When processing ends normally, a return code of `0` is returned. When processing does not end normally, one of the following return codes is returned:

`1`: `tn` is `0`.

`2`: The task specified by `tn` is in the DORMANT state.

`4`: The task specified by `tn` has not been registered.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.

• The task number specified by `tn` is within the range of `0` to the maximum task number.

Name

exit - End a task.

C format

```
exit()
```

Description

The `exit` macro ends the execution of the task that issued it. That is, the task is not made to wait to be executed by the CPU, but is placed in the IDLE state.
If the task priority level was changed by the `chap` macro during execution of the task, the priority level is restored to its value from when the task was registered.
All shared resources that were locked by using the `rserv` macro are freed.
Task end monitoring is also canceled.
If the built-in subroutine EXS is registered, a link is made to EXS.
When a start request is queued, the task is made to wait to be executed by the CPU again after the task completes its end processing.

Diagnostics

Issuing an `exit` macro does not return control to the task. Therefore, there is no return code.

Name

abort - Forcibly end a task.

C format

```
int abort(&tn)
long tn;
```

Description

The `abort` macro forcibly terminates the task specified by the `tn` parameter, and places the task in the DORMANT state. When execution of the task is suppressed or the task is waiting for an event, the `abort` macro cancels those states and places the task in the DORMANT state.
If the task priority level was changed by using the `chap` macro during execution of the task, the priority level is restored to its value from when the task was registered.
All shared resources that were locked by using the `rserv` macro are freed.
Initiation factors set for the task are cleared.
Task end monitoring is also canceled.
If the built-in subroutine ABS is registered, a link is made to ABS.
Timer events registered by the specified task by using the `timer` macro are not canceled. In addition, timer-event-based startup for the task is not canceled unless the task is in the DORMANT state, in which case startup is not processed normally.

Diagnostics

When processing ends normally, a return code of `0` is returned. When processing does not end normally, one of the following return codes is returned:
`1`: `tn` is `0`.
`2`: The task specified by `tn` is in the DORMANT state.
`4`: The task specified by `tn` has not been registered.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.
• The task number specified by `tn` is within the range of `0` to the maximum task number.

Name

wait - Suppress task execution until an event is generated.

C format

```
int wait(&ecb_g)
long ecb_g;
```

Description

The `wait` macro makes the task that issued it wait for an event to be generated by a `post` macro. The event to be awaited is specified in the `ecb_g` parameter. In the `ecb_g` parameter, set the address of the pointer to the event control block (ECB) allocated in the GLB.
If the value of the `ecb_g` parameter specified by using the `post` macro is the same as the value of the `ecb_g` parameter specified by using the `wait` macro, the task that issued the `wait` macro has its event wait state canceled and is made to wait to be executed by the CPU.
If the event to be awaited has already been generated by using a `post` macro, the task that issued the `wait` macro does not enter the event wait state.
When the event is generated by using the `post` macro, the POST code specified in the `post` parameter `pcode` is set in the ECB. The POST code is set in bits 29 to 0 (0 is the LSB) in the ECB.
The POST code is reported as a return code of the `wait` macro. For instance, suppose that the task that issued the `post` macro sets the factor that generated the event as the POST code. Then, after the task that issued the `wait` macro has its event wait state canceled, the task can learn that factor from the POST code.
Define an ECB for each event.

Diagnostics

When processing ends normally, the POST code is returned.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.

• The ECB specified by `ecb_g` is already being used by the `wait` macro of another task.

NOTICE
Make sure that you initialize the ECB allocated in the GLB to `0` before use.

Name

post - Generate an event and restart the task.

C format

```
int post(&ecb_g, &pcode)
long ecb_g;
long pcode;
```

Description

The `post` macro cancels the event wait state of a task that issued a `wait` macro and is waiting for an event. Then, the `post` macro passes the POST code specified by the `pcode` parameter. The event to be generated is specified in the `ecb_g` parameter. In the `ecb_g` parameter, set the address of the pointer to the event control block (ECB) allocated in the GLB. The POST code is set in bits 29 to 0 (0 is the LSB) in the ECB.

If a task whose event wait state is canceled by using the `post` macro has a higher priority level than the task that issued the `post` macro, control is passed to the task whose event wait state was canceled.

If the value of the `ecb_g` parameter specified in the `post` macro is the same as that of the `ecb_g` parameter specified in the `wait` macro, the task that issued the `wait` macro has its event wait state canceled and is made to wait to be executed by the CPU.

If there are no tasks that have issued a `wait` macro and are waiting for an event to be generated, the POST code specified in `pcode` is set in bits 29 to 0 (0 is the LSB) in the ECB, and the event is registered as having already been generated. The POST code specified here is passed later, when a `wait` macro is issued.

Diagnostics

When no tasks are waiting for the event specified by `ecb_g`, processing ends normally and a return code of `3` is returned. If processing ends normally under other conditions, a return code of `0` is returned. In other cases, the following return code is returned:

`2`: The task waiting for the specified event is in the DORMANT state.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.

• The POST code specified in `pcode` is set in bits 29 to 0 (0 is the LSB) in the ECB.

Name

susp - Suppress task execution.

C format

```
int susp(&tn)
long tn;
```

Description

The `susp` macro suppresses execution of the task specified by the `tn` parameter. The specified task must be either waiting to be executed by the CPU, or in the IDLE state.
The task whose execution was suppressed cannot be released from this state until either an `rsum` macro is issued, or the task is forcibly ended by using the `abort` macro.
If two or more `susp` macros are issued for the same task, only one macro takes effect. Therefore, one `rsum` macro is enough to cancel suppression of task execution.

Diagnostics

When processing ends normally, a return code of `0` is returned. If not, the following return codes are returned:
`1`: `tn` is `0`.
`2`: The task specified by `tn` is in the DORMANT state.
`3`: The task specified by `tn` is already in the SUSPENDED state.
`4`: The task specified by `tn` has not been registered.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.
• The task number specified by `tn` is within the range of `0` to the maximum task number.

Name

rsum - Suppress the execution of a task.

C format

```
int rsum(&tn)
long tn;
```

Description

The `rsum` macro cancels suppression of execution of the task specified by the `tn` parameter, if the task's execution has been suppressed by using the `susp` macro.
If the task whose execution is released from suppression has a higher priority than the task that issued the `rsum` macro, control is passed to the higher-priority task.
The `rsum` macro has no effect in a certain case. If a task's execution is suppressed by using the `asusp` macro, issuing the `rsum` macro cannot cancel suppression of execution of this task.

Diagnostics

When processing ends normally, a return code of `0` is returned. When processing does not end normally, one of the following return codes is returned:
`1`: `tn` is `0`.
`2`: The task specified by `tn` is in the DORMANT state.
`3`: The task specified by `tn` has not had its execution suppressed by using the `susp` macro.
`4`: The task specified by `tn` has not been registered.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.
• The task number specified by `tn` is within the range of `0` to the maximum task number.

Name

asusp - Suppress execution of multiple tasks.

C format

```
int asusp()
```

Description

The asusp macro suppresses the execution of all tasks other than the task that issued the asusp macro. The CPMS has an execution suppression counter that records the number of times the asusp macro has been issued. This counter is incremented by one when the asusp macro is issued, and decremented by one when the arsum macro is issued. If the value of the execution suppression counter is 0, it is not decremented. If the value of the execution suppression counter is greater than 0, all tasks other than the task that issued the asusp macro have their execution suppressed. The asusp macro can be issued by only one task at a time. When the task that issued the asusp macro issues the wait macro or exit macro, the execution suppression counter is set to 0. The counter is also set to 0 when the task that issued the asusp macro is aborted.

The rsum macro cannot be used to cancel suppression of execution that was effected by using the asusp macro. Meanwhile, even if the susp macro is issued for a task whose execution was suppressed by using the asusp macro, the execution suppression counter does not change, although the fact that execution was suppressed by using the susp macro is recorded. To cancel suppression of execution of such tasks, you need to use the arsum macro to clear the execution suppression counter to 0, and use the rsum macro to cancel suppression of execution effected by the susp macro.

Diagnostics

After the asusp macro is processed, the value of the execution suppression counter is returned as the return code.

NOTICE
The asusp macro locks the CPU, but this does not guarantee the locking of other resources. That is, if a conflict occurs with another task when locking resources by using the rserv macro, a deadlock occurs. After the asusp macro is issued, do not issue any process or macro that could cause a conflict for resources. If you do not minimize the time for which the asusp macro is active, system operation might be adversely affected. Do not issue any other macros after issuing the asusp macro until the arsum macro is issued.

Name

arsum - Cancel suppression of execution of multiple tasks.

C format

```
int arsum()
```

Description

The `arsum` macro cancels suppression of execution effected by the `asusp` macro. The CPMS has an execution suppression counter that records how many times the `asusp` macro has been issued. This counter is incremented by one when the `asusp` macro is issued, and decremented by one when the `arsum` macro is issued. If the value of the execution suppression counter is `0`, it is not decremented. If the value of the execution suppression counter is greater than 0, all tasks other than the task that issued the `asusp` macro have their execution suppressed.

If the task that issued the `asusp` macro issues a `wait` macro or `exit` macro, the execution suppression counter is set to `0`. The counter is also set to `0` when the task that issued the `asusp` macro is aborted.

The `rsum` macro cannot be used to cancel suppression of execution that was effected by using the `asusp` macro. Meanwhile, even if the `arsum` macro is issued for a task whose execution was suppressed by using the `susp` macro, suppression of execution effected by the `susp` macro cannot be canceled.

If suppression of execution of a higher-priority task than the task that issued the `arsum` macro is canceled, control is passed to that higher-priority task.

Diagnostics

After the `arsum` macro is processed, the value of the execution suppression counter is returned as the return code.

`0`: Suppression of execution effected by the `asusp` macro has already been canceled.

n: To cancel suppression of execution effected by the `asusp` macro, the `arsum` macro must be executed *n* more times.

Name

chap - Temporarily change a task priority level.

C format

```
int chap(&tn, &chgp)
long tn, chgp;
```

Description

The chap macro temporarily changes the priority of the task specified by the tn parameter to the priority level specified by the chgp parameter. After the chap macro is processed, if the priority level of the specified task is made higher than that of the task that issued the chap macro, and if the specified task is waiting to be executed by the CPU, control is passed to the specified task. If the specified task is the task that issued the chap macro, lowering the priority level passes control to the task with the higher priority level.

A priority level that has been changed temporarily by using the chap macro remains in effect until the specified task ends or is aborted.

The chap macro can also be used to increase the priority of a task waiting for a resource to be freed. However, this does not mean that the resource is forcibly allocated to the task.

Diagnostics

When processing ends normally, a return code of 0 is returned. When processing does not end normally, one of the following return codes is returned:

1: tn is 0.
2: The task specified by tn is in the DORMANT state.
4: The task specified by tn has not been registered.

Parameter check

A parameter check is performed to see whether the following conditions are met. If the conditions are not met, a parameter check error is reported.

- The task number specified by tn is within the range of 0 to the maximum task number.
- If the target task is a system task, the priority level set by using the chgp macro is within the range of 0 to 31.
- If the target task is a user task, the priority level set by using the chgp macro is within the range of 4 to 27.

Name

sfact - Set a task initiation factor.

C format

```
int sfact(&tn, &fact)
long tn, fact;
```

Description

The `sfact` macro sets the initiation factor specified by the `fact` parameter for the task specified by the `tn` parameter. If the initiation factor does not fall in the range from 1 to 32, it is assumed that no initiation factor is specified.
The specified initiation factor is loaded by using the `gfact` macro and then cleared. Even if the same initiation factor is set again for the same task, no duplicate records are made. Therefore, that initiation factor can be cleared by issuing the `gfact` macro once.
When the specified task is unregistered or in the DORMANT state, no initiation factor is set. If the specified task is aborted, the initiation factors are all cleared.

Diagnostics

When processing ends normally, a return code of `0` is returned. When processing does not end normally, one of the following return codes is returned:
`1`: `tn` is `0`.
`2`: The task specified by `tn` is in the DORMANT state.
`4`: The task specified by `tn` has not been registered.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.
• The task number specified by `tn` is within the range of `0` to the maximum task number.

Name

gfact - Load a task initiation factor.

C format

```
int gfact(fact)
long *fact;
```

Description

The `gfact` macro loads an initiation factor set in the task that issued the `gfact` macro, into the address set in the `fact` parameter. Only one initiation factor, that with the smallest numerical value, is loaded.

The loaded initiation factor is cleared. Further unloaded initiation factors can be loaded by issuing the `gfact` macro again. When all initiation factors have been loaded, the `gfact` macro returns `0` to `fact`.

When a task is started, use the `gfact` macro to load all initiation factors.

Diagnostics

When processing ends normally, a return code of `0` is returned.

Name

wrtmem - Write to protected memory.

C format

```
int wrtmem(vaddr, dst, size)
long *vaddr;
long *dst;
int size;
```

Description

The `wrtmem` macro writes to write-protected memory. This macro is used by programming tasks to load programs and data.

Parameters

`vaddr`: Starting address of the transfer source (Specify an address on a four-byte boundary.)
`dst`: Starting address of the transfer destination (Specify an address on a four-byte boundary.)
`size`: Number of data items (Specify data in bytes. The number of data items must be a multiple of four less than or equal to 8,192.)

Diagnostics

When processing ends normally, a return code of `0` is returned. When processing does not end normally, the following return code is returned:

`1`: Parameter error (One of the `vaddr`, `data`, and `size` parameters does not meet the conditions.)

NOTICE
Depending on the specified address, the `wrtmem` macro might destroy the program. The CPMS cannot prevent the macro from destroying the program.

Name

chkbmem - Check access to the bus memory.

C format

```
int chkbmem(slot)
long slot;
```

Description

The `chkbmem` macro returns a value indicating whether bus memory in the specified slot is accessible.

Parameters

`slot`: Number of the slot where bus memory is mounted (0 to 7)

Diagnostics

If the bus memory is working normally (is accessible), a return code of `0` is returned. If the bus memory is not working normally, one of the following return codes is returned:
`0x8000`: Parameter error
`0x8001`: Not mounted
`0x8002`: CARD OFF state
`0x8004`: Target abort detected (failure)

Name

chktaer - Check for target abort errors.

C format

```
int chktaer(slot)
long slot;
```

Description

The `chktaer` macro returns a value that indicates whether there is a target abort in the bus memory in the specified slot.

Parameters

`slot`: Bus memory slot number (0 to 7)

Diagnostics

When processing ends, one of the following return codes is returned:
`0`: There was no target abort.
`1`: There was a target abort.
`2`: There was a parameter error.

Name

MRAMmemcpy - Copy memory dedicated to MRAM.

C format

```
void *MRAMmemcpy(dest, src, count)
void *dest;
void *src;
unsigned long count;
```

Description

The `MRAMmemcpy` macro copies memory to MRAM (memory dedicated to MRAM).
The specified byte size of memory area is copied from the copy source to the copy destination.

Parameters

`dest`: MRAM address of the copy destination (Specify an address that is at a long word boundary.)
`src`: MRAM address of the copy source (Specify an address that is at a long word boundary.)
`count`: Transfer byte count (Specify a multiple of 4.)

Diagnostics

When processing ends, the transfer destination address specified by `dest` is returned.

NOTICE
Accessing MRAM repeatedly might cause the CPU to stop working properly. When copying MRAM, make sure you use `MRAMmemcpy()` instead of the standard `memcpy()`.

Name

timer - Register a task that starts up based on a timer event.

C format

```
int timer(&id, &tn, &fact, &t, &cyt)
long id, tn, fact, t, cyt;
```

Description

The `timer` macro registers the task specified by the `tn` parameter to be started up based on a timer event.
The type of timer event is specified by the `id` parameter. Four timer events can be specified: duration-based, time-based, duration-cycle-based, and time-cycle based. These events are explained in the table on the following page.
When a time-based timer event is to be registered, if the specified time has already elapsed, the same time on the following day is registered instead. If the time has been advanced by using the `stime` macro, causing a time-based timer event to be skipped, the event is registered for the same time on the following day.
The initiation factor specified by the `fact` parameter is passed to the task subject to timer-event-based startup.
If the specified initiation factor is not in the range from 1 to 32, it is assumed that no initiation factor is specified.
When a timer event is to be registered, the state of the specified task is not checked. If the specified task is in the DORMANT state when the timer event occurs, the task is not started.
To cancel a timer event, use the `ctime` macro. Aborting or deleting a task does not cancel any timer events to which it is registered.

The following gives the meanings of the parameters:
`id`: Type of timer event (Specify a number from 1 to 4.)
`tn`: Task number of the task to be registered to start based on a timer event
`fact`: Initiation factor to be passed to the task to be started
`t`: Time of the first timer event or relative time based on the current time (both in milliseconds)
`cyt`: Duration of the cycle when events are to be generated cyclically (in milliseconds)
When the value of `id` is `1` or `2`, specify `0`. When the value of `id` is `3` or `4`, specify a value from `0` to `86400000`.

Descriptions of the id, t, and cyt parameters in the timer macro

Timer event	id	t	cyt	Description
Duration-based	1	Relative time until the start time measured from the current time	Specify 0.	After the duration specified by the t parameter, the task specified by the tn parameter is started.
Time-based	2	Start time measured from 00:00 (12 a.m.)	Specify 0.	The task specified by the tn parameter is started at the time specified by the t parameter.
Duration-cycle-based	3	Relative time until the start time, measured from the current time (relative time until the task is first started)	Interval at which to periodically start the task after the first time it is started	After the duration specified by the t parameter, the task specified by the tn parameter is started. Then, the task specified by tn is started periodically at the interval specified by the cyt parameter.
Time-cycle-based	4	Time when the task is started, measured from 00:00 (12 a.m.) (time when the task is first started)	Interval at which to periodically start the task after the first time it is started	The task specified by the tn parameter is started at the time specified by the t parameter. Then, the task specified by the tn parameter is started periodically at the interval specified by the cyt parameter.

Diagnostics

When processing ends normally, a return code of 0 is returned. When processing does not end normally, one of the following return codes is returned:

1: tn is 0.

4: No more timer events can be registered because the system tables are full.

Parameter check

A parameter check is performed to see whether the following conditions are met. If any conditions are not met, a parameter check error is reported.

- The task number specified by tn is within the range of 0 to the maximum task number.
- The type specified by id is within the range of 1 to 4.
- When the value of id is 1 or 3, the time specified by t is within the range of 0 to 86400000.
- When the value of id is 2 or 4, the time specified by t is within the range of 0 to 86400000.
- When the value of id is 1 or 2, the value of cyt is 0.
- When the value of id is 3 or 4, the interval specified by cyt is within the range of 0 to 86400000.

Name

ctime - Cancel starting of a task based on a timer event.

C format

```
int ctime(&tn, &fact)
long tn, fact;
```

Description

The `ctime` macro cancels timer events registered by using the `timer` macro.
The `ctime` macro searches for and cancels timer events whose task number specified by the `tn` parameter matches the initiation factor specified by the `fact` parameter. When the initiation factor is not within the range of `1` to `32`, it is assumed that no initiation factor is specified.
The `ctime` macro cannot cancel execution of a task that has already been started. However, if any tasks that have already been started have any cyclic events registered at or after the current time, those timer events are canceled.

Diagnostics

When processing ends normally, a return code of `0` is returned. When processing does not end normally, the following return code is returned:
`1`: Timer events that match the specified task number and initiation factor are not registered.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.
• The task number specified by `tn` is within the range of `0` to the maximum task number.

Name

delay - Suppress task execution only for a specified duration.

C format

```
int delay(&t)
long t;
```

Description

The `delay` macro suppresses execution of the task that issued the macro for the duration specified by the `t` parameter.
For the `t` parameter, specify in milliseconds the duration for which task execution is to be suppressed. While task execution is suppressed, control is transferred to other tasks. After execution has been suppressed for the specified duration, if there are no other operable tasks (higher-priority tasks or same-priority tasks that started before the task that issued the `delay` macro), control returns to the task that issued the `delay` macro.

Diagnostics

When processing ends normally, a return code of `0` is returned. When processing does not end normally, one of the following return codes is returned without suppressing execution of the task:
`4`: Task execution could not be suppressed because the system tables are full.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.

- The duration for which to suppress execution specified by `t` is within the range of 0 to 86400000 (24 hours).

NOTICE
Do not issue the delay macro while shared resources are locked, or system operation might be adversely affected.

Name

stime - Set the time.

C format

```
int stime(&time)
struct{
    short year;
    short month;
    short day;
    short dummy;
    long msec;
}time;
```

Description

The `stime` macro changes the time managed by the CPMS and the TOD to the time specified by the `time` parameter.
Specify the `time` parameter according to the following rules:
`year`: Set a calendar year from 1970 to 2069.
`month`: Set the month.
`day`: Set the date.
`msec`: Set the time in milliseconds measured from 00:00 (12 a.m.).
Set a value within the range of `0` to `86399999` (23:59:59:999).

Diagnostics

When processing ends normally, a return code of `0` is returned. When processing does not end normally, the following return code is returned:
`1`: The time specified by the `time` parameter is invalid.

Parameter check

A parameter check is performed to see whether the following condition is met. If the condition is not met, a parameter check error is reported.

- The year is from 1970 to 2069, the month is from 1 to 12, the date is from 1 to 31, and the time in milliseconds is from 0 to 86399999.

NOTICE

The time at which to generate timer events registered by using the `timer` macro is sometimes changed. The following table shows when this occurs.

Timer event type	When time is delayed	When time is advanced	Remarks
Duration-based and duration-cycle-based	The time of the timer event is not affected.	The time of the timer event is not affected.	When the duration specified by the `timer` macro has elapsed, the timer event is generated.
Time-based and time-cycle-based	When the time at which to generate the timer event is delayed by 24 hours or more, the time is registered as the same time on the following day.	If the first scheduled start time is skipped, making the task unable to start, the task starts when the time is changed. The scheduled start time is changed to the time the change was made or later, by adding the cycle time to the first scheduled start time.	

Name

gtime - Load the current time.

C format

```
int gtime(&time)
struct{
   short year;
   short month;
   short day;
   short dummy;
   long msec;
}time;
```

Description

The `gtime` macro stores the current time at the address specified by the `time` parameter.
The time is stored in the following parameters:
`year`: Calendar year from 1970 to 2069
`month`: Month
`day`: Date
`msec`: Time in milliseconds measured from 00:00

Diagnostics

When processing ends normally, a return code of `0` is returned.

Name

rserv - Lock shared resources in a batch.

C format

```
#include <cpms/cpms_rserv.h>

int rserv(&n, &para1, &para2,...)
long n;
cpms_rserv_t para1, para2,...;
```

Description

The `rserv` macro locks the shared resources specified by parameters `para1`, `para2`, and so on. The `rserv` macro returns an error if the task that issued the macro has already locked the shared resources by using another `rserv` macro. Locking shared resources in one batch at a time prevents deadlocks.

If the specified task has not yet locked shared resources by using the `rserv` macro, the issued macro checks whether the shared resources specified by parameters `para1`, `para2`, and so on are locked by other tasks. If all specified shared resources are free, the specified task locks them. If any of the specified resources are locked by another task, the `rserv` macro does not return a value, and execution of the specified task is suppressed.

A task whose execution was suppressed because it could not lock shared resources waits for other tasks to free the specified shared resources by using the `free` macro. When this happens, the task whose execution was suppressed locks the shared resources, and the `rserv` macro returns a value.

Shared resources are freed either when the `free` macro is issued, or when the task locking them ends or aborts.

Shared resources locked by using the `rserv` macro cannot be freed by using the `pfree` macro.

When the system tables are full, the task that issued the `rserv` macro has its execution suppressed even if the shared resources are not locked by another task. In this case, the system tables will have available space when another task frees shared resources by using the `free` macro. When this happens, the task that issued the `rserv` macro locks the shared resources, and the `rserv` macro returns a value. The number of system tables managing shared resources is defined in the CPMS. Make sure that you specify no more shared resources than the number of defined system tables.

Do not issue the `rserv` macro after issuing the `sups` or `asusp` macro that suppresses execution of another task. If a task whose execution is suppressed has locked a shared resources, a deadlock occurs.

A shared resource is represented as an area in SAREA of the GLB.

Specify the `n` parameter and the `cpms_rserv_t` structure as shown in the following:

`n`: Number of locked shared resources (1 to 32)

```
typedef struct cpms_rserv{
        long type;
        long addr;
        long top;
        long last;
        }cpms_rserv_t;
```

`type`: This parameter has no meaning. It is always set to `0`.

`addr`: This is the address of SAREA containing the shared resource to be locked

`top`: This is the starting address of the shared resource to be locked. This is a relative address based on the beginning of SAREA.

`last`: This is the ending address of the shared resource to be locked. This is a relative address based on the start of SAREA.

Diagnostics

If all shared resources can be locked, a return code of `0` is returned. If not, the following return code is returned:

`2`: The shared resource has already been locked by using the `rserv` or `prsrv` macro.

Parameter check

A parameter check is performed to see whether the following conditions are met. If any conditions are not met, a parameter check error is reported.

- The value of `addr` is valid.
- The value of `top` or `last` is valid.
- The number of shared resources specified by `n` is within the range of 1 to 32.

NOTICE
When resources are locked by using the `prsrv` macro, no resource can be locked by using the `rserv` macro.

Name

prsrv - Lock shared resources.

C format

```
#include <cpms/cpms_rserv.h>

int prsrv(&n, &para1, &para2,...)
long n;
cpms_rserv_t para1, pata2,...;
```

Description

The prsrv macro locks the shared resources specified by parameters para1, para2, and so on. When the task that issued the prsrv macro has already locked shared resources by using the prsrv macro, the prsrv macro can be used to lock other resources.
The prsrv macro checks whether the shared resources specified by parameters para1, para2, and so on have been locked by other tasks. If all specified shared resources are free, the specified task locks them. If some of the specified shared resources are locked by another task, a value is not returned from the prsrv macro, and execution of the specified task is suppressed.
When a specified shared resource has been locked by the prsrv macro issued by the local task, processing is performed assuming that the shared resource is now locked. Note that the same resource might be locked more than once by the same task by using the prsrv macro. To free such shared resources, issue the pfree macro as many times as the prsrv macro was issued.
A task whose execution was suppressed because it could not lock shared resources waits for other tasks to free the specified resources by using the pfree macro. When this happens, the task whose execution was suppressed locks the shared resources, and the prsrv macro returns a value.
Shared resources are freed either when the pfree macro is issued, or when the task locking them ends or aborts.
Shared resources locked by using the prsrv macro cannot be freed by using the free macro.
When the system tables are full, the task that issued the prsrv macro has its execution suppressed, even if the shared resources are not locked by another task. In this case, the system tables will have available space when another task frees shared resources by using the pfree macro. When this happens, the task that issued the prsrv macro locks the shared resources, and the prsrv macro returns a value. The number of system tables managing shared resources is defined in the CPMS. Make sure that you specify no more shared resources than the number of system tables.
A shared resource is represented as an area in SAREA of the GLB.
Specify the n parameter and the cpms_rserv_t structure as shown in the following:
n: Number of shared resources to be locked (1 to 5)

```
typedef struct cpms_rserv{
        long type;
        long addr;
        long top;
        long last;
}cpms_rserv_t;
```

type: This parameter has no meaning. It is always set to 0.
addr: This is the address of SAREA containing the shared resource to be locked
top: This is the starting address of the shared resource to be locked. This is a relative address based on the beginning of SAREA.
last: This is the ending address of the shared resource to be locked. This is a relative address based on the start of SAREA.

Diagnostics

When all shared resources are locked, a return code of `0` is returned. If not, the following return code is returned:

`2`: The number of shared resources that can be locked by a single task was exceeded.

Parameter check

A parameter check is performed to see whether the following conditions are met. If any conditions are not met, a parameter check error is reported.

- The value of `addr` is valid.
- The value of `top` or `last` is valid.
- The number of shared resources specified by `n` is within the range of 1 to 5.

Name

free - Free the shared resources locked by using the rserv macro.

C format

```
#include <cpms/cpms_rserv.h>

int free(&n, &para1, &para2,...)
long n;
cpms_rserv_t para1, para2,...;
```

Description

The free macro frees shared resources that are locked by using the rserv macro. Among the shared resources in parameters para1, para2, and so on, the free macro frees any shared resources that are locked.
When shared resources are freed, any tasks waiting for those shared resources to be freed no longer have their execution suppressed.
When the specified shared resources contain an unlocked shared resource, a return code of 1 is returned. Even in this case, the locked shared resources are freed.
Shared resources locked by using the prsrv macro cannot be freed by using the free macro.
A shared resource is represented as an area in SAREA of the GLB.
Specify the n parameter and the cpms_rserv_t structure as shown in the following:
n: Number of shared resources to be freed (1 to 32)

```
typedef struct cpms_rserv{
        long type;
        long addr;
        long top;
        long last;
}cpms_rserv_t;
```

type: This parameter has no meaning. It is always set to 0.
addr: This is the address of SAREA containing the shared resource to be freed.
top: This is the starting address of the shared resource to be freed. This is a relative address based on the beginning of SAREA.
last: This is the ending address of the shared resource to be freed. This is a relative address based on the start of SAREA.

Diagnostics

When the shared resources are freed, a return code of 0 or 1 is returned. If not, the following return code is returned:
2: None of the specified shared resources were locked.

Parameter check

A parameter check is performed to see whether the following conditions are met. If any conditions are not met, a parameter check error is reported.

- The value of `addr` is valid.
- The value of `top` or `last` is valid.
- The number of shared resources specified by `n` is within the range of 1 to 32.

Name

pfree - Free shared resources locked by using the prsrv macro.

C format

```
#include <cpms/cpms_rserv.h>

int pfree(&n, &para1, &para2,...)
long n;
cpms_rserv_t para1, para2,...;
```

Description

The pfree macro frees shared resources that are locked by using the prsrv macro. Among the shared resources in parameters para1, para2, and so on, the pfree macro frees any shared resources that are locked.
When shared resources are freed, any tasks waiting for those shared resources to be freed no longer have their execution suppressed.
When the specified shared resources contain an unlocked shared resource, a return code of 1 is returned. Even in this case, the locked shared resources are freed.
Shared resources locked by using the rsrv macro cannot be freed by using the pfree macro.
A shared resource is represented as an area in SAREA of the GLB.
Specify the n parameter and the cpms_rserv_t structure as shown in the following:
n: Number of shared resources to be freed (1 to 5)

```
typedef struct cpms_rserv{
        long type;
        long addr;
        long top;
        long last;
}cpms_rserv_t;
```

type: This parameter has no meaning. It is always set to 0.
addr: This is the address of SAREA containing the shared resource to be freed.
top: This is the starting address of the shared resource to be freed. This is a relative address based on the beginning of SAREA.
last: This is the ending address of the shared resource to be freed. This is a relative address based on the start of SAREA.

Diagnostics

When the shared resources are freed, a return code of 0 or 1 is returned. If not, the following return code is returned:
2: None of the specified shared resources were locked.

Parameter check

A parameter check is performed to see whether the following conditions are met. If any conditions are not met, a parameter check error is reported.

- The value of `addr` is valid.
- The value of `top` or `last` is valid.
- The number of shared resources specified by `n` is within the range of 1 to 5.

Name

wdtset - Control the starting or stopping of the WDT.

C format

```
int wdtset(msec)
long *msec;
```

Description

The `wdtset` macro controls the starting and stopping of the WDT (watchdog timer).
When a WDT timeout occurs, a link is made to the WDTES built-in subroutine.
WDTES must be used to process WDT timeouts.
If this macro is used on the CP side, an interrupt enters only the CP in the event of a timeout, and a link is made to the WDTES built-in subroutine on the CP side.

Parameters

`msec`: WDT duration (0 to 65535) (in milliseconds)
If `msec` is set to a value from 1 to 65535, the WDT starts.
If `msec` is set to `0`, the WDT is stopped.

Diagnostics

`0`: The processing ends normally.
`1`: A parameter error occurs.

Name

getsysinfo - Get the system state.

C format

```
int getsysinfo(type, addr)
int type;
char *addr;
```

Description

The `getsysinfo` macro returns the system information specified by `type` to the address specified by `addr`.
For `type`, specify any one of the following:

- SYS_IDLE
  This returns the cumulative IDLE time.

```
struct sys_idle {
    unsigned int idle_sec;     /* In seconds */
    int idle_nsec;  /* In nanoseconds */
};
```

The IDLE time is counted from `0` as the CPMS start time. Therefore, calculate the IDLE time from the difference between the IDLE time at the last time `SYS_IDLE` was issued and the current IDLE time.
Do not change the system time when measuring the difference between IDLE times.

- SYS_CPMS
  The CPMS version number is returned.

```
int cpms_ver;
```

- SYS_PROC
  A processor number is returned.

```
int proc_no;
```

Diagnostics

If the processing ends normally, the size of the return code information (in bytes) is returned. If the processing does not end normally, one of the following return codes is returned:
`0`: The system information specified by `type` is not the processing target.
`-1`: The macro could not get the system information correctly.

Name

gettaskinfo - Get the task state.

C format

```
int gettaskinfo(type, tn, addr)
int type, tn;
char *addr;
```

Description

The `gettaskinfo` macro returns the information specified by `type` of the task specified by `tn` to the address specified by `addr`. When getting information about the task that issued `gettaskinfo`, set `tn` to `0`.
For `type`, set any one of the following:

- TASK_TN
  The task number of the task that issued the `gettaskinfo` macro is returned. Set `tn` to `0`.
  `int task_tn;`

- TASK_PRI
  The priority level of the task specified by `tn` is returned.
  `int task_pri;`

- TASK_STAT
  The current task state of the task specified by `tn` is returned.
  `int task_stat;`
  `0`: Unregistered, `1`: DORMANT, `2`: IDLE, `3`: READY, `4`: SUSPENDED, `5`: WAIT

Diagnostics

If the processing ends normally, the information size (in bytes) is returned as the return code. If the processing does not end normally, one of the following return codes is returned:
`0`: The task information specified by `type` is not the processing target. Alternatively, the task specified by `tn` is unregistered.
`-1`: The macro could not get the task information correctly.

Name

gtkmem - Read a table managed by the CPMS.

C format

```
int gtkmem(tblno, caseno, offset, size, buf)
int tblno, caseno, offset, size;
char *buf;
```

Description

The `gtkmem` macro reads data from tables managed by the CPMS.
The following shows the meanings of the parameters:
`tblno`: Number specifying the target table

Table:	OSCB	= 1	RSVB = 6
	SYSCB	= 2	UCB = 7
	TCB	= 3	TRB = 8
	TMCB	= 4	
	RSCB	= 5	

`caseno`: Relative case number in the target table
If the target table is OSCB, SYSCB, TMCB, or RSCB, specify `0`.
`offset`: Relative address in the case for the data to be read
`size`: Size of data to be read (in bytes)
`buf`: Address of memory to be read

Diagnostics

When processing ends normally, a return code of `0` is returned. When processing does not end normally, one of the following return codes is returned:
`1`: The table specified by `tblno` is not the processing target.
`2`: The macro could not get the table data correctly.

Name

usrdhp - Write DHP memory.

C format

```
#include <cpms_dhp.h>

int usrdhp(code, data, ndata)
unsigned long code;
long *data;
long ndata;
```

Description

The `usrdhp` macro records a user-defined event in the kernel execution trace data (DHP).

Parameter

`code`: Trace code
Specify any value in the range of DHP_USR0 to DHP_USR7.
`data`: Pointer specifying an array that stores trace data
`ndata`: Array element (0 to 5, where one case is 4 bytes)

Diagnostics

`0`: The processing ended normally.
`1`: A parameter error occurred.

Name

usrel - Write to the user error log.

C format

```
#include <cpms_elog.h>

int usrel(type, class, retcode, errtype, erb)
long type;
long class;
long retcode;
long errtype;
long *erb;
```

Description

After linking to the EAS built-in subroutine, the `usrel` macro writes the error information specified in the argument to the buffer area for error logs in the OS.

Parameters

`type`: This specifies the severity type. Specify one of the following types:

LOG_TYPE_NONFATAL

This type of error does not cause the system to go down. It is specified when some functions are degraded.

Examples include program errors and I/O errors.

LOG_TYPE_WARNING

This is a warning error. This is specified for errors the system can recover from. This includes resource shortage errors, for example those caused by a temporary memory shortage.

LOG_TYPE_NOTE

This is a message for providing information to the user.

`class`: This specifies a class for the error message (subsystem identifier). Specify one of the following classes. The user is responsible for defining the meaning of each class.

LOG_CLASS_MSOFT1 to LOG_CLASS_MSOFT16: For middleware

LOG_CLASS_USER1 to LOG_CLASS_USER16: For applications

`retcode`: This specifies the return value of the function called immediately before the error was detected. If there is no such function, specify `0`.

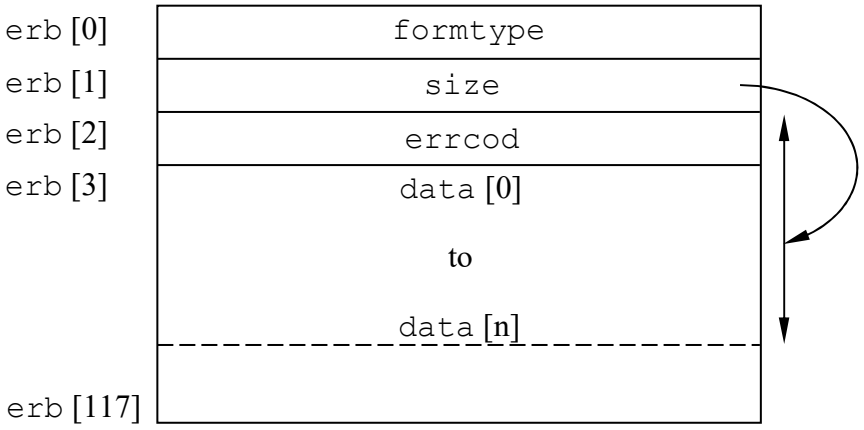
`errtype`: This specifies the type of the possible cause of the failure. Specify one of the following:

LOG_ERRTYPE_HARD: Hardware

LOG_ERRTYPE_SOFT: Software

`erb`: This specifies a pointer to an error block. The error block format is shown on the following page.

● Format of an error block



formtype: This specifies the format type of the error message.
The following are the format type values:
LOG_FORM_MSOFT1 to LOG_FORM_MSOFT16: For middleware
LOG_FORM_USER1 to LOG_FORM_USER16: For applications
LOG_FORM_PIOERR: PI/O error
LOG_FORM_MODULERR: Module error
The user must decide the contents of the values for middleware and for applications.
The PI/O error and module error values are formats specified by the OS.
size: The valid data size for errcode onward is specified in bytes (4 to 464).
errcode: This specifies an error code.
0x08000000 to 0x08FFFFFF: For middleware
0x09000000 to 0x09FFFFFF: For applications
The user is responsible for deciding the meanings of the preceding values.
However, the upper 16 bits must be used for the major number (error type) and the lower 16 bits must be used for the minor number (detailed factor).
In addition to these, you can use the error codes for PI/O errors and module errors specified by the OS.
data: This is detailed error data.
The contents of data must match the format specified in formtype.

Diagnostics

0: The processing ended normally.
1: The processing ended with an error.

Name

save_env - Save the task execution environment.

C format

```
#include <cpms_table.h>
int save_env(env)
struct task_env *env;
```

Description

The save_env macro saves the data of the task execution environment at the time the save_env macro was issued, to the address specified by the env parameter. The saved task execution environment data is used by the resume_env macro.

For the env parameter, set the address of the task_env structure for saving the data of the task execution environment.

The following shows the composition of the task_env structure. This structure requires an area of 240 bytes.

```
struct task_env {
    struct basic_regs      env_basic_regs;
    struct float_regs      env_float_regs;
};
```

Make sure that you allocate an area in the GLB in which to save the data of the task execution environment.

No processing is performed even if the save_env macro is issued in a built-in subroutine.

Diagnostics

If the save_env macro is used to save the data of the task execution environment, a return code of 0 is returned.

If the issuing of the resume_env macro causes a value to be returned from the save_env macro, the value of the val parameter of the resume_env macro is returned as the return code. If val is set to 0, a return code of 1 is returned.

NOTICE
During the return of control when using the resume_env macro to issue the save_env macro, if the BSS or GLB data related to control of the user stack or task differs from that when the save_env macro was issued, issuing the resume_env macro might not result in the same processing as the last time the environment was in operation.

Name

resume_env - Recover a task execution environment.

C format

```
#include <cpms_table.h>
void resume_env(env, val)
struct task_env *env;
int val;
```

Description

The `resume_env` macro recovers the data of the task execution environment specified by the `env` parameter. Because the recovered task execution environment consists only of the register, user stacks and BSS contents are not recovered.

The `resume_env` macro must be issued from the CPES built-in subroutine. If, based on the information output when the CPES built-in subroutine is executed, the system determines not to abort the task or take the CPU down, the `resume_env` macro is enabled. Even if the `resume_env` macro is issued, the task execution environment is not recovered straight away. The environment is recovered after all entries of the CPES built-in subroutine are executed, and control is transferred to the return address of the `save_env` macro corresponding to `env`.

If the task execution environment is recovered normally, a value is returned from the `save_env` macro corresponding to the specified task execution environment. At this time, the `save_env` macro returns the `val` parameter as the return code. If `val` is set to `0`, the `save_env` macro returns `1` as the return code.

If the `resume_env` macro was issued multiple times in entries of the CPES built-in subroutine, the parameter of last `resume_env` macro to be issued is enabled.

Diagnostics

The `resume_env` macro does not return a return code.

NOTICE

- The `resume_env` macro must be issued from the CPES built-in subroutine.
- If the `resume_env` macro is issued from somewhere other than the CPES built-in subroutine, the `resume_env` macro does nothing.
- During the return of control when using the `resume_env` macro to issue the `save_env` macro, if the BSS or GLB data related to control of the user stack or task differs from that when the `save_env` macro was issued, issuing the `resume_env` macro might not result in the same processing as the last time the environment was in operation.
- If the `env` parameter is invalid, the CPU might go down.

Name

gettimebase - Read the time base.

C format

```
void gettimebase(timebase)
unsigned long timebase[2];
```

Description

The `gettimebase` macro gets the 64-bit time base. The time base is incremented every 4 cycles of the bus clock. Because the CPU bus clock speed is 40.0 MHz, the time base is incremented in units of 10.0 MHz. If the time base is divided by 10,000,000, you can get the number of seconds that have elapsed since 00:00:00 of January 1, 1970.
The following parameters are output:
`timebase [0]`: Upper 32 bits of the time base register (TBU)
`timebase [1]`: Lower 32 bits of the time base register (TBL)

NOTICE
The time base is located in the device. In the future, the time base might have to be handled differently due to differences in the device or operating frequency.

Name

TimebaseToSecs - Convert the time base value to seconds and nanoseconds.

C format

```
void TimebaseToSecs(timebase, tval)
unsigned long timebase[2];
struct tval{
   unsigned int tv_sec;
   int tv_nsec;
} tval;
```

Description

The `TimebaseToSecs` macro converts the 64-bit time base value to relative seconds and nanoseconds since 1970.

Name

atmswap, etc. - Atomic operation library

C format

```
long atmswap(addr, data)
long *addr, data;

long atmand(addr, data)
long *addr, data;

long atmor(addr, data)
long *addr, data;

long atmxor(addr, data)
long *addr, data;

long atmadd(addr, data)
long *addr, data;

long atmtas(addr, data)
long *addr, data;

long atmcas(addr, data1, data2)
long *addr, data1, data2;
```

Description

This library guarantees the exclusive right to read, edit, and write data by making it impossible for other tasks or interrupts to overwrite the memory while the memory is being read, edited, or written. This enables exclusive control.
All macros in this library handle only 32-bit integer (long int) data.
The return value `olddata` is the memory value (`addr`) from before the operation.
`->` `addr` means the data is stored in the memory represented by `addr`.

```
olddata = atmswap(addr, data): data -> addr
olddata = atmand(addr, data): (addr) AND data -> addr
olddata = atmor(addr, data): (addr) OR data -> addr
olddata = atmxor(addr, data): (addr) XOR data -> addr
olddata = atmadd(addr, data): (addr) + data -> addr
olddata = atmtas(addr, data): Test And Swap if (addr) = 0, then data -> addr
olddata = atmcas(addr, data1, data2): Compare And Swap if (addr) = data1, then
                                      data2 -> addr
```

NOTICE
This exclusive control is effective only among the processes of the local processor, and is not used for exclusive control among other processors or I/O DMA.

Name

prog_start - Start a subtask.

C format

```
int prog_start(start_addr, initial_usp, main_usp)
viod (*start_addr)();
long *initial_usp;
long *main_usp;
```

Function

The `prog_start` macro starts a subtask. The state of the main program is saved in the stack, and the pointer to that stack is saved in the location represented by `main_usp`. The stack pointer is switched to `initial_usp`, and the subprogram is executed from `start_addr`.

Parameters

`start_addr`: Address of the subprogram before execution (16-byte-aligned)
`initial_usp`: Stack pointer for the subprogram (16-byte-aligned)
`main_usp`: Address where the stack pointer for the main program is saved

Diagnostics

When processing ends, one of the following return codes is returned:
`0`: Returned via the `prog_switch` macro
`1`: Returned via the `prog_exit` macro

Name

prog_switch - Switch among subtasks.

C format

```
int prog_switch(save_usp, resume_usp)
long *save_usp;
long *resume_usp;
```

Description

The `prog_switch` macro switches among subtasks. The state of the subtask is saved in the stack, and the pointer to that stack is saved in the location represented by `save_usp`. The stack pointer is recovered from the location represented by `resume_usp`, the state of the subtask saved in that stack is recovered, and execution is resumed.

Parameters

`save_usp`: Address where the stack pointer for the stopped subtask is saved
`resume_usp`: Address where the stack pointer for the subtask to be resumed is saved

Diagnostics

When processing ends, one of the following return codes is returned:
`0`: Returned via the `prog_switch` macro
`1`: Returned via the `prog_exit` macro

Name

prog_exit - End a subtask.

C format

```
void prog_exit(main_usp)
long *main_usp;
```

Description

The `prog_exit` macro ends a subtask and returns to the main program. The stack pointer is recovered from the location represented by `main_usp`, the saved state of the main program is recovered from that stack, and execution is resumed.

Parameters

`main_usp`: Address where the stack pointer for the main program is saved

Name

prog_call - Switch among stacks and call a subroutine.

C format

```
int prog_call(entry_addr, initial_usp, para_cnt, para1, para2,...)
int (*entry_addr)();
long *initial_usp;
long para_cnt;
long para1, para2,...;
```

Description

The `prog_call` macro switches among stacks and calls subroutines. The stack pointer is switched to `initial_usp`, and the following subroutine is executed:

```
int entry_addr(para1, para2,...);
```

Parameters

`start_addr`: Execution start address of the subroutine (16-byte-aligned)
`initial_usp`: Stack pointer to the subroutine (16-byte-aligned)
`para_cnt`: Number of parameters (0 to 5)
`para1, para2,...`: Parameters (numbering from 0 to 5)

Diagnostics

The return code of the subroutine is returned.

1.6 ET.NET Socket Handler

ET.NET socket handlers are called from user tasks as C functions, and control the ET.NET module.
The functions are provided as library functions.
Therefore, specify `-lsysctl` when loading the program (`svload`).

1.6.1 List of ET.NET socket handlers

Table 2-1 shows a list of ET.NET socket handlers and ET.NET socket handler functions.

Table 2-1 List of ET.NET Socket Handlers

Name	Function	Corresponding program
`tcp_open()`	Actively open TCP	TCP/IP
`tcp_popen()`	Passively open TCP	TCP/IP
`tcp_accept()`	Accept TCP connection request	TCP/IP
`tcp_close()`	Close TCP connection	TCP/IP
`tcp_abort()`	Force abort TCP connection	TCP/IP
`tcp_getaddr()`	Get TCP socket information	TCP/IP
`tcp_stat()`	Get TCP connection status	TCP/IP
`tcp_send()`	TCP data transmission	TCP/IP
`tcp_receive()`	TCP data reception	TCP/IP
`udp_open()`	Open UDP	UDP/IP
`udp_close()`	Close UDP	UDP/IP
`udp_send()`	UDP data transmission	UDP/IP
`udp_receive()`	UDP data reception	UDP/IP
`route_list()`	Get route information	TCP/IP and UDP/IP
`route_del()`	Delete route information	TCP/IP and UDP/IP
`route_add()`	Register route information	TCP/IP and UDP/IP
`arp_list()`	Get ARP information	TCP/IP and UDP/IP
`arp_del()`	Delete ARP information	TCP/IP and UDP/IP
`arp_add()`	Register ARP information	TCP/IP and UDP/IP
`getconfig()`	Get configuration information	TCP/IP and UDP/IP
`set_so_abort()`	Socket close instruction when aborting a task	TCP/IP and UDP/IP

- For both TCP and UDP, you can use up to 96 sockets simultaneously for one module (with two channels).
- Port numbers 0 to 9999 are dedicated to the system, so users must use port number 10000 to 65535.
  The port numbers recommended for users are 10000 to 59999 (TCP ports 60015 and 60016, and UDP ports 60012 to 60015 are dedicated to the system, and cannot be used by users).
- The data length that can be transmitted or received per issue of a function is 1 to 1460 bytes for TCP, and 1 to 1472 bytes for UDP.
- IP addresses and subnet masks are set in the OS table in the CPU. If the CPU is replaced, the settings must be reconfigured.
- Process one task per socket ID (return value for `tcp_open`/`tcp_popen`/`tcp_accept`/`udp_open`). (For example, do not use one socket ID for both a transmitting task and a receiving task.)
- To force abort a task:
  If a task using an ET.NET socket handler is forcibly aborted, the socket it is using remains in a registered state (unless the task has issued `tcp_close()` or `udp_close()` on the socket it is using). Sockets in this state are hereafter referred to as *floating sockets*. Floating sockets cannot be used by other tasks. To free floating sockets, either reset or power cycle the module. Floating sockets can also be recovered by having the task issue `set_so_abort()` in advance. `set_so_abort()` frees sockets on the system if a task is forcibly aborted.

short tcp_open(mj_no, ch_no, &padr)

Function

This function registers TCP/IP program sockets, secures ports, and issues connection requests to remote stations.
This return value is either a registered socket ID or an error code.
This function transmits to SYN, and waits to secure a connection (SYN reception from a remote station).
If there is no response from the remote station, a port freeing error (error code: /F0FF) is returned after 45 seconds. If this happens, reissue the `tcp_open()` function.

Linking procedure

Main module CH1 example

```
struct open_p{
    long  dst_ip;
    short dst_port;
    short src_port;
    char  keepalive;
    char  ttl;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct open_p padr;
    padr.dst_ip    = 0xC0A80001;
    padr.dst_port  = 10000;
    padr.src_port  = 0;
    padr.keepalive = 0;
    padr.ttl       = 0;
    rtn = tcp_open( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identification (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2)
`padr`: Starting address of input parameters (Make sure that you specify an even number as the address.)
`padr` -> `dst_ip`: Remote station IP address
`padr` -> `dst_port`: Remote station port number
`padr` -> `src_port`: Local station port number
(If this is set to `0`, any port number from 1024 to 2047 is used.)
`padr` -> `keepalive`: Transmission interval for alive monitoring packets (*)
`padr` -> `ttl`: Always set `0`. (Time to live is fixed to `30`.)

Details of output parameters

Return value: A registered socket ID or error code is returned.
(`/0001` to `/0060`): Registered socket ID
(`/F000` to `/FFFF`): Error
For the error codes, see 1.6.2 Error codes in PART 2.

(*) The transmission interval for alive monitoring packets is set in seconds.
The specifiable range is from `0` to `75`. If `0` is set, alive monitoring packets are not transmitted.
This function is an added function of the S10VE ET.NET module. After a connection is established, if no response is received from the communication destination ten times in a row, the connection is disconnected.

short tcp_popen(mj_no, ch_no, &padr)

Function

This function registers sockets for TCP/IP programs, and puts those sockets in a passive state.
The return value is either the registered parent socket ID (connection socket) or an error code.
This function is equivalent to `socket+bind+listen` in UNIX.
If `dst_ip` or `dst_port` is set to `0`, connection requests can be received from any remote station.
In addition, if `src_port` is set to `0`, any port from 1024 to 2047 can be secured.

Linking procedure

Main module CH1 example

```
struct popen_p{
    long  dst_ip;
    short dst_port;
    short src_port;
    char  listennum;
    char  ttl;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct popen_p padr;
    padr.dst_ip   = 0xC0A80001;
    padr.dst_port  = 10000;
    padr.src_port  = 1000;
    padr.listennum = 0;
    padr.ttl      = 0;
    rtn = tcp_popen( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2)
`padr`: Starting address of input parameters (Make sure that you specify an even number as the address.)
`padr` -> `dst_ip`: Remote station IP address (If no remote station is specified, `0` is set.)
`padr` -> `dst_port`: Remote station port number (If no remote station is specified, `0` is set.)
`padr` -> `src_port`: Local station port number
`padr` -> `listennum`: Fixed to `0`
`padr` -> `ttl`: Always set `0`. (Time to live is fixed to `30`.)

Details of output parameters

The return value is either a registered socket ID or an error code.
(`/0001` to `/0060`): Registered parent socket ID (connection socket)
(`/F000` to `/FFFF`): Error
For the error codes, see 1.6.2 Error codes in PART 2.

- This function's return value is the parent socket ID (connection socket). (This is different from the socket ID for transmitting and receiving `tcp_send/tcp_recv`.)
- The return value for `tcp_accept` is the child socket ID (communication socket).
- When closing sockets (`tcp_close`/`tcp_abort`), you need to issue separate instructions to close the parent and child sockets.
- When using this function in a similar way to the existing S10V ET.NET module device, after `tcp_accept` ends normally, close the parent socket (`tcp close`). See the notes on `tcp accept`.

short tcp_accept(mj_no, ch_no, &padr)

Function

This function waits for a connection request (reception of SYN) to a socket ID put in the passive state by `tcp_popen()`, and accepts the establishment of a connection.
The return value is either a child socket ID (communication socket) registered after establishment of the connection, or an error code.
The socket ID input parameter differs from the socket ID registered after establishment of the connection. This function waits continuously until a connection is made with the remote station.

Linking procedure

Main module CH1 example

```
struct accept_p{
    short  s_id;
    char   keepalive;
    char   notuse;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct accept_p padr;
    padr.s_id     = 1;
    padr.keepalive = 0;
    padr.notuse    = 0;
    rtn = tcp_accept( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

mj_no: Module identifier (1: Main; 2: Sub)
ch_no: Channel number (1: CH1; 2: CH2)
padr: Starting address of input parameters (Make sure that you specify an even number as the address.)
padr -> s_id: Socket ID, parent socket ID (for connection)
padr -> keepalive: Transmission interval for alive monitoring packets (*)
padr -> notuse: Fixed to 0 (unused)

Details of output parameters

The return value is either the registered socket ID or an error code.
(/0001 to /0060): Registered child socket ID (communication socket)
(/F000 to /FFFF): Error
For the error codes, see 1.6.2 Error codes in PART 2.

When the tcp_accept function is issued, one child socket is reserved. Therefore, if there are no available sockets, a socket number exceeded error (error code: /F011) is returned.

(*) Specify the transmission interval for alive monitoring packets in seconds.
The specifiable range is from 0 to 75. If 0 is set, alive monitoring packets are not transmitted.
After a connection is established, if no response is received from the communication destination ten times in a row, the connection is disconnected.

- The return value of this function is the child socket ID (communication socket). (This is different from the socket ID returned by tcp_popen.)
- For TCP transmission and reception (tcp_send/tcp_recv), specify a child socket ID.
- Also specify a child socket ID when acquiring TCP socket information or connection status (tcp_getaddr/tcp_stat). See the notes on tcp popen.

short tcp_close(mj_no, ch_no, &padr)

Function

This function ends the connection corresponding to a socket ID, and deletes the socket.
The return value is the processing result.
This function transmits an FIN, and waits for the connection to end (to receive an FIN from the remote station).
If there is no response from the remote station, the socket driver times out after 30 seconds, returning an error (error code: /F012). If this happens, issue `tcp_abort()`.

Linking procedure

Main module CH1 example

```
struct close_p{
    short s_id;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct close_p padr;
    padr.s_id = 1;
    rtn = tcp_close( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of input parameters (Make sure you specify an even number as the address.)
`padr` -> `s_id`: Socket ID

Details of output parameters

The return value is the processing result.
(`0`): The processing ended normally.
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short tcp_abort(mj_no, ch_no, &padr)

Function

This function forcibly ends the connection corresponding to the socket ID (transmits an RST), and deletes the socket. The return value is the processing result.

Linking procedure

Main module CH1 example

```
struct abort_p{
    short s_id;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct abort_p padr;
    padr.s_id = 1;
    rtn = tcp_abort( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `s_id`: Socket ID

Details of output parameters

The return value is the processing result.
(`0`): The processing ended normally.
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short tcp_getaddr(mj_no, ch_no, &padr)

Function

This function acquires the following information on the connection corresponding to the socket ID: IP address of the remote station, local station port number, and remote station port number. The return value is the processing result. When the processing result ends normally, the information acquired by `outinf` is enabled.

Linking procedure

Main module CH1 example

```
struct sid_p{
    short s_id;
};
struct getaddr_p{
    long  ipaddr;
    short src_port;
    short dst_port;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct sid_p    padr;
    struct getaddr_p outinf;
    padr.s_id = 1;
    rtn = tcp_getaddr( mj_no, ch_no, &padr, &outinf );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `s_id`: Socket ID (in the case of a TCP server, specify a child socket ID.)

Details of output parameters

`outinf`: Starting address of the output parameters (Make sure that you specify an even number as the address.)
`outinf` -> `ipaddr`: Remote station IP address
`outinf` -> `src_port`: Local station port number
`outinf` -> `dst_port`: Remote station port number
The return value is the processing result.
(`0`): The processing ended normally.
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short tcp_stat(mj_no, ch_no, &padr)

Function

This function acquires the status of the connection corresponding to the socket ID.
The return value is the processing result.
If the processing result indicates that the processing ended normally, the information acquired by `outinf` is enabled.

Linking procedure

Main module CH1 example

```
struct sid_p{
    short s_id;
};
struct stat_p{
    unsigned short stat;
    unsigned short urg;
    unsigned short sendwin;
    unsigned short recvwin;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct sid_p  padr;
    struct stat_p outinf;
    padr.s_id = 1;
    rtn = tcp_stat( mj_no, ch_no, &padr, &outinf );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `s_id`: Socket ID (In the case of a TCP server, specify a child socket ID.)

Details of output parameters

`outinf`: Starting address of the output parameters (Make sure that you specify an even number as the address.)
`outinf` -> `stat`: Connection status
`0`: CLOSED
`1`: LISTEN
`2`: SYN_SENT
`3`: SYN_RECEIVED
`4`: ESTABLISHED
`5`: CLOSE_WAIT
`6`: FIN_WAIT_1
`7`: CLOSING
`8`: LAST_ACK
`9`: FIN_WAIT_2
`10`: TIME_WAIT
`outinf` -> `urg`: Existence of urgent data
`0`: No urgent data
Other than `0`: Number of urgent data items
`outinf` -> `sendwin`: Remaining capacity of the transmission window
`outinf` -> `recvwin`: Amount of reception data that arrived
The return value is the processing result.
(`0`): The processing ended normally.
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short tcp_send(mj_no, ch_no, &padr)

Function

This function transmits the data of the `buf` to `len` parameters of the connection corresponding to the socket ID. The return value is the processing result.
This function returns a value when data is stored in the transmission window.
Check the data transmission state according to the remaining capacity of the transmission window of `tcp_stat()`.
If the transmission window is full, the transmission waits until there is available capacity.
If the wait times out, the return value is `/F012`.
In this case, if you are using a TCP client, perform `tcp_abort()` and reissue the functions starting from `tcp_open()`.
If you are using a TCP server, perform `tcp_abort()` for the child socket and reissue the functions starting from `tcp_accept()`.
If the parent socket is already closed, reissue the functions starting from `tcp_popen()`, not from `tcp_accept()`.

Remarks:

To avoid making `tcp_send()` transmissions wait, when issuing `tcp_send()` multiple times in succession (when transmitting a total of 4096 bytes or more without performing a TCP reception check via `tcp_receive()`), issue `tcp_stat()` before issuing `tcp_send()`, and check whether the remaining capacity of the transmission window is the same length as or longer than the transmission data.

Linking procedure

Main module CH1 example

```
struct send_p{
    short s_id;
    short len;
    char  *buf;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct send_p padr;
    padr.s_id = 1;
    padr.len  = 1024;
    padr.buf  = 0x50000000;
    rtn = tcp_send( mj_no, ch_no, &padr );
}
```

Parameter

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `s_id`: Socket ID (In the case of a TCP server, specify a child socket ID.)
`padr` -> `len`: Transmission data length (1 to 1460 bytes)
`padr` -> `buf`: Starting address of the transmission data (Make sure that you specify an even number as the address.)

Details of output parameters

The return value is the processing result.
(`0`): The processing ended normally. (All data was stored in the transmission window.)
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short tcp_receive(mj_no, ch_no, &padr)

Function

This function receives the `len` parameter data from the connection corresponding to the socket ID, in the `buf` parameter. The return value is the processing result.
Although you can specify the reception wait time for the `tim` parameter, this function returns a value at the time data is received, regardless of whether the reception wait time has elapsed.

Linking procedure

Main module CH1 example

```
struct receive_p{
    short s_id;
    short len;
    char  *buf;
    long  tim;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct receive_p padr;
    padr.s_id = 1;
    padr.len  = 2048;
    padr.buf  = 0x50001000;
    padr.tim  = 0;
    rtn = tcp_receive( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `s_id`: Socket ID (In the case of a TCP server, specify a child socket ID.)
`padr` -> `len`: Reception data length (1 to 1460 bytes)
`padr` -> `buf`: Starting address of the reception data (Make sure that you specify an even number as the address.)
`padr` -> `tim`: Reception wait time (ms: 0 to 86,400,000 (24 hours))

Details of output parameters

The return value is the processing result.
(`0`): The processing ended normally (no reception data).
(`/0001` to `/05B4`): The processing ended normally (number of received bytes).
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short udp_open(mj_no, ch_no, &padr)

Function

This function registers a UDP/IP program socket, and secures a port. The return value is either the registered socket ID or an error code.
If `0` is set for the `dst_ip` parameter, packets can be received from any host.
If `0` is set for the `dst_port` parameter, data can be received from any port.
If `0` is set for the `src_port` parameter, an unused port from 1024 to 2047 is secured.

Linking procedure

Main module CH1 example

```
struct uopen_p{
    long  dst_ip;
    short dst_port;
    short src_port;
    char  pktmode;
    char  ttl;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct uopen_p padr;
    padr.dst_ip   = 0xC0A800101;
    padr.dst.port = 1000;
    padr.src_port = 1000;
    padr.pktmode  = 0;
    padr.ttl      = 0;
    rtn = udp_open( mj_no, ch_no, &padr );
}
```

Parameter

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `dst_ip`: Remote station IP address
`padr` -> `dst_port`: Remote station port number
`padr` -> `src_port`: Local station port number
`padr` -> `pktmode`: Maximum reception buffer size (*)
`padr` -> `ttl`: Always specify `0`. (Time to live is fixed to `30`.)

Details of output parameters

The return value is either the registered socket ID or an error code.
(`/0201` to `/0260`): Registered socket ID
(`/F000` to `/FFFF`): Error
For the error codes, see 1.6.2 Error codes in PART 2.

(*) The maximum reception buffer size is specified in kilobytes.
The specifiable range is from `0` to `18`. If `0` is set, a value of 6.1 KB (default value) is used. Even if a reception request from the `udp_receive` function is delayed, you can increase the number of packets that can be accumulated.
The number of packets that can be accumulated is 4 to 6 by default, and 12 to 16 when 18 KB is specified. (The number of packets that can be accumulated differs according to the reception data size.)

short udp_close(mj_no, ch_no, &padr)

Function

This function deletes the socket corresponding to the socket ID.
The return value is the processing result.

Linking procedure

Main module CH1 example

```
struct uclose_p{
    short s_id;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct uclose_p padr;
    padr.sid = 0x201;
    rtn = udp_close( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1, `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `s_id`: Socket ID

Details of output parameters

The return value is the processing result.
(`0`): The processing ended normally.
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short udp_send(mj_no, ch_no, &padr)

Function

This function transmits the data of the `buf` to `len` parameters to the socket corresponding to the socket ID.
The return value is the processing result.
Of the `dst_ip` and `dst_port` specifications, those of `udp_open()` are prioritized.

Linking procedure

Main module CH1 example

```
struct usend_p{
    short s_id;
    short notuse;
    long  dst_ip;
    short dst_port;
    short len;
    char  *buf;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct usend_p padr;
    padr.s_id    = 0x201;
    padr.notuse   = 0;
    padr.dst_ip   = 0xC0A80101;
    padr.dst_port = 1000;
    padr.len     = 1024;
    padr.buf     = 0x50002000;
    rtn = udp_send( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

mj_no: Module identifier (1: Main; 2: Sub)
ch_no: Channel number (1: CH1; 2: CH2) (unused)
padr: Starting address of the input parameters (Make sure that you specify an even number as the address.)
padr -> s_id: Socket ID
padr -> notuse: Fixed to 0 (unused)
padr -> dst_ip: Remote station IP address
padr -> dst_port: Remote station port number
padr -> len: Transmission data length (byte count: 1 to 1472)
padr -> buf: Starting address of transmission data (Make sure that you specify an even number as the address.)
If a value other than 0 is specified for udp_open(), the dst_ip and dst_port values for udp_open() are used.

Details of output parameters

The return value is the processing result.
(0): The processing ended successfully.
(/F000 to /FFFF): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

■ On the specifications of dst_ip and dst_port
- If a value other than 0 is specified for udp_open(), the parameters specified for udp_open() are used.
- If 0 is specified for udp_open(), the parameters specified for udp_send() are used.
- If 0 is specified for both udp_open() and udp_send(), an invalid address error code (error code: /FFF0) is returned.

short udp_receive(mj_no, ch_no, &padr)

Function

This function receives the `buf` parameter data from the socket corresponding to the socket ID.
The return value is the processing result.
With this function, the reception wait time can be specified in the `tim` parameter.
However, this function returns a value at the time data is received, regardless of whether the reception wait time has elapsed.

Linking procedure

Main module CH1 example

```
struct ureceive_p{
    short s_id;
    short notuse;
    char  *buf;
    long  tim;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct ureceive_p padr;
    padr.s_id   = 0x201;
    padr.notuse = 0;
    padr.buf    = 0x50003000;
    padr.tim    = 0;
    rtn = udp_receive( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `s_id`: Socket ID
`padr` -> `notuse`: Fixed to `0` (unused)
`padr` -> `buf`: Starting address of the reception buffer (Make sure that you specify an even number as the address.)
`padr` -> `tim`: Reception wait time (ms: 0 to 86,400,000 (24 hours))

Details of output parameters

The return value is the processing result.
(`0`): The processing ended normally (no reception data).
(`/0001` to `/05C0`): The processing ended normally (number of received bytes).
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

> `udp_receive()` receives data on a per-packet basis. Accordingly, ensure a buffer area of 1,472 bytes.

short route_list(mj_no, ch_no, &padr)

Function

This function acquires routing information (the maximum size of the routing information table is 34).
The return value is the number of acquired entries.
If `0` is specified for the `len` parameter, the number of registered entries is returned.
For `len`, specify a multiple of 16 bytes.

Linking procedure

Main module CH1 example

```
struct lstrt_p{
    short len;
    short notuse;
    void  *buf;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct lstrt_p padr;
    padr.len    = 544;
    padr.notuse = 0;
    padr.buf    = 0x50008000;
    rtn = route_list( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

mj_no: Module identifier (1: Main; 2: Sub)
ch_no: Channel number (1: CH1; 2: CH2) (unused)
padr: Starting address of the input parameters (Make sure that you specify an even number as the address.)
padr -> len: Data length (byte count: multiple of 16)
padr -> notuse: Fixed to 0 (unused)
padr -> buf: Starting address of the data (Make sure that you specify an even number as the address.)

Details of output parameters

The return value is the number of acquired entries.
(0): No entries
(/0001 to /0022): Number of acquired entries
Acquired data structure (contents of buf)

```
typedef struct{
unsigned long dstaddr: Remote station IP address
unsigned long gtwayaddr: Gateway IP address
unsigned short metric: Metric (number of gateway routes)
unsigned short rt_types: Type
unsigned short refcnt: Reference counter
unsigned short notuse: (Unused)
}routeentry
```

For the error codes, see 1.6.2 Error codes in PART 2.

■ On the data length (len) specification

If the specified data length (len) is smaller than the registered data length (number of registered entries x the size of one entry (16 bytes)), you can only acquire routing information for the specified data length's worth of data (the return value is the number of acquired entries). If a value smaller than the size of one entry is specified, the return value is 0.
To acquire all registered routing information, specify 0 for the data length (len), issue route_list(), and acquire the number of registered entries. Then, specify a data length that is the number of registered entries times the size of one entry (16 bytes), and issue route_list().
Alternatively, specify a data length that is the maximum number of registered entries (34 entries) times the size of one entry (16 bytes), and then issue route_list().

short route_del(mj_no, ch_no, &padr)

Function

This function deletes routing information from the routing information table.
The return value is the processing result.

Linking procedure

Main module CH1 example

```
struct delrt_p{
    long dstaddr;
    long gtwayaddr;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct delrt_p padr;
    padr.dstaddr   = 0xC0A80300;
    padr.gtwayaddr = 0xC0A80110;
    rtn = route_del( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `dstaddr`: Remote station network address
`padr` -> `gtwayaddr`: Gateway IP address

Details of output parameters

The return value is the processing result.
(`0`): The processing ended normally.
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short route_add(mj_no, ch_no, &padr)

Function

This function registers routing information to the routing information table.
The return value is the processing result.
If you cannot register information because the routing information table is full, an internal buffer insufficiency error (error code: /FFFF) is returned. Delete unnecessary routing information by using `route_del()`, and then reissue the function.

Linking procedure

Main module CH1 example

```
struct addrt_p{
    long  dstaddr;
    long  gtwayaddr;
    short metric;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct addrt_p padr;
    padr.dstaddr   = 0xC0A80400;
    padr.gtwayaddr = 0xC0A80111;
    padr.metric    = 1;
    rtn = route_add( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `dstaddr`: Remote station network address
`padr` -> `gtwayaddr`: Gateway IP address
`padr` -> `metric`: Metric (number of gateway routes)

Details of output parameters

The return value is the processing result.
(`0`): The processing ended normally.
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short arp_list(mj_no, ch_no, &padr)

Function

This function acquires ARP information (the maximum size of the ARP information table is 32).
The return value is the number of acquired entries.
If `0` is specified for the `len` parameter, the number of registered entries is returned.
For `len`, specify a multiple of 12 bytes.

Linking procedure

Main module CH1 example

```
struct lstarp_p{
    short len;
    short notuse;
    void  *buf;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct lstarp_p padr;
    padr.len   = 384;
    padr.notuse = 0;
    padr.buf   = 0x50009000;
    rtn = arp_list( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `len`: Data length (number of bytes: multiple of 12)
`padr` -> `notuse`: Fixed at `0` (unused)
`padr` -> `buf`: Starting address of the data (Make sure that you specify an even number as the address.)

Details of output parameters

The return value is the number of acquired entries.
(`0`): No entries
(`/0001` to `/0020`): Number of acquired entries
Acquired data structure (contents of `buf`)

```
typedef struct{
unsigned long dstaddr: Remote station IP address
unsigned char et_addr[6]: Remote station physical address
unsigned char ar_timer: Timer
unsigned char ar_flags: Flag
}arpt_t
```

■ On the data length (`len`) specification

If the specified data length (`len`) is smaller than the registered data length (number of registered entries × the size of one entry (12 bytes)), you can only acquire the specified data length's worth of ARP information (and the return value is the number of acquired entries).
If the specified value is smaller than the size of one entry, the return value is `0`.
To acquire all registered ARP information, specify `0` for the data length (`len`), issue `arp_list()`, and acquire the number of registered entries. Then, specify a data length that is the number of registered entries times the size of one entry (12 bytes) and issue `arp_list()`.
Alternatively, specify a data length that is the maximum number of registered entries (32 entries) times the size of one entry (12 bytes), and then issue `arp_list()`.

short arp_del(mj_no, ch_no, &padr)

Function

This function deletes ARP information from the ARP information table.
The return value is the processing result.

Linking procedure

Main module CH1 example

```
struct delarp_p{
    long ipaddr;
    char etaddr[6];
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct delarp_p padr;

    padr.ipaddr   = 0xC0A80108;
    padr.etaddr[0] = 0xE0;
    padr.etaddr[1] = 0xDB;
    padr.etaddr[2] = 0x18;
    padr.etaddr[3] = 0x3C;
    padr.etaddr[4] = 0x50;
    padr.etaddr[5] = 0x11;
    rtn = arp_del( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `mj_no`: Module identifier (`1`: Main; `2`: Sub)
`padr` -> `ch_no`: Channel number (`1`: CH1; `2`: CH2)
`padr` -> `ipaddr`: Remote station IP address
`padr` -> `etaddr[6]`: Remote station physical address

Details of output parameters

The return value is the processing result.
(`0`): The processing ended normally.
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short arp_add(mj_no, ch_no, &padr)

Function

This function registers ARP information to the ARP information table.
The return value is the processing result.
If you cannot register information because the ARP information table is full, an internal buffer insufficiency error (error code: /FFFF) is returned. Delete unnecessary ARP information by using `arp_del()`, and then reissue the function.

Linking procedure

Main module CH1 example

```
struct addarp_p{
    long  ipaddr;
    char  etaddr[6];
    short flag;
};
main()
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct addarp_p padr;
    padr.ipaddr   = 0xC0A80108;
    padr.etaddr[0] = 0xE0;
    padr.etaddr[1] = 0xDB;
    padr.etaddr[2] = 0x18;
    padr.etaddr[3] = 0x3C;
    padr.etaddr[4] = 0x50;
    padr.etaddr[5] = 0x11;
    padr.flag     = 0;
    rtn = arp_add( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `ipaddr`: Remote station IP address
`padr` -> `etaddr[6]`: Remote station physical address
`padr` -> `flag`: Flag (fixed at `0`)

Details of output parameters

The return value is the processing result.
(`0`): The processing ended normally.
(`/F000` to `/FFFF`): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

short getconfig(mj_no, ch_no, &padr)

Function

This function acquires configuration blocks.
The return value is the processing result.

Linking procedure

Main module CH1 example

```
struct config_p{
    void *config_ptr;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct config_p padr;
    padr.config_ptr = 0x5000A000;
    rtn = getconfig( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

`mj_no`: Module identifier (`1`: Main; `2`: Sub)
`ch_no`: Channel number (`1`: CH1; `2`: CH2) (unused)
`padr`: Starting address of the input parameters (Make sure that you specify an even number as the address.)
`padr` -> `config_ptr`: Starting address of the configuration block

Details of output parameters

The return value is the processing result.
(`0`): The processing ended successfully.

Details of configuration blocks

A configuration block consists of the following data structure:

```
struct config_ptr{
long ip_addr: Local station IP address (network order) (optional)
long netmask: Subnet mask (optional)
long broadcast: Broadcast address (optional)
char tcp_num: Maximum number of TCP sockets (96)
char udp_num: Maximum number of UDP sockets (96)
char rt_num: Routing information table size (34)
char arp_num: ARP information table size (32)
short tcp_win: TCP transmission and reception window size (4096)
};
```

short set_so_abort(mj_no, ch_no, &padr)

Function

This function is issued when you want to close a specified socket ID when a specified task is aborted.
In the case of a TCP socket, the connection is forcibly closed (an RST is transmitted) in a similar way to the `tcp_abort` function, and the socket is closed.
If the task ends (if an exit is performed), the socket is not closed.
If the specified task number is outside the user-specified range, an invalid argument error (error code: /FFF3) is returned.

Linking procedure

Main module CH1 example

```
struct so_abort_p{
    short s_id;
    short task_no;
};
main()
{
    int mj_no = 1;
    int ch_no = 1;
    short rtn = 0;
    struct so_abort_p padr;
    padr.s_id   = 1;
    padr.task_no = 15;
    rtn = set_so_abort( mj_no, ch_no, &padr );
}
```

Parameters

Details of input parameters

mj_no: Module identifier (1: Main; 2: Sub)
ch_no: Channel number (1: CH1; 2: CH2) (unused)
padr: Starting address of the input parameters (Make sure that you specify an even number as the address.)
padr -> s_id: Socket ID
padr -> task_no: Task number

Details of output parameters

The return value is the processing result.
(0): The processing ends normally.
(/F000 to /FFFF): An error occurred.
For the error codes, see 1.6.2 Error codes in PART 2.

If this function is issued multiple times for the same socket ID, only the task number for the last time the function was issued is valid.

1.6.2 Error codes

The following shows what corrective action to take for the error codes of ET.NET socket handlers.

Table 2-2 List of Error Codes of ET.NET Socket Handlers (1/2)

Error code	Contents	Cause
/F000	Connection not established	A connection has not been established.
/F010	Invalid socket ID	• The socket ID is out of range. TCP: /01 ≤ ID ≤ /60 UDP: /201 ≤ ID ≤ /260 • An unused socket ID or freed socket ID was specified. • A connection has not been made, or has not been established. Alternatively, a connection has already been established (only `tcp_accept()`). • If the socket was closed due to a task abort, a record might be made in the error trace log.
/F011	Number of sockets exceeded	• The number of registered sockets exceeds the maximum (96 total including both TCP and UDP). • The system is retaining sockets temporarily (for 20 seconds).
/F012	Socket driver timeout	• There was no response from the ET.NET module after the fixed duration. • A timeout occurred because the transmission window was full (only `tcp send()`).
/F013	Module stopped	• An unrecoverable error was detected in the ET.NET module. • The ET.NET module has not been mounted.
/F020	Invalid transmission data length	The transmission data length is not within the valid range. TCP: 1 ≤ data length ≤ 1460 UDP: 1 ≤ data length ≤ 1472
/F021	Invalid reception data length	The reception data length is not within the valid range (only TCP). TCP: 1 ≤ data length ≤ 1460
/F0FE	Socket freed	• The socket was closed after the handler was started. • If the socket is closed due to a task abort, a record might be made in the error trace log.
/F0FF	Port function	The port was freed (RST received) after the handler was started.
/FFF0	Invalid address	• Both `udp_open()` and `udp_send()` have `0` set as the remote station's IP address and port number. • Data was transmitted to a remote station whose routing information has not been set (`udp_send()`, `tcp_open()`). • The broadcast address is set as the remote station's IP address (`tcp open()`).

Table 2-2 List of Error Codes of ET.NET Socket Handlers (2/2)

Error code	Contents	Cause
/FFF3	Invalid argument	• An invalid parameter was specified. • A request was received for a socket whose port was already freed (only `tcp abort()` is issued, and the socket is deleted).
/FFF5	Connection timeout	There is no response from the remote station.
/FFF8	FIN reception	An FIN was received from the remote station.
/FFFA	Connection forcibly ended	The remote station forcibly ended the connection (an RST was received) (after the RST was received, `tcp_receive()` was issued).
/FFFD	Duplicate socket error	The same socket (remote station IP address, remote station port number, or local station port number) already exists.
/FFFE	Invalid control block	The request was received after the connection was disconnected.
/FFFF	Internal buffer insufficiency	• There is insufficient capacity in the ET.NET module management buffer. • When the router route was communicated, the router received an error notification. • When the routing information or ARP information was registered, the maximum number of registrations was exceeded.

1.6.3 Steps to take in the event of an error

The following sections show the steps to take in the event of an error when issuing an ET.NET socket handler.

(1) Steps to take in the event of a TCP client-side error

The following shows what steps to take in the event of a TCP client-side error.

Table 2-3 Steps to Take in the Event of a TCP Client-side Error

Error code / Handler function	/F000	/F010	/F011	/F012	/F013	/F020	/F021	/F0FE	/F0FF	/FFF0	/FFF3	/FFF5	/FFF8	/FFFA	/FFFD	/FFFE	/FFFF
tcp_open	–	–	(1)	(4)	(4)	–	–	–	(5)	(9)	(3)	–	–	–	(7)	–	(8)
set_so_abort	–	(3)	–	–	–	–	–	–	–	–	(3)	–	–	–	–	–	–
tcp_send	(2)	(3)	–	(4)	(4)	(3)	–	(6)	–	–	(3)	(10)	–	(10)	–	(2)	(8)
tcp_receive	(2)	(3)	–	(4)	(4)	–	(3)	(6)	–	–	(3)	(10)	(2)	(10)	–	(2)	(8)
tcp_getaddr/ tcp_stat	(2)	(3)	(1)	(4)	(4)	–	–	(6)	–	–	(3)	(10)	–	(10)	–	(2)	(8)
tcp_close	–	(3)	–	(4)	(4)	–	–	–	–	–	(3)	–	–	–	–	–	–
tcp_abort	–	(3)	–	(4)	(4)	–	–	–	–	–	(11)	–	–	–	–	–	–

Details of steps:

(1) If tcp_close is issued because an FIN/RST was not received, reissue the function after waiting 20 seconds, because the system retains sockets for 20 seconds. If the problem persists, check the program (for unclosed sockets).
(2) Perform tcp_close or tcp_abort, and reissue the functions starting from tcp_open.
(3) Check the program (for mistakes in the function parameter specifications).
(4) Check the ET.NET module (mount state or whether the module has stopped abnormally).
(5) Reissue the function. If the problem persists, check the server-side program (whether the corresponding port is open).
(6) Reissue the functions starting from tcp_open.
(7) Reissue the function after about 20 seconds. If the problem persists, check the program (for duplicate port numbers or unclosed sockets).
(8) Reissue the function.
(9) Check the program (for mistakes in the destination IP address, and whether the routing information has been defined).
(10) Perform tcp_close, and reissue the functions starting from tcp_open.
(11) Check the program (for mistakes in the function parameter specifications).
If the function is issued on a socket whose port has been freed, use tcp_close (if the function is issued on a socket whose port has been freed, the socket is deleted).

(2) Steps to take in the event of a TCP server-side error
The following shows the steps to take in the event of a TCP server-side error.

Table 2-4 Steps to Take in the Event of a TCP Server-side Error

Error code / Handler function	/F000	/F010	/F011	/F012	/F013	/F020	/F021	/F0FE	/F0FF	/FFF0	/FFF3	/FFF5	/FFF8	/FFFA	/FFFD	/FFFE	/FFFF
tcp_popen	−	−	(1)	(4)	(4)	−	−	−	−	−	(3)	−	−	−	(7)	−	(5)
set_so_abort	−	(3)	−	−	−	−	−	−	−	−	(3)	−	−	−	−	−	−
tcp_accept	−	(3)	(1)	(4)	(4)	−	−	(8)	(5)	−	(3)	−	−	(5)	−	−	(5)
tcp_send	(2)	(3)	−	(4)	(4)	(3)	−	(6)	−	−	(3)	(9)	−	(9)	−	(2)	(5)
tcp_receive	(2)	(3)	−	(4)	(4)	−	(3)	(6)	−	−	(3)	(9)	(2)	(9)	−	(2)	(5)
tcp_getaddr/ tcp_stat	(2)	(3)	(1)	(4)	(4)	−	−	(6)	−	−	(3)	(9)	−	(9)	−	(2)	(5)
tcp_close	−	(3)	−	(4)	(4)	−	−	−	−	−	(3)	−	−	−	−	−	−
tcp_abort	−	(3)	−	(4)	(4)	−	−	−	−	−	(10)	−	−	−	−	−	−

Details of steps:
(1) If tcp_close is issued because an FIN/RST was not received, reissue the function after waiting 20 seconds, because the system retains sockets for 20 seconds. If the problem persists, check the program (for unclosed sockets).
(2) Perform tcp_close or tcp_abort on the child socket, and reissue the functions starting from tcp_accept. (*)
(3) Check the program (for mistakes in the function parameter specifications).
(4) Check the ET.NET module (mount state or whether the module stopped abnormally).
(5) Reissue the function.
(6) Reissue the functions starting from tcp_accept. (*)
(7) Reissue the function after about 20 seconds. If the problem persists, check the program (for duplicate port numbers or for unclosed sockets).
(8) Reissue the functions starting from tcp_popen.
(9) Perform tcp_close on the child socket, and reissue the functions starting from tcp_accept. (*)
(10) Check the program (for mistakes in the function parameter specifications).
If the function is issued on a socket whose port has been freed, use tcp_close (if the function is issued on a socket whose port has been freed, the socket is deleted).

(*) If the parent socket is already closed, reissue the functions starting from tcp_popen, not from tcp_accept.

(3) Steps to take in the event of a UDP error
The following shows the steps to take in the event of a UDP error.

Table 2-5 Steps to Take in the Event of a UDP Error

Handler function \ Error code	/F000	/F010	/F011	/F012	/F013	/F020	/F021	/F0FE	/F0FF	/FFF0	/FFF3	/FFF5	/FFF8	/FFFA	/FFFD	/FFFE	/FFFF
udp_open	−	−	(1)	(4)	(4)	−	−	(5)	−	−	(3)	−	−	−	(7)	−	(5)
set_so_abort	−	(3)	−	−	−	−	−	−	−	−	(3)	−	−	−	−	−	−
udp_send	−	(3)	−	(4)	(4)	(3)	−	(6)	−	(8)	(3)	−	−	−	−	(2)	(5)
udp_receive	−	(3)	−	(4)	(4)	−	−	(6)	−	−	(3)	−	−	−	−	(2)	(5)
udp_close	−	(3)	−	(4)	(4)	−	−	−	−	−	(3)	−	−	−	−	−	−

Details of steps:
(1) Check for any unclosed sockets, for example due to udp_close not being issued (if you are using a TCP socket, when tcp_close is issued due to FIN/RST not being received, reissue the function after waiting 20 seconds, because the system retains sockets for 20 seconds).
(2) Perform udp_close, and reissue the functions starting from udp_open.
(3) Check the program (for mistakes in the function parameter specifications).
(4) Check the ET.NET module (mount state or whether the module stopped abnormally).
(5) Reissue the function.
(6) Reissue the functions starting from udp_open.
(7) Check the program (for duplicate port numbers or unclosed sockets).
(8) Check the program (for mistakes in the destination IP address or whether the routing information has been defined).

1.6.4 Transmission timeout detection time

The following describes the TCP transmission timeout detection time for ET.NET socket handlers.
If a socket library function is issued and an ACK packet timeout occurs, for example due to a communication error or a remote device going down, the timeout detection time is as shown in Table 2-6.
Accordingly, even if a timeout is detected in the ET.NET socket handler and the transmission is either retried or the connection reestablished, this period shown in Table 2-6 still elapses.
When designing the system, always assume a communication error will occur, and confirm that there are no problems with the timeout periods shown in Table 2-6.

Table 2-6 Timeout Detection Time

Item		Detection period	Contents
`tcp_open()` timeout detection time (SYN retry interval)		45 seconds	If there is no response from the remote device, a SYN retry is performed at the following intervals: 6 seconds, 6 seconds, 12 seconds (*1) A connection timeout is detected 46 seconds after `tcp_open()` is issued (return value: /FFF5).
`tcp_send()` timeout detection time (SEND retry interval)		46 seconds	If there is no response from the remote device, the SEND is retried at the following intervals: 1 second, 2 seconds, 4 seconds, 6 seconds, 8 seconds, 10 seconds (*2) (`tcp_send()` ends normally if all data is stored in the transmission window, even during a SEND retry.) If there is no response from the remote device and not all data can be stored in the transmission window, a connection timeout (return value: /FFF5) is detected 46 seconds after `tcp_send()` was issued, and the connection is forcibly ended.
Response timeout detection time	`tcp_open()`, `tcp_send()`	60 seconds	This is the period in which no response is detected after the command is issued.
	`tcp_popen()`, `tcp_close()`, `tcp_abort()`, `udp_open()`, `udp_close()`, `udp_send()`, `route_list()`, `route_del()`, `route_add()`, `arp_list()`, `arp_del()`, `arp_add()`, `getconfig()`, `tcp_getaddr()`, `tcp_stat()`	6 seconds	
	`tcp_accept()`, `tcp_recive()`, `udp_recive()`	–	Because these depend on when the remote device makes a transmission, timeout monitoring is not performed.

(*1) If there is no response within 21 seconds after a 12-second retry, the communication times out.
(*2) If there is no response within 15 seconds after a 10-second retry, the communication times out.

1.6.5 Procedure for issuing ET.NET socket handlers

The following shows the procedure for issuing ET.NET socket handlers.

(1) TCP/IP program (general example of a simultaneously connected client)
Condition: When alive monitoring packet (KeepAlive) transmissions can be made on the TCP server side, or reception timeout monitoring can be performed (transmission and reception at fixed cycles).

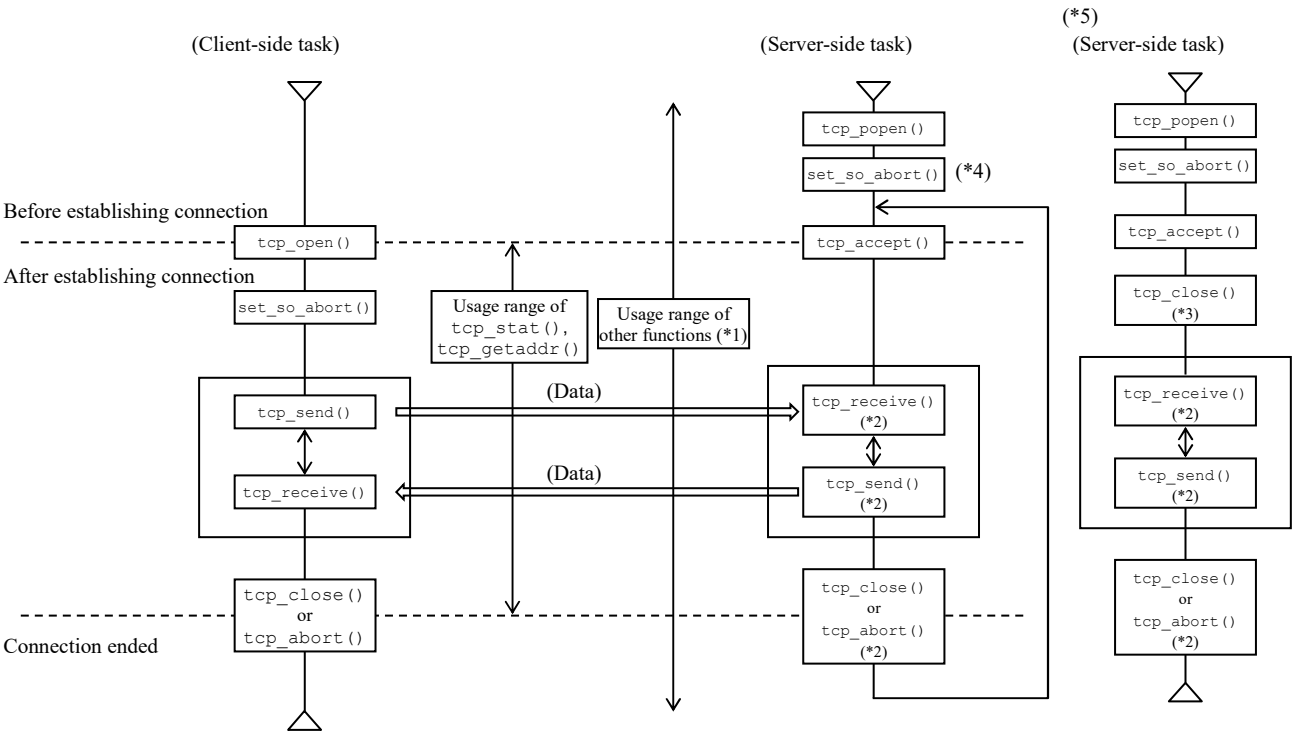


Figure 2-3 ET.NET Socket Handler Procedure for TCP/IP Programs (General Example of a Simultaneously Connected Client)

(*1) `route_list()`, `route_add()`, `route_del()`, `arp_list()`, `arp_add()`, `arp_del()`, `getconfig()`
(*2) Issued for a child socket ID (communication socket)
(*3) Issued for a parent socket ID (connection socket)
(*4) Because the task number issued for the parent socket ID is carried over to the child sockets, if the same task uses the parent and child sockets, the `set_so_abort` function does not need to be issued for the child socket ID.
(*5) An example is shown for using server-side tasks in a similar way to the past S10V ET.NET module.

(2) TCP/IP program (special example for a simultaneously connected client)
Condition: When alive monitoring packet (KeepAlive) transmissions cannot be made on the TCP server side, and reception timeout monitoring cannot be performed (no transmission or reception at fixed cycles).

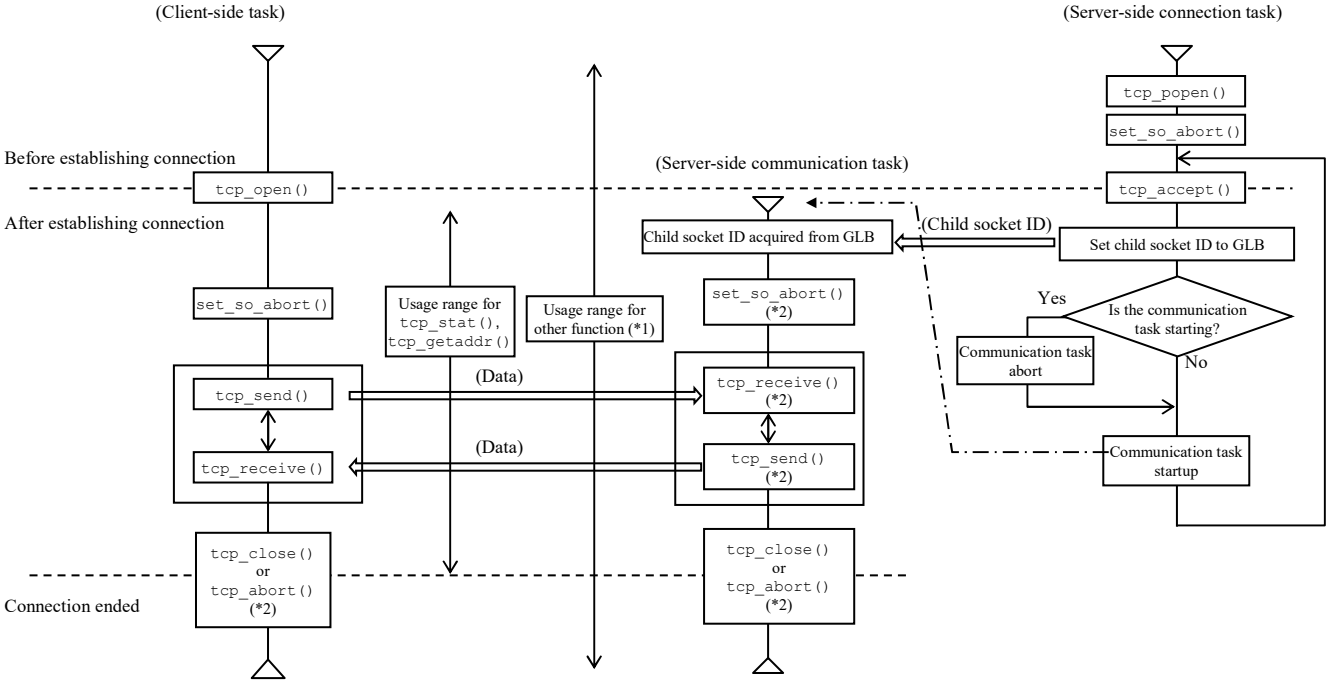


Figure 2-4 ET.NET Socket Handler Procedure for TCP/IP Programs (Special Example of a Simultaneously Connected Client)

(*1) `route_list()`, `route_add()`, `route_del()`, `arp_list()`, `arp_add()`, `arp_del()`, `getconfig()`
(*2) Issued for a child socket ID (communication socket)

(3) TCP/IP programs (example of multiple simultaneously connected clients)
The following shows some example server-side tasks of multiple simultaneously connected clients that have the same port number.
(For client-side tasks, the details shown in (1) apply.)

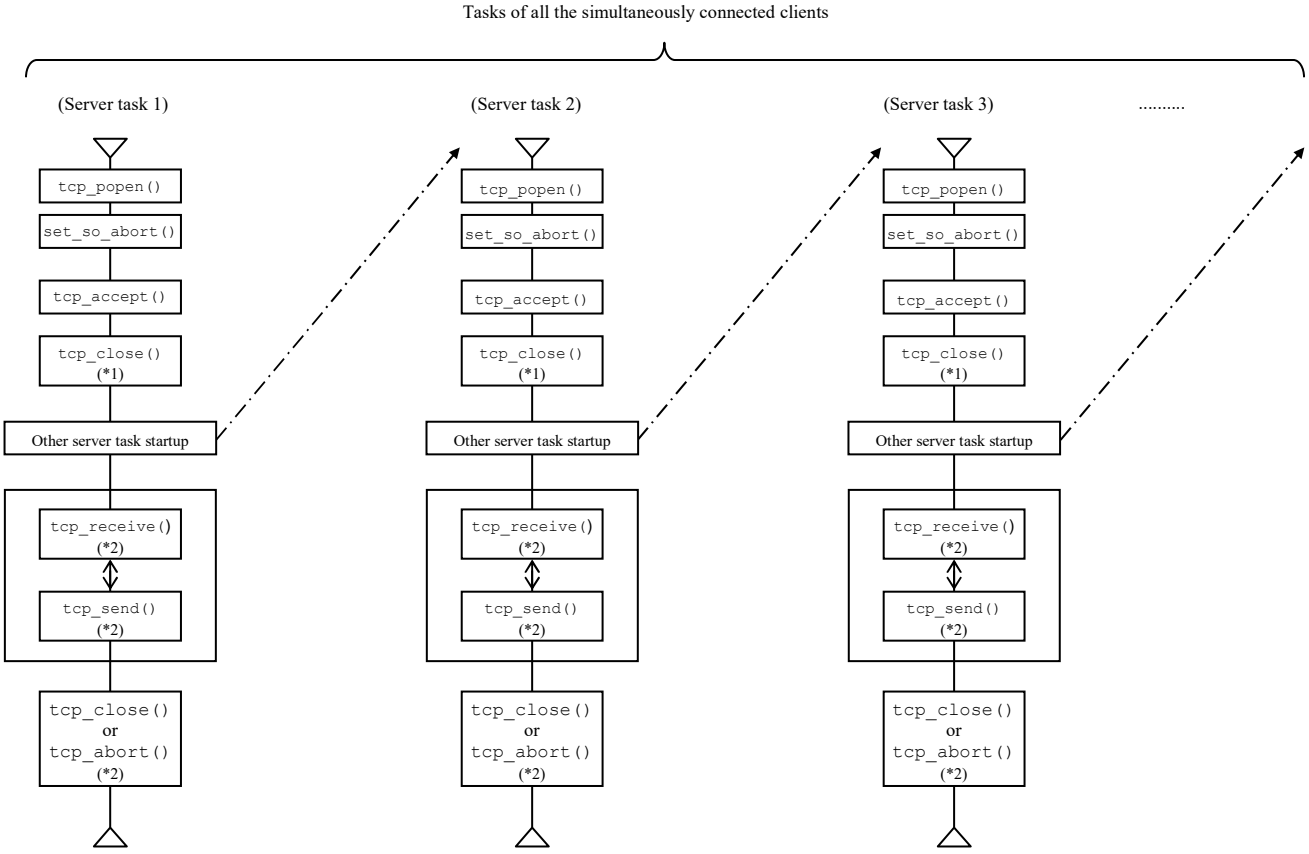


Figure 2-5 ET.NET Socket Handler Procedure for TCP/IP Programs (Example of Multiple Simultaneously Connected Clients)

(*1) Issued for a parent socket ID (connection socket)
(*2) Issued for a child socket ID (communication socket)

(4) UDP/IP programs (general example)

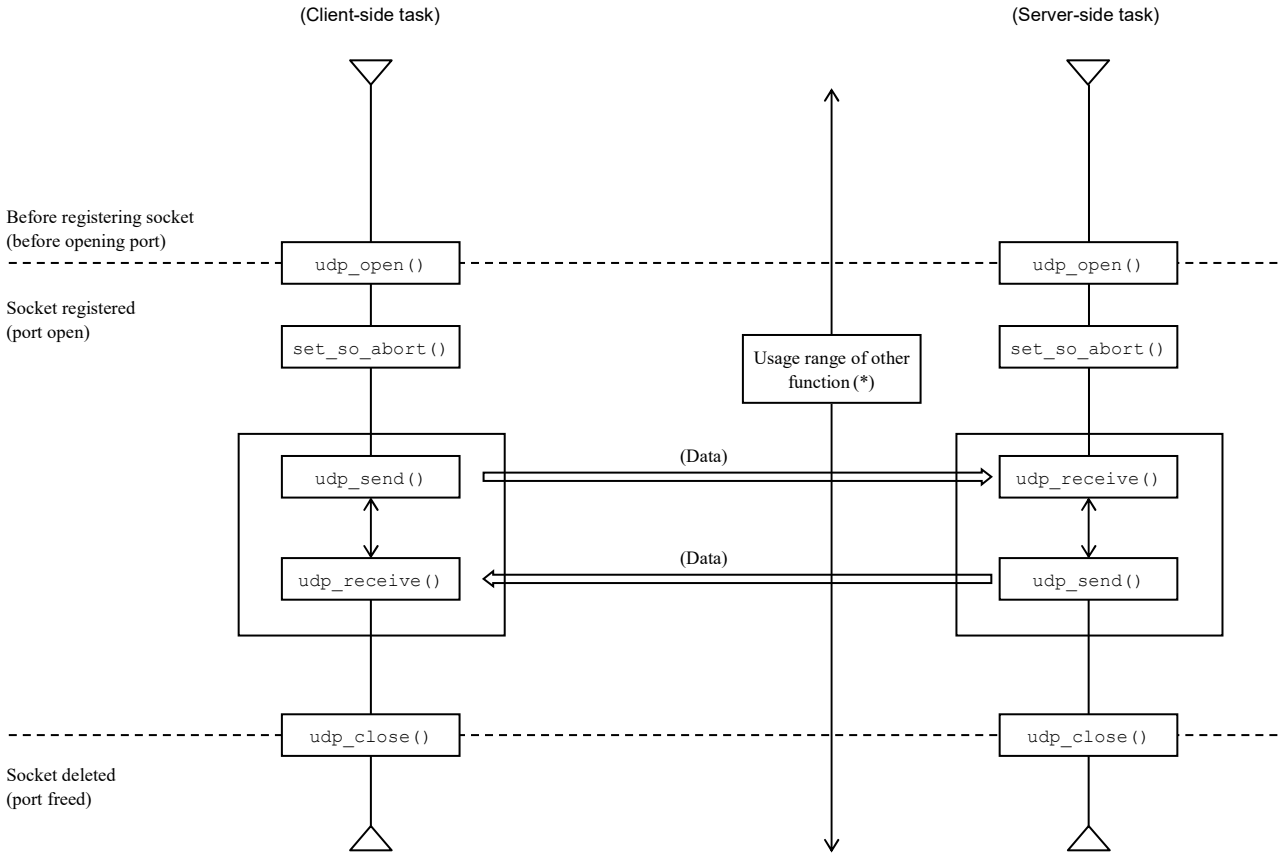


Figure 2-6 ET.NET Socket Handler Procedure for UDP/IP Programs (General Example)

(*) `route_list()`, `route_add()`, `route_del()`, `arp_list()`, `arp_add()`, `arp_del()`, `getconfig()`

This Page Intentionally Left Blank.

PART 3 LIBRARIES

CHAPTER 1 OVERVIEW

1.1 Conditions for Library Specification

When creating a program by using library subroutines, link libraries by specifying the `-l` option of the `svload` command. When linking libraries, note the following:

- If you are using a subroutine in `libcrs.a`, specify the `-lcrs` option of the `svload` command.

1.2 Order of Library Specification

When specifying a library by using the `svload` macro, note the following:

- If two or more libraries sharing the same name are to be specified, first specify the library containing the option file to be linked.

1.3 Names Defined in Libraries

The following lists the names defined in libraries. When creating programs, avoid using duplicate names. If you do use duplicate names, specify the order in which to specify library files after specifying the object file to be linked. This prevents the library files from being linked first.

- libcrs.a

  fpcheck fpchecko fpsetmask fpgetmask fpsetround fpgetround
  fpsetsticky fpgetsticky

Subroutines for controlling IEEE floating-point processing environments

Name

fpgetround, fpsetround, fpgetmask, fpsetmask, fpgetsticky, fpsetsticky - Control IEEE floating-point processing environments.

Format

```
#include <ieeefp.h>

typedef enum {
               FP_RN =0,  /* Round to nearest          */
               FP_RZ =1   /* Round to zero (truncate)  */
             } fp_rnd;

#define fp_except  int
#define FP_X_INV       0x10  /* Invalid operation exception   */
#define FP_X_OFL       0x04  /* Overflow exception            */
#define FP_X_UFL       0x02  /* Underflow exception           */
#define FP_X_DZ        0x08  /* Divide by zero exception      */
#define FP_X_IMP       0x01  /* Imprecise (loss of precision) */

fp_rnd       fpgetround(void);
fp_rnd       fpsetround(fp_rnd rnd_dir);
fp_except    fpgetmask(void);
fp_except    fpsetmask(fp_except mask);
fp_except    fpgetsticky(void);
fp_except    fpsetsticky(fp_except sticky);
```

Description

These macros control floating-point rounding and generation of floating-point exceptions.

(1) Rounding

Rounding is divided into two modes, and controlled by `fpgetround()` and `fpswtround()`.

`FP_RN`: Round to nearest

`FP_RZ`: Round to zero

The default rounding value is `FP_RN`.

(2) Floating-point exceptions

The following floating-point exceptions might occur in the S10VE CPU:

- FPU error (E): FPSCR.DN is `0`, and a non-normalized value is input. (*)
- Invalid operation (V): Invalid operation such as NaN input
- Division by zero (Z): Division by zero
- Overflow (O): Overflow of an operation result
- Underflow (U): Underflow of an operation result
- Imprecise exception (I): Occurrence of overflow, underflow, or rounding

(*) Because FPSCR.DN is set to `1` in the S10VE CPU, a non-normalized value is handled as `0`, and no FPU error occurs.

A floating-point exception occurs when `1` is set in the enable bit corresponding to the exception in the floating-point control register (FPSCR).

When a floating-point exception occurs, the bit corresponding to the FPU exception factor field in the floating-point control register (FPSCR) is set to `1`, and `1` is stacked in the bit corresponding to the FPU exception flag field. When no FPU exception occurs, the bit corresponding to the FPU exception factor field is cleared to `0`, and the bit corresponding to the FPU exception flag field is not changed.

The following are the default values of the enable bits in a floating-point exception:
• Invalid operation (V): Enabled
• Division by zero (Z): Enabled
• Overflow (O): Enabled
• Underflow (U): Disabled
• Imprecise exception (I): Disabled
Floating-point exceptions are controlled by `fpgetmask()`, `fpsetmask()`, `fpgetsticky()`, and `fpsetsticky()`.
• `fpgetround()` returns the current rounding mode.
`FP_RN`: Round to nearest
`P_RZ`: Round to zero
• `fpsetround()` sets the rounding mode and returns the previous rounding mode.
• `fpgetmask()` returns the current value of the FPSCR exception enable bit.
The following table lists exception masks and their corresponding FPSCR exception enable bits.

Exception mask	FPSCR enable bit
`FP_X_INV`	Invalid operation (V)
`FP_X_DZ`	Division by zero (Z)
`FP_X_OFL`	Overflow (O)
`FP_X_UFL`	Underflow (U)
`FP_X_IMP`	Imprecise exception (I)

• `fpsetmask()` sets the FPSCR exception enable bit according to the value of the exception mask, and returns the previous setting value.
Exception masks correspond to FPSCR exception enable bits in the same way as for `fpgetmask()`.
• `fpgetsticky()` returns the value of the FPU exception flag field.
The following table lists sticky flags and their corresponding FPSCR FPU exception flag fields.

Sticky flag	FPSCR flag field
`FP_X_INV`	Invalid operation (V)
`FP_X_DZ`	Division by zero (Z)
`FP_X_OFL`	Overflow (O)
`FP_X_UFL`	Underflow (U)
`FP_X_IMP`	Imprecise exception (I)

• `fpsetsticky()` sets the value of the FPU exception flag field according to the value of the sticky flag, and returns the previous setting value.
Sticky flags correspond to FPSCR FPU exception flag fields in the same way as for `fpgetsticky()`.

NOTICE

`fpsetsticky()` changes the values of all sticky flags' corresponding FPU exception flag fields.
`fpsetmask()` changes all exception mask values' corresponding exception enable bits.
The following modes are not available for rounding control by using `fpgetround()` and `fpsetround()`:

- FP_RP: Negative values are truncated and positive values are rounded up (round to plus)
- FP_RM: Positive values are truncated and negative values are rounded up (round to minus)

Name

fpcheck, fpchecko - Detect a floating-point exception.

Format

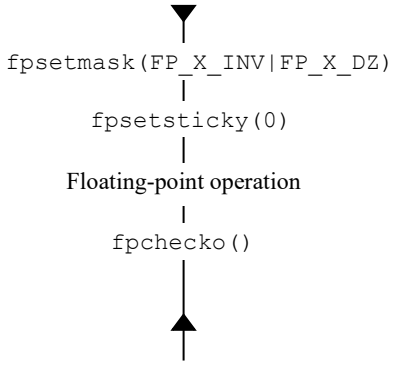
```
#include <ieeefp.h>

typedef enum {
               FP_RN =0, /*   Round to nearest           */
               FP_RZ =1  /*   Round to zero (truncate)   */
             } fp_rnd;

#define fp_except  int
#define FP_X_INV     0x10 /*  Invalid operation exception   */
#define FP_X_OFL     0x04 /*  Overflow exception            */
#define FP_X_UFL     0x02 /*  Underflow exception           */
#define FP_X_DZ      0x08 /*  Divide by zero exception      */
#define FP_X_IMP     0x01 /*  Imprecise (loss of precision) */

void         fpcheck(fp_except flg);
void         fpchecko(void);
```

Description

These macros detect the occurrence status of floating-point exceptions whose occurrence has been suppressed.
If the occurrence of a floating-point exception is detected, a program error occurs, and the task is aborted.
`fpchecko()` detects whether an overflow has occurred.
`fpcheck()` detects whether the exception specified by the parameter has occurred. To detect multiple exceptions simultaneously, specify the OR exception factor.

Example

The following is an example of when the occurrence of an overflow is suppressed and then detected after operations are performed.



... Only invalid operations and division by zero are enabled (overflow is disabled).

... The factor flag bit is cleared.

... A check is performed to see whether an overflow has occurred.
If an overflow has occurred, a program error occurs.

APPENDIXES

APPENDIX A MACRO PARAMETERS

(1/2)

Macro name	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5	Parameter 6	Parameter 7
rleas	tn						
queue	tn	fact					
exit							
abort	tn						
wait	ecb						
post	ecb	pcode					
susp	tn						
rsum	tn						
asusp							
arsum							
chap	tn	chgp					
sfact	tn	fact					
gfact	fact						
wrtmem	vaddr	data	size				
chkbmem	slot						
chktaer	slot						
timer	id	tn	fact	t	cyt		
ctime	tn	fact					
delay	t						
stime	time						
gtime	time						
rserv	n	para1	para2	para3	para4	para5	para6
free	n	para1	para2	para3	para4	para5	para6
prsrv	n	para1	para2	para3	para4	para5	
pfree	n	para1	para2	para3	para4	para5	
wdtset	msec						
getsysinfo	type	addr					
gettaskinfo	type	tn	addr				
gtkmem	tblno	caseno	offset	size	buf		
usrdhp	code	data	ndata				
usrel	type	class	retcode	errtype	erb		
save_env	env						
resume_env	env	val					
gettimebase	timebase						
TimebaseToSecs	timebase	tval					

(2/2)

Macro name	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5	Parameter 6	Parameter 7
atmswap	addr	data					
atmand	addr	data					
atmor	addr	data					
atmxor	addr	data					
atmadd	addr	data					
atmtas	addr	data					
atmcas	addr	data1	data2				
prog_start	start_addr	initial_usp	main_usp				
prog_switch	save_usp	resume_usp					
prog_exit	main_usp						
prog_call	entry_addr	initial_usp	para_cnt	para1	para2	para3	para4
MRAMmemcpy	dest	src	count				

APPENDIX B DIFFERENCES IN MACRO SPECIFICATIONS FROM S10V CMU

The following table lists the differences in the macro specifications from the previous model, S10V CMU.

(1/2)

Macro name (S10V)	Difference in specifications (*)	Details	Reference
rleas	No change	–	–
queue	No change	–	–
exit	No change	–	–
abort	No change	–	–
wait	No change	–	–
post	No change	–	–
susp	No change	–	–
rsum	No change	–	–
asusp	No change	–	–
arsum	No change	–	–
chap	No change	–	–
sfact	No change	–	–
gfact	No change	–	–
wrtmem	Functionally enhanced	Access area expansion (OPTPRM)	2-20
chkbmem	No change	–	–
chktaer	No change	–	–
mvmem	No longer supported	–	–
uspchk	No longer supported	–	–
timer	No change	–	–
ctime	No change	–	–
delay	No change	–	–
stime	Parameters changed	The fourth parameter, week, is no longer supported.	2-28
gtime	Parameters changed	The fourth parameter week, is no longer supported.	2-30
wake	No more supported	–	–
cwake	No more supported	–	–
rserv	No change	–	–
prsrv	No change	–	–
free	No change	–	–
pfree	No change	–	–
wdtset	Specifications changed	The ERR LED ON conditions have been changed.	2-39
getsysinfo	No change	–	–
gettaskinfo	No change	–	–
gtkmem	No change	–	–
usrdhp	No change	–	–
usrel	No change	–	–
save env	No change	–	–
resume_env	No change	–	–

(2/2)

Macro name (S10V)	Difference in specifications (*)	Details	Reference
gettimebase	No change	–	–
TimebaseToSecs	No change	–	–
atmswap	No change	–	–
atmand	No change	–	–
atmor	No change	–	–
atomxor	No change	–	–
atmtas	No change	–	–
atmcas	No change	–	–

(*) Functional enhancement: This macro is backward-compatible with the old macro. You can use S10V applications for this macro unchanged.

No longer supported: This macro is no longer supported.
Check and modify applications as necessary; for example, to use a different macro.

Parameters changed: The usage of the macro has changed.
See the description in the macro specifications, and modify applications accordingly.

Specifications changed: Return codes, parameter checks, and other details have changed.
See the description in the macro specifications, and modify applications accordingly.

APPENDIX C LIST OF ERROR MESSAGES

Table C-1 Error Messages (1/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
1	%CPMS-E-SOFT-0001	EC=03030000	Program error (Inst. Alignment Error)	Instruction alignment error	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
2	%CPMS-E-SOFT-0001	EC=03040000	Program error (Illegal Instruction)	Illegal instruction error	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
3	%CPMS-E-SOFT-0001	EC=03080000	Program error (Privileged Instruction)	Privileged instruction error	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
4	%CPMS-E-SOFT-0001	EC=03390000	Program error (FP Program Error)	Floating-point calculation error	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
5	%CPMS-E-SOFT-0001	EC=03400000	Program error (Instruction Page Fault)	Instruction access page fault	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
6	%CPMS-E-SOFT-0001	EC=03420000	Program error (Invalid Inst. Access)	Instruction access error	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
7	%CPMS-E-SOFT-0001	EC=03460000	Program error (Inst. Access Protection)	Instruction access protect error	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
8	%CPMS-E-SOFT-0001	EC=03470000	Program error (Data Alignment Error)	Data alignment error	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
9	%CPMS-E-SOFT-0001	EC=03600000	Program error (Data Page Fault)	Data access page fault	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
10	%CPMS-E-SOFT-0001	EC=03620000	Program error (Invalid Data Access)	Data access error	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
11	%CPMS-E-SOFT-0001	EC=03660000	Program error (Data Access Protection)	Data access protect error	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
12	%CPMS-E-SOFT-0001	EC=05C70005	Program error (Program WDT Timeout Error)	WDT timeout error	CPES, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
13	%CPMS-E-SOFT-0002	EC=05110000	Macro parameter error	Macro parameter error	PCKS, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
14	%CPMS-E-SOFT-0002	EC=05130000	Macro parameter error	Issuing of undefined-macro	PCKS, EAS	Software	TASK	TASK ABORT	–	–	Correct the program.
15	%LNET-x-xxxx-0004	EC=07xxxxxx	I/O error (Error detailed message)	Network I/O error	IES, EAS	Hardware	I/O	–	–	–	For more details, see Table C-3.
16	%CPMS-E-HARD-0004	EC=07395020	I/O error (ROM (NANDF) Error)	ROM (NAND-Flash) memory access failure	EAS	Hardware	ROM	–	–	–	Replace the hardware.

Table C-1 Error Messages (2/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
17	%CPMS-E-HARD-0004	EC=50010100	I/O error (OD.RING Module switch setting error)	OD.RING module switch setting error	IES, EAS	Hardware	OD.RING	–	–	ON	Check and correct the switch setting.
18	%CPMS-E-HARD-0004	EC=50010101	I/O error (OD.RING CPL switch setting error)	OD.RING CPL switch setting error	IES, EAS	Hardware	OD.RING	–	–	ON	Check and correct the switch setting.
19	%CPMS-E-HARD-0004	EC=50010111	I/O error (OD.RING Duplicate CPL No.)	OD.RING duplicate CPL no.	IES, EAS	Hardware	OD.RING	–	–	ON	Check and correct the switch setting.
20	%CPMS-E-HARD-0004	EC=50010112	I/O error (OD.RING Parameter type Mismatch/SUM err)	OD.RING parameter error (SUM value error)	IES, EAS	Hardware	OD.RING	–	–	ON	Reconfigure the parameters or replace the optional module.
21	%CPMS-E-HARD-0004	EC=5002010B	I/O error (FL.NET Parameter type Mismatch/SUM error)	FL.NET ROM3 sum error	IES, EAS	Hardware	FL.NET	–	–	ON	Reconfigure the parameters or replace the optional module.
22	%CPMS-E-HARD-0004	EC=50020113	I/O error (FL.NET IP address not registered)	FL.NET IP address not registered	IES, EAS	Hardware	FL.NET	–	–	ON	Reconfigure the parameters or replace the optional module.
23	%CPMS-E-HARD-0004	EC=50020200	I/O error (FL.NET NetWK participation not completed)	FL.NET not yet joined the network	IES, EAS	Hardware	FL.NET	–	–	–	Connect to the network.
24	%CPMS-E-HARD-0004	EC=50020201	I/O error (FL.NET Duplicate common memory settings)	FL.NET duplicate common memory setting	IES, EAS	Hardware	FL.NET	–	–	ON	Reconfigure the parameters or replace the optional module.
25	%CPMS-E-HARD-0004	EC=50020202	I/O error (FL.NET Duplicate node numbers)	FL.NET duplicate node numbers	IES, EAS	Hardware	FL.NET	–	–	ON	Reconfigure the parameters or replace the optional module.
26	%CPMS-E-HARD-0004	EC=50020203	I/O error (FL.NET FL.NET module setting error)	FL.NET FL.NET module setting error	IES, EAS	Hardware	FL.NET	–	–	ON	Reconfigure the parameters or replace the optional module.
27	%CPMS-E-HARD-0004	EC=50020204	I/O error (FL.NET Token hold timeout)	FL.NET token hold time timeout error	IES, EAS	Hardware	FL.NET	–	–	–	Check and correct the line load or replace the optional module.
28	%CPMS-E-HARD-0004	EC=50027310	I/O error (FL.NET I/O CARRIER LOSS)	FL.NET carrier loss error	IES, EAS	Hardware	FL.NET	–	–	–	Check and correct the line connection.
29	%CPMS-E-HARD-0004	EC=50027311	I/O error (FL.NET I/O RETRY)	FL.NET retry error	IES, EAS	Hardware	FL.NET	–	–	–	Check and correct the line connection.
30	%CPMS-E-HARD-0004	EC=50027312	I/O error (FL.NET I/O LATE)	FL.NET late collision error	IES, EAS	Hardware	FL.NET	–	–	–	Check and correct the line connection.
31	%CPMS-E-HARD-0004	EC=50027351	I/O error (FL.NET I/O TX_ABORT)	FL.NET transmission aborted	IES, EAS	Hardware	FL.NET	–	–	–	Check and correct the line connection.
32	%CPMS-E-HARD-0004	EC=50027353	I/O error (FL.NET I/O TX_DEFER)	FL.NET transmission error due to transmission delay	IES, EAS	Hardware	FL.NET	–	–	–	Check and correct the line connection.
33	%CPMS-E-HARD-0004	EC=50027375	I/O error (FL.NET I/O RX_STAT_OVER)	FL.NET reception status FIFO overrun	IES, EAS	Hardware	FL.NET	–	–	–	Check and correct the line load.
34	%CPMS-E-HARD-0004	EC=50027376	I/O error (FL.NET I/O TX_DATA_UNDER)	FL.NET reception data FIFO underrun	IES, EAS	Hardware	FL.NET	–	–	–	Check and correct the line load.
35	%CPMS-E-HARD-0004	EC=50027377	I/O error (FL.NET I/O RX_DATA_OVER)	FL.NET reception data FIFO overrun	IES, EAS	Hardware	FL.NET	–	–	–	Check and correct the line load.
36	%CPMS-E-HARD-0004	EC=50027508	I/O error (FL.NET I/O BUF_OVF)	FL.NET overflow of transmission/reception buffer managed by OS	IES, EAS	Hardware	FL.NET	–	–	–	Check and correct the line load.
37	%CPMS-E-HARD-0004	EC=5002750F	I/O error (FL.NET I/O SOCKET_OVF)	FL.NET socket table overflow	IES, EAS	Hardware	FL.NET	–	–	–	Replace the optional module.
38	%CPMS-E-HARD-0004	EC=50027512	I/O error (FL.NET I/O IPADDR_DUPL)	FL.NET duplicate IP address error	IES, EAS	Hardware	FL.NET	–	–	ON	Reconfigure the parameters or replace the optional module.
39	%CPMS-E-HARD-0004	EC=50027D10	I/O error (FL.NET INVALID MAIN/SUB SWITCH SETTING)	FL.NET wrong setting error in the MAIN/SUB setting switch	IES, EAS	Hardware	FL.NET	–	–	ON	Check and correct the switch setting.
40	%CPMS-E-HARD-0004	EC=50027D12	I/O error (FL.NET MAIN/SUB SW SETTING DUPLICATION)	FL.NET duplicate setting of the MAIN/SUB setting switch	IES, EAS	Hardware	FL.NET	–	–	ON	Check and correct the switch setting.

Table C-1 Error Messages (3/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
41	%CPMS-E-HARD-0004	EC=50030100	I/O error (J.NET Module switch setting error)	J.NET module switch setting error	IES, EAS	Hardware	J.NET	–	–	ON	Check and correct the switch setting.
42	%CPMS-E-HARD-0004	EC=50030101	I/O error (J.NET Baud rate switch setting error)	J.NET bit rate switch setting error	IES, EAS	Hardware	J.NET	–	–	ON	Check and correct the switch setting.
43	%CPMS-E-HARD-0004	EC=50030112	I/O error (J.NET Parameter type Mismatch/SUM error)	J.NET parameter error (SUM error)	IES, EAS	Hardware	J.NET			ON	Replace the optional module.
44	%CPMS-E-HARD-0004	EC=50032010	I/O error (J.NET CRC error)	J.NET CRC error	IES, EAS	Hardware	J.NET	–	–	–	Check and correct the line connection.
45	%CPMS-E-HARD-0004	EC=50032020	I/O error (J.NET Station No. error)	J.NET station number error	IES, EAS	Hardware	J.NET	–	–	–	Check and correct the line connection or parameters.
46	%CPMS-E-HARD-0004	EC=50032030	I/O error (J.NET Undefined service operated)	J.NET undefined service instruction	IES, EAS	Hardware	J.NET	–	–	–	Check and correct the line connection or parameters.
47	%CPMS-E-HARD-0004	EC=50032040	I/O error (J.NET I / UI-frame length error)	J.NET I-frame length/UI-frame length error	IES, EAS	Hardware	J.NET	–	–	–	Check and correct the line connection.
48	%CPMS-E-HARD-0004	EC=50032041	I/O error (J.NET I-frame format error(non Exist))	J.NET I response error (I-frames not included)	IES, EAS	Hardware	J.NET	–	–	–	Check and correct the line connection.
49	%CPMS-E-HARD-0004	EC=50032042	I/O error (J.NET I-frame format error(Exist))	J.NET supervisory frame error (I-frames included)	IES, EAS	Hardware	J.NET	–	–	–	Check and correct the line connection.
50	%CPMS-E-HARD-0004	EC=50032050	I/O error (J.NET Data link sequence error)	J.NET data link sequence error	IES, EAS	Hardware	J.NET	–	–	–	Check and correct the line connection.
51	%CPMS-E-HARD-0004	EC=50032060	I/O error (J.NET Slave response Timeout error)	J.NET timeout generated (no response from the slave)	IES, EAS	Hardware	J.NET	–	–	–	Check and correct the line connection or parameters.
52	%CPMS-E-HARD-0004	EC=50032061	I/O error (J.NET recover not successful)	J.NET recovery after retry failed	IES, EAS	Hardware	J.NET	–	–	–	Check and correct the line connection or parameters.
53	%CPMS-E-HARD-0004	EC=50032070	I/O error (J.NET Transmit/Receive error)	J.NET line frame transmission/reception error	IES, EAS	Hardware	J.NET	–	–	–	Check and correct the line connection or parameters.
54	%CPMS-E-HARD-0004	EC=50032080	I/O error (J.NET error occurred (.etc))	J.NET error generated (miscellaneous error)	IES, EAS	Hardware	J.NET	–	–	–	Replace the optional module.
55	%CPMS-E-HARD-0004	EC=50037061	I/O error (J.NET Waiting Input data)	J.NET input data pending	IES, EAS	Hardware	J.NET	–	–	–	–
56	%CPMS-E-HARD-0004	EC=50037110	I/O error (J.NET Undefined service operated)	J.NET undefined service instruction	IES, EAS	Hardware	J.NET	–	–	–	Replace the optional module.
57	%CPMS-E-HARD-0004	EC=50037120	I/O error (J.NET Transmission data length error)	J.NET data length error	IES, EAS	Hardware	J.NET	–	–	–	Replace the optional module.
58	%CPMS-E-HARD-0004	EC=50037130	I/O error (J.NET Transmission packet error)	J.NET packet structure error	IES, EAS	Hardware	J.NET	–	–	–	Replace the optional module.
59	%CPMS-E-HARD-0004	EC=50038020	I/O error (J.NET Initialize refused)	J.NET SVPT error (initialization command refused)	IES, EAS	Hardware	J.NET	–	–	–	Reconfigure the parameters or reconfigure the station.
60	%CPMS-E-HARD-0004	EC=50038081	I/O error (J.NET SVPT TX Bytes unmatched(Auto mode))	J.NET SVPT error (mismatch in transfer byte count in AUTO mode)	IES, EAS	Hardware	J.NET	–	–	–	Reconfigure the parameters or reconfigure the station.
61	%CPMS-E-HARD-0004	EC=50038082	I/O error (J.NET SVPT TX Bytes unmatched(Slot))	J.NET SVPT error (mismatch in transfer byte count when the slot is specified)	IES, EAS	Hardware	J.NET	–	–	–	Reconfigure the parameters or reconfigure the station.
62	%CPMS-E-HARD-0004	EC=50039001	I/O error (J.NET Station stopped)	J.NET station stopped	IES, EAS	Hardware	J.NET	–	–	–	Replace the station.
63	%CPMS-E-HARD-0004	EC=50039002	I/O error (J.NET Station error status detected)	J.NET station error	IES, EAS	Hardware	J.NET	–	–	–	Replace the station.
64	%CPMS-E-HARD-0004	EC=50039003	I/O error (J.NET St.err status detected and Stopped)	J.NET station error detected and stopped	IES, EAS	Hardware	J.NET	–	–	–	Replace the station.
65	%CPMS-E-HARD-0004	EC=5003A020	I/O error (J.NET PUT/GET(Insufficient address data))	J.NET PUT/GET error (insufficient address data)	IES, EAS	Hardware	J.NET	–	–	–	Replace the station.
66	%CPMS-E-HARD-0004	EC=5003A021	I/O error (J.NET PUT/GET(addr field number illegal))	J.NET PUT/GET error (address field number error)	IES, EAS	Hardware	J.NET	–	–	–	Replace the station.
67	%CPMS-E-HARD-0004	EC=5003A022	I/O error (J.NET PUT/GET(addr field format error))	J.NET PUT/GET error (numerical address field)	IES, EAS	Hardware	J.NET	–	–	–	Replace the station.
68	%CPMS-E-HARD-0004	EC=5003A040	I/O error (J.NET PUT/GET(Slot setting))	J.NET PUT/GET error (odd address)	IES, EAS	Hardware	J.NET	–	–	–	Replace the station.

Table C-1 Error Messages (4/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
69	%CPMS-E-HARD-0004	EC=5004140A	I/O error (D.NET Invalid MODU No. switch setting)	D.NET module switch setting error	IES, EAS	Hardware	D.NET	–	–	ON	Check and correct the switch setting.
70	%CPMS-E-HARD-0004	EC=50045188	I/O error (D.NET TX data size setting error)	D.NET transmission word count setting error	IES, EAS	Hardware	D.NET	–	–	ON	Reconfigure the parameters.
71	%CPMS-E-HARD-0004	EC=50045189	I/O error (D.NET Parameter type Mismatch/SUM error)	D.NET parameter error (SUM value error)	IES, EAS	Hardware	D.NET	–	–	ON	Reconfigure the parameters.
72	%CPMS-E-HARD-0004	EC=50047082	I/O error (D.NET Recover from Transmission Bus Off)	D.NET recovery from bus off	IES, EAS	Hardware	D.NET	–	–	–	–
73	%CPMS-E-HARD-0004	EC=50047381	I/O error (D.NET Transmission Bus Off)	D.NET transmission line bus off	IES, EAS	Hardware	D.NET	–	–	–	Check and correct the line connection.
74	%CPMS-E-HARD-0004	EC=50048181	I/O error (D.NET CAN Transmission Timeout Error.)	D.NET CAN transmission timeout error	IES, EAS	Hardware	D.NET	–	–	–	Check and correct the line connection.
75	%CPMS-E-HARD-0004	EC=500E7510	I/O error (ET.NET IFCONFIG_UP)	ET.NET network driver initialization error	IES, EAS	Hardware	ET.NET	–	–	ON	Check and correct the setting or replace the optional module.
76	%CPMS-E-HARD-0004	EC=500E7511	I/O error (ET.NET NETADDR_DUPL)	ET.NET duplicate network address error (system setup setting error)	IES, EAS	Hardware	ET.NET	–	–	ON	Check and correct the setting or replace the optional module.
77	%CPMS-E-HARD-0004	EC=500E7512	I/O error (ET.NET IPADDR_DUPL)	ET.NET duplicate IP address error (system setup setting error)	IES, EAS	Hardware	ET.NET	–	–	ON	Check and correct the IP address setting or replace the optional module.
78	%CPMS-E-HARD-0004	EC=500E7D12	I/O error (ET.NET Invalid MAIN/SUB switch setting Duplication)	ET.NET duplicate MAIN/SUB switch setting	IES, EAS	Hardware	ET.NET	–	–	ON	Check and correct the MAIN/SUB switch setting or replace the optional module.
79	%CPMS-E-HARD-0004	EC=500E7D1A	I/O error (ET.NET Invalid MAIN/SUB switch setting)	ET.NET MAIN/SUB switch setting error	IES, EAS	Hardware	ET.NET	–	–	ON	Check and correct the MAIN/SUB switch setting or replace the optional module.
80	%CPMS-E-HARD-0004	EC=500E7D1B	I/O error (ET.NET Invalid ST. No. switch setting)	ET.NET ST.no. switch setting error	IES, EAS	Hardware	ET.NET	–	–	ON	Check and correct the ST.no. switch setting or replace the optional module.
81	%CPMS-E-HARD-0004	EC=500E 7D1C	I/O error (ET.NET Invalid network setting)	ET.NET communication setting undefined	IES, EAS	Hardware	ET.NET	–	–	ON	Configure the communication settings or replace the optional module.
82	%CPMS-E-SOFT-0005	EC=05C70000	WDT timeout error	WDT timeout	WDTES, EAS	Software	TASK	–	ON	–	Correct the program
83	%CPMS-E-HARD-0006	EC=03820001	Module Error (Memory Error(MRAM))	Memory error (MRAM)	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
84	%CPMS-E-HARD-0006	EC=03B60000	Module Error (RI/O-IF Module Error)	RI/O-IF module error	MODES, EAS	Hardware	RI/O-IF	–	–	–	Replace the hardware.
85	%CPMS-E-HARD-0006	EC=03B90000	Module Error (PCI Bus Error)	PCI bus error	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
86	%CPMS-E-HARD-0006	EC=03BD0000	Module Error (LSI Internal Timeout Error)	LSI internal timeout error	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
87	%CPMS-E-HARD-0006	EC=03BE0000	Module Error (SPU Error)	SPU error	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
88	%CPMS-E-HARD-0006	EC=03BF0000	Module Error (RI/O Error)	RI/O error	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
89	%CPMS-E-SOFT-0006	EC=03E00000	Module Error (System task error(Table not found))	System task table address acquisition error	EAS	Software	CPMS	PU STOP	ON	–	Reload the CPMS.
90	%CPMS-E-SOFT-0006	EC=03E00001	Module Error (System task error(Task queue failed))	System task startup error	EAS	Software	CPMS	PU STOP	ON	–	Reload the CPMS.

Table C-1 Error Messages (5/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
91	%CPMS-E-HARD-0006	EC=05000000	Module Error (Invalid Interrupt)	Invalid interrupt	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
92	%CPMS-E-HARD-0006	EC=05000001	Module Error (Undefined Interrupt)	Undefined invalid interrupt	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
93	%CPMS-E-HARD-0006	EC=05000002	Module Error (INTEVT Invalid Interrupt)	INTEVT invalid interrupt	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
94	%CPMS-E-HARD-0006	EC=05001011	Module Error (RI/O INTR Invalid Interrupt)	RI/O invalid interrupt status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
95	%CPMS-E-HARD-0006	EC=05003001	Module Error (LV3 INTST Invalid Interrupt)	Level 3 invalid interrupt status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
96	%CPMS-E-HARD-0006	EC=05003002	Module Error (RQI6 INF Invalid Interrupt)	RQI6 invalid status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
97	%CPMS-E-HARD-0006	EC=05004001	Module Error (RINTR Invalid Interrupt)	RINT invalid status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
98	%CPMS-E-HARD-0006	EC=05006001	Module Error (SPU INTR Invalid Interrupt)	SPU invalid interrupt status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
99	%CPMS-E-HARD-0006	EC=0500A001	Module Error (NINTR Invalid Interrupt)	NINT invalid status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
100	%CPMS-E-HARD-0006	EC=0500B001	Module Error (PUINTR Invalid Interrupt)	PUINT invalid status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
101	%CPMS-E-HARD-0006	EC=0500F001	Module Error (HERST Invalid Interrupt)	Serious fault invalid interrupt	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
102	%CPMS-E-HARD-0006	EC=0500F002	Module Error (HERST Invalid Interrupt(2))	Serious fault invalid interrupt 2	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
103	%CPMS-E-HARD-0006	EC=0500F003	Module Error (BUERRSTAT Invalid Interrupt)	PCI bus error serious fault invalid interrupt status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
104	%CPMS-E-HARD-0006	EC=0500F004	Module Error (P2NHERREQ Invalid Interrupt)	CP to HP serious fault invalid interrupt status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
105	%CPMS-E-HARD-0006	EC=0500F005	Module Error (N2PHERREQ Invalid Interrupt)	HP to CP serious fault invalid interrupt status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
106	%CPMS-E-HARD-0006	EC=0500F006	Module Error (NHPMCLG Invalid Interrupt)	Memory serious fault invalid interrupt status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
107	%CPMS-E-HARD-0006	EC=0500F007	Module Error (ECC 2bit Master Invalid Interrupt)	Memory ECC 2 bit error serious fault invalid status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
108	%CPMS-E-HARD-0006	EC=0500F008	Module Error (RERRMST Invalid Interrupt)	PERR invalid interrupt status	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
109	%CPMS-E-HARD-0006	EC=0500F009	Module Error (Invalid P2NHERR Interrupt (CP Alive))	CP to HP serious fault invalid interrupt	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
110	%CPMS-E-HARD-0006	EC=0500F00B	Module Error (NP_ERRLOGMP Invalid Interrupt)	NPU serious fault invalid interrupt	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
111	%CPMS-E-HARD-0006	EC=0500F00C	Module Error (SPU HERR Invalid Interrupt)	SPU serious fault invalid interrupt	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
112	%CPMS-E-HARD-0006	EC=0500F00D	Module Error (RIO HERR Invalid Interrupt)	RI/O serious fault invalid interrupt	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
113	%CPMS-E-HARD-0006	EC=05110000	Module Error (Macro Parameter Error)	Failure to issue a macro to a PU other than itself	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
114	%CPMS-E-HARD-0006	EC=0739D001	Module Error (RQI6 Interrupt Received)	RQI6 interrupt generated	MODES, EAS	Hardware	Optional module	–	–	–	Replace the optional module.
115	%CPMS-E-HARD-0006	EC=0739D002	Module Error (RQI6 Interrupt Factor (ISW6) Clear Error)	RQI6 interrupt request clear error	MODES, EAS	Hardware	Optional module	–	–	–	Replace the optional module.
116	%CPMS-W-HARD-0006	EC=0D010000	Module Error (Memory Alarm)	Memory 1-bit error (solid)	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
117	%CPMS-E-HARD-0006	EC=0D010001	Module Error (Memory Patrol Error)	Memory patrol error	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
118	%CPMS-E-HARD-0006	EC=0D300010	Module Error (Primary Battery Error)	Primary battery error	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
119	%CPMS-E-HARD-0006	EC=0D320000	Module Error (Memory Error)	Memory error	MODES, EAS	Hardware	PU, I/O	–	–	–	Replace the hardware.
120	%CPMS-E-HARD-0006	EC=0D330000	Module Error (Hardware WDT timeout)	Hardware WDT timeout	MODES, EAS	Hardware	PU, I/O	–	–	–	Replace the hardware.
121	%CPMS-E-HARD-0006	EC=0D340000	Module Error (Software WDT Timeout)	Software WDT timeout	MODES, EAS	Hardware	PU, I/O	–	–	–	Replace the hardware or correct the program.
122	%CPMS-E-HARD-0006	EC=0D350000	Module Error (RAM Sum Check Error)	RAM checksum error	MODES, EAS	Hardware	PU, I/O	–	–	–	Replace the hardware or correct the program.

Table C-1 Error Messages (6/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
123	%CPMS-E-HARD-0006	EC=0D360000	Module Error (ROM Sum Check Error)	ROM checksum error	MODES, EAS	Hardware	PU, I/O	–	–	–	Replace the hardware.
124	%CPMS-E-HARD-0006	EC=0D370000	Module Error (External Error)	External error	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
125	%CPMS-W-HARD-0006	EC=0D380000	Module Error (OS Clear Error)	OS clear error	MODES, EAS	Hardware	PU, I/O	–	–	–	Load the program.
126	%CPMS-E-HARD-0006	EC=0D390000	Module Error (Clock Stop Error)	Clock stop error	MODES, EAS	Hardware	PU, I/O	–	–	–	Replace the hardware.
127	%CPMS-W-HARD-0006	EC=0D800000	Module Error (TOD Error)	Backup clock error	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
128	%CPMS-E-HARD-0006	EC=50010010	Module Error (OD.RING Bus error)	OD.RING bus error	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
129	%CPMS-E-HARD-0006	EC=50010011	Module Error (OD.RING Invalid address)	OD.RING address error	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
130	%CPMS-E-HARD-0006	EC=50010012	Module Error (OD.RING Invalid instruction)	OD.RING invalid instruction	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
131	%CPMS-E-HARD-0006	EC=50010013	Module Error (OD.RING Division by zero)	OD.RING division-by-zero	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
132	%CPMS-E-HARD-0006	EC=50010014	Module Error (OD.RING Privilege violation)	OD.RING privilege violation	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
133	%CPMS-E-HARD-0006	EC=50010015	Module Error (OD.RING WDT timeout error)	OD.RING WDT timeout error	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
134	%CPMS-E-HARD-0006	EC=50010016	Module Error (OD.RING Format error)	OD.RING format error	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
135	%CPMS-E-HARD-0006	EC=50010017	Module Error (OD.RING Spurious Interrupt)	OD.RING spurious interrupt	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
136	%CPMS-E-HARD-0006	EC=50010018	Module Error (OD.RING Unused exception)	OD.RING unused exception	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
137	%CPMS-E-HARD-0006	EC=50010019	Module Error (OD.RING Parity error)	OD.RING parity error	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
138	%CPMS-E-HARD-0006	EC=5001001A	Module Error (OD.RING Prepare for Grand Reset)	OD.RING GR notice	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
139	%CPMS-E-HARD-0006	EC=50010102	Module Error (OD.RING ROM1 checksum error)	OD.RING ROM1 checksum error	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
140	%CPMS-E-HARD-0006	EC=50010103	Module Error (OD.RING RAM1 compare error)	OD.RING RAM1 compare error	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
141	%CPMS-E-HARD-0006	EC=50010105	Module Error (OD.RING RAM2 compare error)	OD.RING RAM2 compare error	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
142	%CPMS-E-HARD-0006	EC=5001010B	Module Error (OD.RING ROM3 checksum error)	OD.RING ROM3 checksum error	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
143	%CPMS-E-HARD-0006	EC=5001010C	Module Error (OD.RING ROM erasing error (program))	OD.RING ROM3 erase error (program)	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
144	%CPMS-E-HARD-0006	EC=5001010D	Module Error (OD.RING ROM writing error (program))	OD.RING ROM3 write error (program)	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
145	%CPMS-E-HARD-0006	EC=5001010E	Module Error (OD.RING ROM erasing error (parameter))	OD.RING ROM3 erase error (parameter)	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
146	%CPMS-E-HARD-0006	EC=5001010F	Module Error (OD.RING ROM writing error (parameter))	OD.RING ROM3 write error (parameter)	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
147	%CPMS-E-HARD-0006	EC=50010110	Module Error (OD.RING ROM writing over 50000 times)	OD.RING ROM rewrite count limit exceeded	MODES, EAS	Hardware	OD.RING	–	–	ON	Replace the optional module.
148	%CPMS-E-HARD-0006	EC=50020114	Module Error (FL.NET MAC address not registered)	FL.NET MAC address not registered	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
149	%CPMS-E-HARD-0006	EC=50023031	Module Error (FL.NET Inst. Alignment Error)	FL.NET instruction alignment error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
150	%CPMS-E-HARD-0006	EC=50023041	Module Error (FL.NET Illegal Instruction)	FL.NET illegal instruction error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
151	%CPMS-E-HARD-0006	EC=50023081	Module Error (FL.NET Privileged Instruction)	FL.NET privileged instruction error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
152	%CPMS-E-HARD-0006	EC=500230F9	Module Error (FL.NET Illegal Exception)	FL.NET illegal exception error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
153	%CPMS-E-HARD-0006	EC=50023389	Module Error (FL.NET FP Unavailable)	FL.NET floating-point unavailable exception	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
154	%CPMS-E-HARD-0006	EC=50023391	Module Error (FL.NET FP Program Error)	FL.NET floating-point calculation error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
155	%CPMS-E-HARD-0006	EC=50023401	Module Error (FL.NET Instruction Page Fault)	FL.NET instruction access page fault	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.

Table C-1 Error Messages (7/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
156	%CPMS-E-HARD-0006	EC=50023421	Module Error (FL.NET Invalid Inst. Access)	FL.NET instruction access error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
157	%CPMS-E-HARD-0006	EC=50023461	Module Error (FL.NET Inst. Access Protection)	FL.NET instruction access protection error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
158	%CPMS-E-HARD-0006	EC=50023471	Module Error (FL.NET Data Alignment Error)	FL.NET data alignment error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
159	%CPMS-E-HARD-0006	EC=50023601	Module Error (FL.NET Data Page Fault)	FL.NET data access page fault	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
160	%CPMS-E-HARD-0006	EC=50023621	Module Error (FL.NET Invalid Data Access)	FL.NET data access error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
161	%CPMS-E-HARD-0006	EC=50023661	Module Error (FL.NET Data Access Protection)	FL.NET data access protection error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
162	%CPMS-E-HARD-0006	EC=50023820	Module Error (FL.NET Memory Error)	FL.NET memory error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
163	%CPMS-E-HARD-0006	EC=500238A0	Module Error (FL.NET Memory Access Error)	FL.NET memory access error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
164	%CPMS-E-HARD-0006	EC=500238B0	Module Error (FL.NET Internal Bus Parity)	FL.NET internal bus parity error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
165	%CPMS-E-HARD-0006	EC=500238C0	Module Error (FL.NET System Bus Parity)	FL.NET system bus parity error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
166	%CPMS-E-HARD-0006	EC=500238F0	Module Error (FL.NET Undefined Machine Check)	FL.NET undefined machine check error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
167	%CPMS-E-HARD-0006	EC=50023B70	Module Error (FL.NET Bus Target Abort)	FL.NET bus target abort	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
168	%CPMS-E-HARD-0006	EC=50025000	Module Error (FL.NET Invalid Interrupt)	FL.NET invalid interrupt	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
169	%CPMS-E-HARD-0006	EC=50025001	Module Error (FL.NET Undefined Invalid Interrupt)	FL.NET undefined invalid interrupt	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
170	%CPMS-E-HARD-0006	EC=50025002	Module Error (FL.NET INTEVT Invalid Interrupt)	FL.NET INTEVT invalid interrupt	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
171	%CPMS-E-HARD-0006	EC=50025011	Module Error (FL.NET RQI3 INT Invalid Interrupt)	FL.NET RQI3 invalid status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
172	%CPMS-E-HARD-0006	EC=50025012	Module Error (FL.NET RQI3 Link Invalid Interrupt)	FL.NET RQI3 invalid link status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
173	%CPMS-E-HARD-0006	EC=50025013	Module Error (FL.NET RQI3 Module Invalid Interrupt)	FL.NET RQI3 invalid module status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
174	%CPMS-E-HARD-0006	EC=50025031	Module Error (FL.NET LV3 INTST Invalid Interrupt)	FL.NET level 3 invalid interrupt status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
175	%CPMS-E-HARD-0006	EC=50025032	Module Error (FL.NET RQI6 INF Invalid Interrupt)	FL.NET RQI6 invalid status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
176	%CPMS-E-HARD-0006	EC=50025051	Module Error (FL.NET RINTR Invalid Interrupt)	FL.NET RINT invalid status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
177	%CPMS-E-HARD-0006	EC=500250B1	Module Error (FL.NET PUINTR Invalid Interrupt)	FL.NET PUINT invalid status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
178	%CPMS-E-HARD-0006	EC=500250C1	Module Error (FL.NET NINTR Invalid Interrupt)	FL.NET NINT invalid status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
179	%CPMS-E-HARD-0006	EC=500250F1	Module Error (FL.NET HERST Invalid Interrupt)	FL.NET serious fault invalid interrupt	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
180	%CPMS-E-HARD-0006	EC=500250F2	Module Error (FL.NET HERST2 Invalid Interrupt)	FL.NET serious fault invalid interrupt 2	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
181	%CPMS-E-HARD-0006	EC=500250F3	Module Error (FL.NET BUERRSTAT Invalid Interrupt)	FL.NET bus error serious fault invalid interrupt status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
182	%CPMS-E-HARD-0006	EC=500250F6	Module Error (FL.NET NHPMCLG Invalid Interrupt)	FL.NET memory serious fault invalid interrupt status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
183	%CPMS-E-HARD-0006	EC=500250F7	Module Error (FL.NET ECC 2bit Master Invalid Interrupt)	FL.NET memory ECC 2-bit error serious fault invalid status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
184	%CPMS-E-HARD-0006	EC=500250F8	Module Error (FL.NET RERRMST Invalid Interrupt)	FL.NET RERR invalid interrupt status	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
185	%CPMS-E-HARD-0006	EC=50025110	Module Error (FL.NET Macro parameter error)	FL.NET macro parameter error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
186	%CPMS-E-HARD-0006	EC=50025130	Module Error (FL.NET Undefined Macro)	FL.NET undefined macro issued	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
187	%CPMS-E-HARD-0006	EC=50025700	Module Error (FL.NET System Error)	FL.NET system down (system error)	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.

Table C-1 Error Messages (8/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
188	%CPMS-E-HARD-0006	EC=50025800	Module Error (FL.NET Kernel Trap)	FL.NET system down (kernel trap)	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
189	%CPMS-E-HARD-0006	EC=50025C70	Module Error (FL.NET WDT timeout error)	FL.NET WDT timeout	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
190	%CPMS-E-HARD-0006	EC=50027308	Module Error (FL.NET I/O SEND_TIMEOUT)	FL.NET transmission timeout error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
191	%CPMS-E-HARD-0006	EC=5002730A	Module Error (FL.NET I/O RESET_ERROR)	FL.NET hardware reset error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
192	%CPMS-E-HARD-0006	EC=5002730E	Module Error (FL.NET I/O MEMORY)	FL.NET memory error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
193	%CPMS-E-HARD-0006	EC=50027370	Module Error (FL.NET I/O EC_PCI_ERROR)	FL.NET PCI error detected by the communication LSI	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
194	%CPMS-E-HARD-0006	EC=50027400	Module Error (FL.NET I/O PCI_BUS_ERR)	FL.NET PCI bus error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
195	%CPMS-E-HARD-0006	EC=50027505	Module Error (FL.NET I/O INV_INTR)	FL.NET invalid interrupt generated from the line	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
196	%CPMS-E-HARD-0006	EC=50027510	Module Error (FL.NET I/O IFCONFIG_UP)	FL.NET network interface initialization error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
197	%CPMS-E-HARD-0006	EC=50027D01	Module Error (FL.NET INVALID EXCEPTION)	FL.NET invalid exception generated	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
198	%CPMS-E-HARD-0006	EC=50027D13	Module Error (FL.NET ETHERNET LSI CHECK ERROR)	FL.NET LANCE diagnosis error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
199	%CPMS-E-HARD-0006	EC=50027D14	Module Error (FL.NET SDRAM CHECK ERROR)	FL.NET SDRAM initialization error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
200	%CPMS-E-HARD-0006	EC=50027D15	Module Error (FL.NET OS-ROM CHECKSUM ERROR)	FL.NET ROM checksum error (CPMS)	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
201	%CPMS-E-HARD-0006	EC=50027D18	Module Error (FL.NET TASK-ROM CHECKSUM ERROR)	FL.NET ROM checksum error (communication task)	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
202	%CPMS-E-HARD-0006	EC=5002D010	Module Error (FL.NET Memory Alarm)	FL.NET memory 1-bit error (solid)	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
203	%CPMS-E-HARD-0006	EC=5002D330	Module Error (FL.NET Hardware WDT timeout)	FL.NET hardware WDT timeout	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
204	%CPMS-E-HARD-0006	EC=5002D340	Module Error (FL.NET Software WDT Timeout)	FL.NET software WDT timeout	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
205	%CPMS-E-HARD-0006	EC=5002D810	Module Error (FL.NET BPU Error)	FL.NET BPU error	MODES, EAS	Hardware	FL.NET	–	–	ON	Replace the optional module.
206	%CPMS-E-HARD-0006	EC=50030010	Module Error (J.NET Bus error)	J.NET bus error	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
207	%CPMS-E-HARD-0006	EC=50030011	Module Error (J.NET Invalid address)	J.NET address error	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
208	%CPMS-E-HARD-0006	EC=50030012	Module Error (J.NET Invalid instruction)	J.NET invalid instruction	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
209	%CPMS-E-HARD-0006	EC=50030013	Module Error (J.NET Division by zero)	J.NET division-by-zero	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
210	%CPMS-E-HARD-0006	EC=50030014	Module Error (J.NET Privilege violation)	J.NET privilege violation	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
211	%CPMS-E-HARD-0006	EC=50030015	Module Error (J.NET WDT timeout error)	J.NET WDT timeout error	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
212	%CPMS-E-HARD-0006	EC=50030016	Module Error (J.NET Format error)	J.NET format error	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
213	%CPMS-E-HARD-0006	EC=50030017	Module Error (J.NET Spurious Interrupt)	J.NET spurious interrupt	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
214	%CPMS-E-HARD-0006	EC=50030018	Module Error (J.NET Unused exception)	J.NET unused exception	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
215	%CPMS-E-HARD-0006	EC=50030019	Module Error (J.NET Parity error)	J.NET parity error	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
216	%CPMS-E-HARD-0006	EC=50030102	Module Error (J.NET ROM1 checksum error)	J.NET ROM1 checksum error	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
217	%CPMS-E-HARD-0006	EC=50030103	Module Error (J.NET RAM1 compare error)	J.NET RAM1 compare error	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
218	%CPMS-E-HARD-0006	EC=50030105	Module Error (J.NET RAM2 compare error)	J.NET RAM2 compare error	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
219	%CPMS-E-HARD-0006	EC=50030107	Module Error (J.NET DMA1 send error)	J.NET DMA1 transfer error (transmission)	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
220	%CPMS-E-HARD-0006	EC=50030108	Module Error (J.NET DMA2 send error)	J.NET DMA2 transfer error (transmission)	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
221	%CPMS-E-HARD-0006	EC=50030109	Module Error (J.NET DMA1 receive error)	J.NET DMA1 transfer error (reception)	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
222	%CPMS-E-HARD-0006	EC=5003010A	Module Error (J.NET DMA2 receive error)	J.NET DMA2 transfer error (reception)	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.

Table C-1 Error Messages (9/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
223	%CPMS-E-HARD-0006	EC=5003010B	Module Error (J.NET ROM3 checksum error)	J.NET ROM3 checksum error	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
224	%CPMS-E-HARD-0006	EC=5003010C	Module Error (J.NET ROM erasing error (program))	J.NET ROM erase error (program)	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
225	%CPMS-E-HARD-0006	EC=5003010D	Module Error (J.NET ROM writing error (program))	J.NET ROM write error (program)	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
226	%CPMS-E-HARD-0006	EC=5003010E	Module Error (J.NET ROM erasing error (parameter))	J.NET ROM erase error (parameter)	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
227	%CPMS-E-HARD-0006	EC=5003010F	Module Error (J.NET ROM writing error (parameter))	J.NET ROM write error (parameter)	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
228	%CPMS-E-HARD-0006	EC=50030110	Module Error (J.NET ROM writing error (writing over)	J.NET ROM rewrite count limit exceeded	MODES, EAS	Hardware	J.NET	–	–	ON	Replace the optional module.
229	%CPMS-E-HARD-0006	EC=50041401	Module Error (D.NET MPU Register Compare Error)	D.NET MPU register compare error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
230	%CPMS-E-HARD-0006	EC=50041402	Module Error (D.NET MPU Operation Check Error)	D.NET MPU operation check error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
231	%CPMS-E-HARD-0006	EC=50041403	Module Error (D.NET CAN Register Compare Error)	D.NET CAN register compare check error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
232	%CPMS-E-HARD-0006	EC=50041405	Module Error (D.NET FROM Compare Check Error)	D.NET FROM compare check error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
233	%CPMS-E-HARD-0006	EC=50041406	Module Error (D.NET FROM Checksum Error (microprogram))	D.NET FROM checksum error (microprogram)	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
234	%CPMS-E-HARD-0006	EC=50041407	Module Error (D.NET SRAM Compare Check Error)	D.NET SRAM compare check error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
235	%CPMS-E-HARD-0006	EC=50041409	Module Error (D.NET MPU Built-in Timer Diagnosis Error)	D.NET MPU built-in timer diagnosis error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
236	%CPMS-E-HARD-0006	EC=5004140D	Module Error (D.NET FROM Checksum Error(parameter))	D.NET FROM checksum error (parameter)	MODES, EAS	Hardware	D.NET	–	–	ON	Reconfigure the parameters or replace the optional module.
237	%CPMS-E-HARD-0006	EC=50042403	Module Error (D.NET Parity Error)	D.NET parity error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
238	%CPMS-E-HARD-0006	EC=50042404	Module Error (D.NET Watch-Dog-Timer Timeout Error)	D.NET watchdog timeout error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
239	%CPMS-E-HARD-0006	EC=50043400	Module Error (D.NET Undefined interrupt)	D.NET undefined interrupt generated	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
240	%CPMS-E-HARD-0006	EC=50043404	Module Error (D.NET General Invalid Instruction)	D.NET general illegal instruction	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
241	%CPMS-E-HARD-0006	EC=50043406	Module Error (D.NET Slot Invalid Instruction)	D.NET slot illegal instruction	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
242	%CPMS-E-HARD-0006	EC=50043409	Module Error (D.NET Address Error)	D.NET address error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
243	%CPMS-E-HARD-0006	EC=50044181	Module Error (D.NET Duplicated MAC ID(Other-Node Stop))	D.NET MAC ID duplicate (other node stopped)	MODES, EAS	Hardware	D.NET	–	–	ON	Check and correct the settings.
244	%CPMS-E-HARD-0006	EC=50044281	Module Error (D.NET Duplicated MAC ID(Self-Node Stop))	D.NET MAC ID duplicate (local node stopped)	MODES, EAS	Hardware	D.NET	–	–	ON	Check and correct the settings.
245	%CPMS-E-HARD-0006	EC=50044401	Module Error (D.NET I/O Transmission Stop Error (ch0))	D.NET CH0 I/O transmission stop error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
246	%CPMS-E-HARD-0006	EC=50044402	Module Error (D.NET I/O Transmission Stop Error (ch1))	D.NET CH1 I/O transmission stop error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
247	%CPMS-E-HARD-0006	EC=50049001	Module Error (D.NET T/M Error)	D.NET T/M error	MODES, EAS	Hardware	D.NET	–	–	ON	Replace the optional module.
248	%CPMS-E-HARD-0006	EC=500E 3031	Module error (ET.NET Inst. Alignment Error)	ET.NET instruction alignment error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
249	%CPMS-E-HARD-0006	EC=500E 3041	Module error (ET.NET Illegal Instruction)	ET.NET illegal instruction error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
250	%CPMS-E-HARD-0006	EC=500E 3081	Module error (ET.NET Privileged Instruction)	ET.NET privileged instruction error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
251	%CPMS-E-HARD-0006	EC=500E 30F9	Module error (ET.NET Illegal Exception)	ET.NET illegal exception error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
252	%CPMS-E-HARD-0006	EC=500E 3389	Module error (ET.NET FP Unavailable)	ET.NET floating-point unavailable exception	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
253	%CPMS-E-HARD-0006	EC=500E 3391	Module error (ET.NET FP Program Error)	ET.NET floating point operation error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
254	%CPMS-E-HARD-0006	EC=500E 3401	Module error (ET.NET Instruction Page Fault)	ET.NET instruction access page fault	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
255	%CPMS-E-HARD-0006	EC=500E 3421	Module error (ET.NET Invalid Inst. Access)	ET.NET instruction access error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
256	%CPMS-E-HARD-0006	EC=500E 3461	Module error (ET.NET Inst. Access Protection)	ET.NET instruction access protection error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
257	%CPMS-E-HARD-0006	EC=500E 3471	Module error (ET.NET Data Alignment Error)	ET.NET data alignment error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
258	%CPMS-E-HARD-0006	EC=500E 3601	Module error (ET.NET Data Page Fault)	ET.NET data access page fault	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
259	%CPMS-E-HARD-0006	EC=500E 3621	Module error (ET.NET Invalid Data Access)	ET.NET data access error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.

Table C-1 Error Messages (10/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
260	%CPMS-E-HARD-0006	EC=500E 3661	Module error (ET.NET Data Access Protection)	ET.NET data access protection error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
261	%CPMS-E-HARD-0006	EC=500E 3820	Module error (ET.NET Memory Error)	ET.NET memory error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
262	%CPMS-E-HARD-0006	EC=500E 3B70	Module error (ET.NET Bus Target Abort)	ET.NET bus target abort	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
263	%CPMS-E-HARD-0006	EC=500E 3B81	Module error (ET.NET System Bus Error CPU Master)	ET.NET system bus error (access from CPU)	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
264	%CPMS-E-HARD-0006	EC=500E 3B82	Module error (ET.NET System Bus Error CPU Target)	ET.NET system bus error (access to CPU)	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
265	%CPMS-E-HARD-0006	EC=500E 3B90	Module error (ET.NET PCI_BUS_ERR)	ET.NET PCI bus error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
266	%CPMS-E-HARD-0006	EC=500E 5001	Module error (ET.NET Undefined Invalid Interrupt)	ET.NET undefined invalid interrupt	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
267	%CPMS-E-HARD-0006	EC=500E 5002	Module error (ET.NET INTEVT Invalid Interrupt)	ET.NET INTEVT invalid interrupt	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
268	%CPMS-E-HARD-0006	EC=500E 50F1	Module error (ET.NET HERST Invalid Interrupt)	ET.NET serious fault invalid interrupt	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
269	%CPMS-E-HARD-0006	EC=500E 50F2	Module error (ET.NET HERST2 Invalid Interrupt)	ET.NET serious fault invalid interrupt 2	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
270	%CPMS-E-HARD-0006	EC=500E 50F3	Module error (ET.NET BUERRSTAT Invalid Interrupt)	ET.NET bus error serious fault invalid interrupt status	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
271	%CPMS-E-HARD-0006	EC=500E 50F6	Module error (ET.NET NHPMCLG Invalid Interrupt)	ET.NET memory serious fault invalid interrupt status	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
272	%CPMS-E-HARD-0006	EC=500E 50F7	Module error (ET.NET ECC 2bit Master Invalid Interrupt)	ET.NET memory ECC 2-bit error serious fault invalid interrupt status	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
273	%CPMS-E-HARD-0006	EC=500E 50F8	Module error (ET.NET RERRMST Invalid Interrupt)	ET.NET RERR invalid interrupt status	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
274	%CPMS-E-HARD-0006	EC=500E 5110	Module error (ET.NET Macro parameter error)	ET.NET macro parameter error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
275	%CPMS-E-HARD-0006	EC=500E 5130	Module error (ET.NET Undefined Macro error)	ET.NET undefined macro issued	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
276	%CPMS-E-HARD-0006	EC=500E 5700	Module error (ET.NET System Error)	ET.NET system down (system error)	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
277	%CPMS-E-HARD-0006	EC=500E 5800	Module error (ET.NET Kernel Trap)	ET.NET system down (kernel trap)	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
278	%CPMS-E-HARD-0006	EC=500E 5C70	Module error (ET.NET WDT timeout error)	ET.NET watchdog timeout	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
279	%CPMS-E-HARD-0006	EC=500E 7308	Module error (ET.NET SEND_TIMEOUT)	ET.NET transmission timeout error (*)	MODES, EAS	Hardware	ET.NET	–	–	ON	Turn the module off and then on again, or replace the optional module.
280	%CPMS-E-HARD-0006	EC=500E 730A	Module error (ET.NET RESET_ERROR)	ET.NET hard reset error	MODES, EAS	Hardware	ET.NET	–	–	ON	Turn the module off and then on again, or replace the optional module.
281	%CPMS-E-HARD-0006	EC=500E 7505	Module error (ET.NET INV_INTR)	ET.NET invalid interrupt generated from the line	MODES, EAS	Hardware	ET.NET	–	–	ON	Turn the module off and then on again, or replace the optional module.
282	%CPMS-E-HARD-0006	EC=500E 7D01	Module error (ET.NET INVALID EXCEPTION)	ET.NET invalid exception generated	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
283	%CPMS-E-HARD-0006	EC=500E 7D11	Module error (ET.NET Invalid MAC ADDRESS)	ET.NET MAC address error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
284	%CPMS-E-HARD-0006	EC=500E 7D13	Module error (ET.NET ETHERNET LSI CHECK ERROR)	ET.NET LANCE diagnosis error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
285	%CPMS-E-HARD-0006	EC=500E 7D14	Module error (ET.NET SDRAM CHECK ERROR)	ET.NET SDRAM initialization error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
286	%CPMS-E-HARD-0006	EC=500E 7D18	Module error (ET.NET ROM CHECKSUM ERROR)	ET.NET ROM checksum error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
287	%CPMS-E-HARD-0006	EC=500E D010	Module error (ET.NET Memory Alarm)	ET.NET memory 1-bit error (solid)	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.
288	%CPMS-E-HARD-0006	EC=500E D810	Module error (ET.NET BPU Error)	ET.NET BPU error	MODES, EAS	Hardware	ET.NET	–	–	ON	Replace the optional module.

(*) The system outputs this message once when it detects this error 5 times in a row.

Table C-1 Error Messages (11/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
289	%CPMS-E-HARD-0006	EC=51000000	Module Error (Optional Module startup check error)	Optional module startup check error	MODES, EAS	Hardware	Optional module	–	–	–	Replace the optional module.
290	%CPMS-E-HARD-0006	EC=51000001	Module Error (System Register Clear Time Out)	System register initialization timeout error	MODES, EAS	Hardware	PU	–	–	–	Replace the hardware.
291	%CPMS-E-HARD-0006	EC=51000002	Module Error (Optional Paramater size Error)	Optional module parameter size error	MODES, EAS	Hardware	PU	–	–	–	Reconfigure the parameters or replace the optional module.
292	%CPMS-W-SOFT-0007	EC=05A00000	Kernel warning	Kernel warning	EAS	Software	–	–	–	–	–
293	%CPMS-W-SOFT-0007	EC=05A00001	Kernel warning	Clock synchronization (>15sec)	EAS	Hardware	–	–	–	–	–
294	%CPMS-W-SOFT-0007	EC=05A00006	Kernel warning	PU macro error	EAS	Software	–	–	–	–	–
295	%CPMS-I-SOFT-0008	EC=05D00000	Kernel information	Kernel information	EAS	Software	–	–	–	–	–
296	%CPMS-F-CPMS-0009	EC=03030000	System down (Inst. Alignment Error)	Instruction alignment error	–	Software	CPMS	PU STOP	ON	–	Correct the program.
297	%CPMS-F-CPMS-0009	EC=03040000	System down (Illegal Instruction)	Illegal instruction error	–	Software	CPMS	PU STOP	ON	–	Correct the program.
298	%CPMS-F-CPMS-0009	EC=030F0000	System down (Illegal Exception)	Illegal exception error	–	Software	CPMS	PU STOP	ON	–	Correct the program.
299	%CPMS-F-CPMS-0009	EC=03380000	System down (FP Unavailable)	Floating-point unavailable exception	–	Software	CPMS	PU STOP	ON	–	Correct the program.
300	%CPMS-F-CPMS-0009	EC=03390000	System down (FP System down)	Floating-point calculation error	–	Software	CPMS	PU STOP	ON	–	Correct the program.
301	%CPMS-F-CPMS-0009	EC=03400000	System down (Instruction Page Fault)	Instruction access page fault	–	Software	CPMS	PU STOP	ON	–	Correct the program.
302	%CPMS-F-CPMS-0009	EC=03470000	System down (Data Alignment Error)	Data alignment error	–	Software	CPMS	PU STOP	ON	–	Correct the program.
303	%CPMS-F-CPMS-0009	EC=03600000	System down (Data Page Fault)	Data access page fault	–	Software	CPMS	PU STOP	ON	–	Correct the program.
304	%CPMS-F-CPMS-0009	EC=03660000	System down (Data Access Protection)	Data access protect error	–	Software	CPMS	PU STOP	ON	–	Correct the program.
305	%CPMS-F-CPMS-0009	EC=03820000	System down (Memory Error)	Memory error	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
306	%CPMS-F-CPMS-0009	EC=03820001	System down (Memory Error(MRAM))	Memory error (MRAM)	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
307	%CPMS-F-CPMS-0009	EC=03B60000	System down (RI/O-IF Module Error)	RI/O-IF module error	–	Hardware	RI/O-IF	PU STOP	ON	–	Replace the hardware.
308	%CPMS-F-CPMS-0009	EC=03B80000	System down (R700/S10 Bus Error)	System bus error serious fault invalid interrupt status (ten times in a row)	–	Hardware	I/O, PU	PU STOP	ON	–	Replace the hardware.
309	%CPMS-F-CPMS-0009	EC=03B80001	System down (CPU Master)	CPU master access system bus error	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
310	%CPMS-F-CPMS-0009	EC=03B90000	System down (PCI Bus Error)	PCI bus error	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
311	%CPMS-F-CPMS-0009	EC=03BD0000	System down (LSI Internal Timeout Error)	LSI internal timeout error	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
312	%CPMS-F-CPMS-0009	EC=03BE0000	System down (SPU Error)	SPU error	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
313	%CPMS-F-CPMS-0009	EC=03BF0000	System down (RI/O Error)	RI/O error	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
314	%CPMS-F-CPMS-0009	EC=0500F001	System down (HERST Invalid Interrupt)	Serious fault invalid interrupt status (ten times in a row)	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
315	%CPMS-F-CPMS-0009	EC=0500F003	System down (BUERRSTAT Invalid Interrupt)	PCI bus error serious fault invalid interrupt status (twice in a row)	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.

Table C-1 Error Messages (12/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
316	%CPMS-F-CPMS-0009	EC=0500F004	System down (P2NHERRQ Invalid Interrupt)	CP to HP serious fault invalid interrupt status (twice in a row)	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
317	%CPMS-F-CPMS-0009	EC=0500F005	System down (N2PHERRQ Invalid Interrupt)	HP to CP serious fault invalid interrupt status (twice in a row)	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
318	%CPMS-F-CPMS-0009	EC=0500F00B	System down (NP_ERRLOGMP Invalid Interrupt)	NPU serious fault invalid interrupt status (twice in a row)	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
319	%CPMS-F-CPMS-0009	EC=05700000	System down (System Error)	System down (system error)	–	Software	CPMS	PU STOP	ON	–	Correct the program.
320	%CPMS-F-CPMS-0009	EC=05700001	System down (CP Infinit loop Detect)	CP-side OS infinite loop detected	–	Software	CPMS	PU STOP	ON	–	Correct the program.
321	%CPMS-F-CPMS-0009	EC=05700002	System down (HP Infinit loop Detect)	HP-side OS infinite loop detected	–	Software	CPMS	PU STOP	ON	–	Correct the program.
322	%CPMS-F-CPMS-0009	EC=05900000	System down (CP Down)	HP down because CP down	–	Software Hardware	CPMS Hardware	PU STOP	ON	–	Correct the program. Replace the hardware.
323	%CPMS-F-CPMS-0009	EC=05900001	System down (HP Down)	CP down because HP down	–	Software Hardware	CPMS Hardware	PU STOP	ON	–	Correct the program. Replace the hardware.
324	%CPMS-F-CPMS-0009	EC=0D010001	System down (Memory Patrol Error)	Memory patrol error	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
325	%CPMS-F-CPMS-0009	EC=0D810000	System down (BPU Error)	BPU error	–	Hardware	PU	PU STOP	ON	–	Replace the hardware.
326	%CPMS-F-CPMS-000A	EC=05800000	System down (Kernel Trap)	System down (kernel trap)	–	Software	CPMS	PU STOP	ON	–	Correct the program.
327	%CPMS-F-SOFT-000B	EC=03030000	ULSUB down (Inst. Alignment Error)	Instruction alignment error	–	Software	ULSUB	PU STOP	ON	–	Correct the program.
328	%CPMS-F-SOFT-000B	EC=03040000	ULSUB down (Illegal Instruction)	Illegal instruction error	–	Software	ULSUB	PU STOP	ON	–	Correct the program.
329	%CPMS-F-SOFT-000B	EC=030F0000	ULSUB down (Illegal Exception)	Illegal exception error	–	Software	ULSUB	PU STOP	ON	–	Correct the program.
330	%CPMS-F-SOFT-000B	EC=03380000	ULSUB down (FP Unavailable)	Floating-point unavailable exception	–	Software	ULSUB	PU STOP	ON	–	Correct the program.
331	%CPMS-F-SOFT-000B	EC=03390000	ULSUB down (FP System down)	Floating-point calculation error	–	Software	ULSUB	PU STOP	ON	–	Correct the program.
332	%CPMS-F-SOFT-000B	EC=03400000	ULSUB down (Instruction Page Fault)	Instruction access page fault	–	Software	ULSUB	PU STOP	ON	–	Correct the program.
333	%CPMS-F-SOFT-000B	EC=03470000	ULSUB down (Data Alignment Error)	Data alignment error	–	Software	ULSUB	PU STOP	ON	–	Correct the program.
334	%CPMS-F-SOFT-000B	EC=03600000	ULSUB down (Data Page Fault)	Data access page fault	–	Software	ULSUB	PU STOP	ON	–	Correct the program.
335	%CPMS-F-SOFT-000B	EC=03660000	ULSUB down (Data Access Protection)	Data access protection error	–	Software	ULSUB	PU STOP	ON	–	Correct the program.
336	%CPMS-F-SOFT-000C	EC=05140000	System down (ULSUB Stop)	System down (built-in sub-stop)	–	Software	ULSUB	PU STOP	ON	–	–

Table C-1 Error Messages (13/13)

No.	Error log title	Error code	Error message	Description	Built-in subroutine	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
337	%CPMS-E-SOFT-0010	EC=03D00001	Ladder Program error (Data Access Protection)	Data access protect error	EAS	Software	LADDER	–	–	–	Check and correct the LADDER program.
338	%CPMS-E-SOFT-0010	EC=03D00002	Ladder Program error (Stack Overflow)	Stack overflow error	EAS	Software	LADDER	–	–	–	Check and correct the LADDER program.
339	%CPMS-E-SOFT-0010	EC=03D00003	Ladder Program error (Illegal Instruction)	Illegal instruction error	EAS	Software	LADDER	–	–	–	Check and correct the LADDER program.
340	%CPMS-E-SOFT-0010	EC=03D00004	Ladder Program error (FP Program Error)	Floating-point calculation error	EAS	Software	LADDER	–	–	–	Check and correct the LADDER program.
341	%CPMS-E-SOFT-0010	EC=03D00005	Ladder Program error (Segment Address Overflow)	Segment address overflow error	EAS	Software	LADDER	–	–	–	Check and correct the LADDER program.
342	%CPMS-E-SOFT-0010	EC=03D00006	Ladder Program error (Illegal SH Instruction)	SH illegal instruction error	EAS	Software	LADDER	–	–	–	Check and correct the LADDER program.
343	%CPMS-E-SOFT-0010	EC=03D01101	Ladder Program error (P-Coil CP DOWN Detect)	P coil CP down detected	EAS	Software	PU	–	–	–	–
344	%CPMS-E-SOFT-0010	EC=03D01208	Ladder Program error (N-Coil Nesting Over)	N coil overflow error	EAS	Software	LADDER	–	–	–	Check and correct the LADDER program.
345	%CPMS-E-SOFT-0010	EC=03D0120A	Ladder Program error (Illegal User Function)	User operation function address error	EAS	Software	LADDER	–	–	–	Check and correct the LADDER program.
346	%CPMS-E-SOFT-0010	EC=03D0120C	Ladder Program error (Illegal Function Parameter)	System operation function parameter error	EAS	Software	LADDER	–	–	–	Check and correct the LADDER program.
347	%CPMS-E-SOFT-0010	EC=03D01210	Ladder Program error (Ladder Area Sum Mismatch)	Ladder area SUM error	EAS	Software	LADDER	–	–	–	Reload the LADDER program.
348	%CPMS-E-SOFT-0010	EC=03D01212	Ladder Program error (Ladder Table Empty)	Table not registered	EAS	Software	TASK	–	–	–	–
349	%CPMS-E-SOFT-0010	EC=03D01214	Ladder Program error (Illegal Factor)	Initiation factor error	EAS	Software	TASK	–	–	–	–
350	%CPMS-E-HARD-0012	EC=03820000	Memory Error	Memory-related serious fault detected	EAS	Hardware	PU	PU STOP	ON	–	Replace the hardware.
351	%CPMS-E-HARD-0013	EC=03B70000	System Bus Error (Master/Target Abort)	Bus target abort	MODES, EAS	Hardware	I/O	–	–	–	Replace the hardware.
352	%CPMS-E-HARD-0013	EC=03B70001	System Bus Error (S10 Bus DTACK Timeout)	Timeout detected when the CPU accessed the S10 bus	MODES, EAS	Hardware	Optional module	–	–	–	If an optional module is not mounted, mount the module. If mounted, replace the module.
353	%CPMS-E-HARD-0013	EC=03B80001	System Bus Error (CPU Master)	CPU master access system bus error	EAS	Hardware	PU	PU STOP	ON	–	Replace the hardware.
354	%CPMS-E-HARD-0013	EC=03B80002	System Bus Error (CPU Target)	CPU target access system bus error	EAS	Hardware	I/O	–	–	–	Replace the hardware.

Table C-2 Error Messages (ET.NET) (1/2)

No.	Error log title	Error code	Error message	Description	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
1	%CPMS-E-SOFT-0001	EC=03030000	Program error (Inst. Alignment Error)	Instruction alignment error	Hardware	ET.NET	–	–	ON	Replace the optional module.
2	%CPMS-E-SOFT-0001	EC=03040000	Program error (Illegal Instruction)	Illegal instruction error	Hardware	ET.NET	–	–	ON	Replace the optional module.
3	%CPMS-E-SOFT-0001	EC=03080000	Program error (Privileged Instruction)	Privileged instruction error	Hardware	ET.NET	–	–	ON	Replace the optional module.
4	%CPMS-E-SOFT-0001	EC=03390000	Program error (FP Program Error)	Floating-point calculation error	Hardware	ET.NET	–	–	ON	Replace the optional module.
5	%CPMS-E-SOFT-0001	EC=03400000	Program error (Instruction Page Fault)	Instruction access page fault	Hardware	ET.NET	–	–	ON	Replace the optional module.
6	%CPMS-E-SOFT-0001	EC=03420000	Program error (Invalid Inst. Access)	Instruction access error	Hardware	ET.NET	–	–	ON	Replace the optional module.
7	%CPMS-E-SOFT-0001	EC=03460000	Program error (Inst. Access Protection)	Instruction access protect error	Hardware	ET.NET	–	–	ON	Replace the optional module.
8	%CPMS-E-SOFT-0001	EC=03470000	Program error (Data Alignment Error)	Data alignment error	Hardware	ET.NET	–	–	ON	Replace the optional module.
9	%CPMS-E-SOFT-0001	EC=03600000	Program error (Data Page Fault)	Data access page fault	Hardware	ET.NET	–	–	ON	Replace the optional module.
10	%CPMS-E-SOFT-0001	EC=03620000	Program error (Invalid Data Access)	Data access error	Hardware	ET.NET	–	–	ON	Replace the optional module.
11	%CPMS-E-SOFT-0001	EC=03660000	Program error (Data Access Protection)	Data access protect error	Hardware	ET.NET	–	–	ON	Replace the optional module.
12	%CPMS-E-SOFT-0002	EC=05110000	Macro parameter error	Macro parameter error	Hardware	ET.NET	–	–	ON	Replace the optional module.
13	%CPMS-E-SOFT-0002	EC=05130000	Invalid Macro	Issuing of undefined-macro	Hardware	ET.NET	–	–	ON	Replace the optional module.
14	%LNET-x-xxxx-0004	EC=07xxxxxx	I/O error (error detailed message)	Network I/O error	Hardware	I/O	–	–	–	For more details, see Table C-4.
15	%CPMS-E-SOFT-0005	EC=05C70000	WDT timeout error	WDT timeout	Hardware	ET.NET	–	–	ON	Replace the optional module.
16	%CPMS-E-HARD-0006	EC=03B90000	Module Error (PCI Bus Error)	PCI bus error	Hardware	ET.NET	–	–	ON	Replace the optional module.
17	%CPMS-E-HARD-0006	EC=05000001	Module Error (Undefined Interrupt)	Undefined invalid interrupt	Hardware	ET.NET	–	–	ON	Replace the optional module.
18	%CPMS-E-HARD-0006	EC=05000002	Module Error (INTEVT Invalid Interrupt)	INTEVT invalid interrupt	Hardware	ET.NET	–	–	ON	Replace the optional module.
19	%CPMS-E-HARD-0006	EC=0500F001	Module Error (HERST Invalid Interrupt)	Serious fault invalid interrupt	Hardware	ET.NET	–	–	ON	Replace the optional module.
20	%CPMS-E-HARD-0006	EC=0500F002	Module Error (HERST Invalid Interrupt(2))	Serious fault invalid interrupt 2	Hardware	ET.NET	–	–	ON	Replace the optional module.
21	%CPMS-E-HARD-0006	EC=0500F003	Module Error (BUERRSTAT Invalid Interrupt)	PCI bus error serious fault invalid interrupt status	Hardware	ET.NET	–	–	ON	Replace the optional module.
22	%CPMS-E-HARD-0006	EC=0500F006	Module Error (MHPMCLG Invalid Interrupt)	Memory serious fault invalid interrupt status	Hardware	ET.NET	–	–	ON	Replace the optional module.
23	%CPMS-E-HARD-0006	EC=0500F007	Module Error (ECC 2bit Master Invalid Interrupt)	Memory ECC 2-bit error serious fault invalid status	Hardware	ET.NET	–	–	ON	Replace the optional module.
24	%CPMS-E-HARD-0006	EC=0500F008	Module Error (RERRMST Invalid Interrupt)	RERR interrupt invalid status	Hardware	ET.NET	–	–	ON	Replace the optional module.
25	%CPMS-E-HARD-0006	EC=0500xxxx	Module Error (Invalid Interrupt)	Invalid interrupt	Hardware	ET.NET	–	–	ON	Replace the optional module.
26	%CPMS-E-HARD-0006	EC=05110000	Module Error (Macro Parameter Error)	Failure in issuing a macro to a PU other than itself	Hardware	ET.NET	–	–	ON	Replace the optional module.
27	%CPMS-W-HARD-0006	EC=0D010000	Module Error (Memory Alarm)	Memory 1-bit error (solid)	Hardware	ET.NET	–	–	ON	Replace the optional module.

Table C-2 Error Messages (ET.NET) (2/2)

No.	Error log title	Error code	Error message	Description	Fault category	Fault location	Abortion	ERR LED	ALM LED	Recovery measure
28	%CPMS-F-CPMS-0009	EC=03030000	System down (Inst. Alignment Error)	Instruction alignment error	Hardware	ET.NET	–	–	ON	Replace the optional module.
29	%CPMS-F-CPMS-0009	EC=03040000	System down (Illegal Instruction)	Illegal instruction error	Hardware	ET.NET	–	–	ON	Replace the optional module.
30	%CPMS-F-CPMS-0009	EC=030F0000	System down (Illegal Exception)	Illegal exception error	Hardware	ET.NET	–	–	ON	Replace the optional module.
31	%CPMS-F-CPMS-0009	EC=03380000	System down (FP Unavailable)	Floating-point unavailable exception	Hardware	ET.NET	–	–	ON	Replace the optional module.
32	%CPMS-F-CPMS-0009	EC=03390000	System down (FP System down)	Floating-point calculation error	Hardware	ET.NET	–	–	ON	Replace the optional module.
33	%CPMS-F-CPMS-0009	EC=03400000	System down (Instruction Page Fault)	Instruction access page fault	Hardware	ET.NET	–	–	ON	Replace the optional module.
34	%CPMS-F-CPMS-0009	EC=03470000	System down (Data Alignment Error)	Data alignment error	Hardware	ET.NET	–	–	ON	Replace the optional module.
35	%CPMS-F-CPMS-0009	EC=03600000	System down (Data Page Fault)	Data access page fault	Hardware	ET.NET	–	–	ON	Replace the optional module.
36	%CPMS-F-CPMS-0009	EC=03660000	System down (Data Access Protection)	Data access protect error	Hardware	ET.NET	–	–	ON	Replace the optional module.
37	%CPMS-F-CPMS-0009	EC=03820000	System down (Memory Error)	Memory error	Hardware	ET.NET	–	–	ON	Replace the optional module.
38	%CPMS-F-CPMS-0009	EC=03B80001	System down (CPU Master)	CPU master access system bus error	Hardware	ET.NET	–	–	ON	Replace the optional module.
39	%CPMS-F-CPMS-0009	EC=03B90000	System down (PCI Bus Error)	PCI bus error	Hardware	ET.NET	–	–	ON	Replace the optional module.
40	%CPMS-F-CPMS-0009	EC=0500F001	System down (HERST Invalid Interrupt)	Serious fault invalid interrupt	Hardware	ET.NET	–	–	ON	Replace the optional module.
41	%CPMS-F-CPMS-0009	EC=0500F003	System down (BUERRSTAT Invalid Interrupt)	PCI bus error serious fault interrupt status invalid	Hardware	ET.NET	–	–	ON	Replace the optional module.
42	%CPMS-F-CPMS-0009	EC=0500xxxx	System down (Invalid Interrupt)	Invalid interrupt	Hardware	ET.NET	–	–	ON	Replace the optional module.
43	%CPMS-F-CPMS-0009	EC=0570000x	System down (System Error)	System down (system error)	Hardware	ET.NET	–	–	ON	Replace the optional module.
44	%CPMS-F-CPMS-0009	EC=0D810000	System down (BPU Error)	BPU error	Hardware	ET.NET	–	–	ON	Replace the optional module.
45	%CPMS-F-CPMS-000A	EC=05800000	System down (Kernel Trap)	System down (kernel trap)	Hardware	ET.NET	–	–	ON	Replace the optional module.
46	%CPMS-F-SOFT-000C	EC=05140000	System down (ULSUB Stop)	System down (built-in sub-stop)	Hardware	ET.NET	–	–	ON	Replace the optional module.
47	%CPMS-E-HARD-000E	EC=03820000	Memory Error	Memory-related serious fault detected	Hardware	ET.NET	–	–	ON	Replace the optional module.
48	%CPMS-E-HARD-0013	EC=03B70000	System Bus Error (Master/Target Abort)	Bus target abort	Hardware	–	–	–	ON	Identify the failure location and replace the module.
49	%CPMS-E-HARD-0013	EC=03B80001	System Bus Error (CPU Master)	System bus error (access from ET.NET)	Hardware	ET.NET	–	–	ON	Replace the optional module.
50	%CPMS-E-HARD-0013	EC=03B80002	System Bus Error (CPU Target)	System bus error (access to ET.NET)	Hardware	ET.NET	–	–	ON	Replace the optional module.

Table C-3 LNET Error Messages (Built-in Ethernet)

No.	Error log title	Error code	Error message	Description	Fault category	Fault location	Recovery measure
1	%LNET-E-HARD-0004	EC=07801308	I/O error (SEND_TIMEOUT)	Transmission timeout error	LSI error	Hardware	Replace the hardware.
2	%LNET-E-HARD-0004	EC=0780130A	I/O error (RESET_ERROR)	Hardware reset error			Replace the hardware.
3	%LNET-W-HARD-0004	EC=07801308	I/O error (SEND_TIMEOUT)	Transmission timeout error (automatic recovery)			–
4	%LNET-W-HARD-0004	EC=0780130A	I/O error (RESET_ERROR)	Hardware reset error (automatic recovery)			–
5	%LNET-W-HARD-0004	EC=07801310	I/O error (LOSS)	Carrier loss error	Channel error	Line error handling	Check and correct the transmission line and replace the hardware (*1)
6	%LNET-W-HARD-0004	EC=07801311	I/O error (RETRY)	Retry error			Check and correct the transmission line.
7	%LNET-W-HARD-0004	EC=07801312	I/O error (LATE)	Late collision error			Check and correct the transmission line.
8	%LNET-E-HARD-0004	EC=07801505	I/O error (INV_INTR)	Invalid interrupt generated (detected by OS)	Software interface error	Interface error handling	Replace the hardware.
9	%LNET-E-SOFT-0004	EC=07801508	I/O error (BUF_OVF)	Overflow of transmission/reception buffer managed by OS			Check and correct the system design. (*2)
10	%LNET-W-SOFT-0004	EC=0780150D	I/O error (STATION_NUM)	System configuration error detected by OS (mismatch in station number)			Check and correct the system configuration.
11	%LNET-W-SOFT-0004	EC=0780150F	I/O error (SOCKET_OVF)	Socket table overflow (detected by OS)			Check and correct the system design.
12	%LNET-W-SOFT-0004	EC=07801510	I/O error (IFCONFIG_UP)	Initialization error (detected by OS)			Check and correct the system design.
13	%LNET-W-SOFT-0004	EC=07801511	I/O error (NETADDR_DUPL)	Duplicate network address error (detected by OS)			Check and correct the system configuration. (*3)
14	%LNET-W-SOFT-0004	EC=07801512	I/O error (IPADDR_DUPL)	Duplicate IP address error (detected by OS)			Check and correct the system configuration. (*4)

(*1) When the system detects an LSI carrier loss error 32 times, this message is output once.
In the case of built-in Ethernet, performing data transmission when LINK LED is off (no link is established) causes an LSI carrier loss error.
Therefore, this error occurs if an application program sends a transmission request 32 times or more before LINK LED turns on at system startup.
In this case, you must correct the application program to, for example, send requests after LINK LED turns on.

(*2) This error is caused by a buffer shortage due to high communication load. In this case, you must check and modify the system design as necessary, for example to adjust network load across the whole system.

(*3) You must check and modify the network settings as necessary, to assign unique addresses to ET1 and ET2 for built-in Ethernet.

(*4) Another Ethernet device is using the same IP address. You must check and modify the network settings as necessary.

Table C-4 LNET Error Messages (ET.NET)

No.	Error log title	Error code	Error message	Description	Fault category	Fault location	Recovery measure
1	%LNET-E-HARD-0004	EC=07801308	I/O error (SEND_TIMEOUT)	Transmission timeout error	Hardware	ET.NET	Turn the module off and then on again, or replace the optional module.
2	%LNET-E-HARD-0004	EC=0780130A	I/O error (RESET_ERROR)	Hardware reset error			Turn the module off and then on again, or replace the optional module.
3	%LNET-E-HARD-0004	EC=07801505	I/O error (INV_INTR)	Invalid interrupt generated			Replace the optional module.
4	%LNET-W-SOFT-0004	EC=07801510	I/O error (IFCONFIG_UP)	Network driver initialization error			Check and correct the settings.
5	%LNET-W-SOFT-0004	EC=07801511	I/O error (NETADDR_DUPL)	Duplicate network address error			Check and correct the network settings.
6	%LNET-W-SOFT-0004	EC=07801512	I/O error (IPADDR_DUPL)	Duplicate IP address error			Check and correct the IP address.
7	%LNET-W-SOFT-0004	EC=07807D1C	I/O error (Invalid network setting)	Communication setting not yet set			Configure the communication settings.

APPENDIX D INPUT DATA FOR BUILT-IN SUBROUTINES

(1) CPES input data format (PRGEB)

	Name	Description
0	pge_form	LOG_FORM_PRGERR
4	pge_frsz	Data size after pge_ecd (in bytes)
8	pge_ecd	Error code
12	pge_tn	Task number
16	pge_gr0_b0	General register R0 BANK0
20	pge_gr1_b0	General register R1 BANK0
24	pge_gr2_b0	General register R2 BANK0
28	pge_gr3_b0	General register R3 BANK0
32	pge_gr4_b0	General register R4 BANK0
36	pge_gr5_b0	General register R5 BANK0
40	pge_gr6_b0	General register R6 BANK0
44	pge_gr7_b0	General register R7 BANK0
48	pge_gr8	General register R8
52	pge_gr9	General register R9
56	pge_gr10	General register R10
60	pge_gr11	General register R11
64	pge_gr12	General register R12
68	pge_gr13	General register R13
72	pge_gr14	General register R14
76	pge_gr15	General register R15
80	pge_pc	Program counter
84	pge_sr	Status register
88	pge_pr	Procedure register
92	pge_gbr	Global base register
96	pge_mach	MAC upper register
100	pge_macl	MAC lower register
104	pge_expevt	expevt register
108	pge_fadr	Fault address
112	pge_fr0	Floating-point register FPR0 BANK0
116	pge_fr1	Floating-point register FPR1 BANK0
120	pge_fr2	Floating-point register FPR2 BANK0
124	pge_fr3	Floating-point register FPR3 BANK0
128	pge_fr4	Floating-point register FPR4 BANK0
132	pge_fr5	Floating-point register FPR5 BANK0
136	pge_fr6	Floating-point register FPR6 BANK0
140	pge_fr7	Floating-point register FPR7 BANK0
144	pge_fr8	Floating-point register FPR8 BANK0
148	pge_fr9	Floating-point register FPR9 BANK0
152	pge_fr10	Floating-point register FPR10 BANK0
156	pge_fr11	Floating-point register FPR11 BANK0
160	pge_fr12	Floating-point register FPR12 BANK0
164	pge_fr13	Floating-point register FPR13 BANK0
168	pge_fr14	Floating-point register FPR14 BANK0
172	pge_fr15	Floating-point register FPR15 BANK0
176	pge_fr16	Floating-point register FPR0 BANK1
180	pge_fr17	Floating-point register FPR1 BANK1
184	pge_fr18	Floating-point register FPR2 BANK1
188	pge_fr19	Floating-point register FPR3 BANK1
192	pge_fr20	Floating-point register FPR4 BANK1
196	pge_fr21	Floating-point register FPR5 BANK1
200	pge_fr22	Floating-point register FPR6 BANK1
204	pge_fr23	Floating-point register FPR7 BANK1
208	pge_fr24	Floating-point register FPR8 BANK1
212	pge_fr25	Floating-point register FPR9 BANK1
216	pge_fr26	Floating-point register FPR10 BANK1
220	pge_fr27	Floating-point register FPR11 BANK1
224	pge_fr28	Floating-point register FPR12 BANK1
228	pge_fr29	Floating-point register FPR13 BANK1
232	pge_fr30	Floating-point register FPR14 BANK1
236	pge_fr31	Floating-point register FPR15 BANK1
240	pge_fpscr	Floating-point status, control register
244	pge_fpul	Floating-point communication register
248	pge_iarvn9	Contents of address - 36 indicated by program counter
252	pge_iarvn8	Contents of address - 32 indicated by program counter
256	pge_iarvn7	Contents of address - 28 indicated by program counter
260	pge_iarvn6	Contents of address - 24 indicated by program counter
264	pge_iarvn5	Contents of address - 20 indicated by program counter
268	pge_iarvn4	Contents of address - 16 indicated by program counter
272	pge_iarvn3	Contents of address - 12 indicated by program counter
276	pge_iarvn2	Contents of address - 8 indicated by program counter
280	pge_iarvn1	Contents of address - 4 indicated by program counter
284	pge_iarv0	Contents of address indicated by program counter
288	pge_iarv1	Contents of address + 4 indicated by program counter

(2) IES input data format (IOERB)

	Name	Description
0	ioe_form	Format type (in this case, LOG_FORM_IOERR)
4	ioe_frsz	Data size after ioe_ecd (in bytes)
8	ioe_ecd	Error code
12	ioe_uno	Unit number
16	ioe_dev	Device number
20	ioe_dva	Device address
24	ioe_ioec	Detailed error code
28	ioe_tn	Task number (-1 is used if the task number is invalid.)
32	ioe_data[110]	Detailed information about I/O error This information varies depending on the I/O.
472		

(3) EAS input data format (ADB)

	Name	Description
0	adb_logno	Error log number
4	adb_timestamp	Time (host clock value)
8	adb_type	Severity type
12	adb_class	Failure-detecting component class
16	adb_retcode	Return code from function when failure is detected
20	adb_errtype	Failure type (Hardware/MK/CPMS/other)
22	adb_flag	Error message flag (for example, to suppress display)
24	adb_site[16]	Site name
40	erb[118]	Error block (failure report data) Although the area size is fixed at 472 bytes, the effective data size varies depending on the format type.
512	adb_dhpbuf[128]	DHP data (512 bytes)
1024		

(4) PCKS input data format (SVCEB)

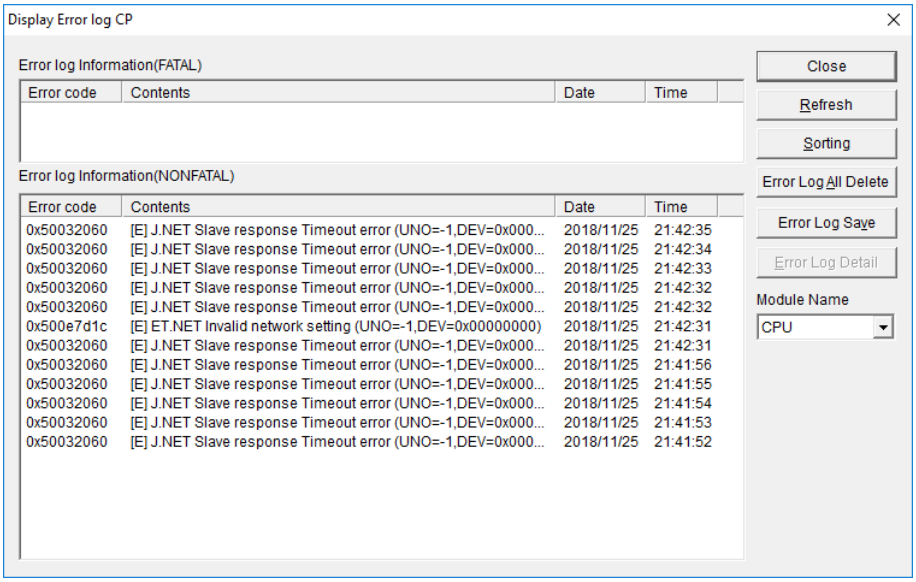
	Name	Description
0	sve_form	Format type (in this case, LOG_FORM_PARAMERR)
4	sve_frsz	Data size after sve_ecd (in bytes)
8	sve_ecd	Error code
12	sve_tn	Task number
16	sve_svc	Macro ID
20	sve_epn	Error parameter number
24	sve_p1	Macro instruction parameter 1
28	sve_p2	Macro instruction parameter 2
32	sve_p3	Macro instruction parameter 3
36	sve_p4	Macro instruction parameter 4
40	sve_p5	Macro instruction parameter 5
44	sve_p6	Macro instruction parameter 6
48	sve_p7	Macro instruction parameter 7

(5) MODES input data format (HARDEB)

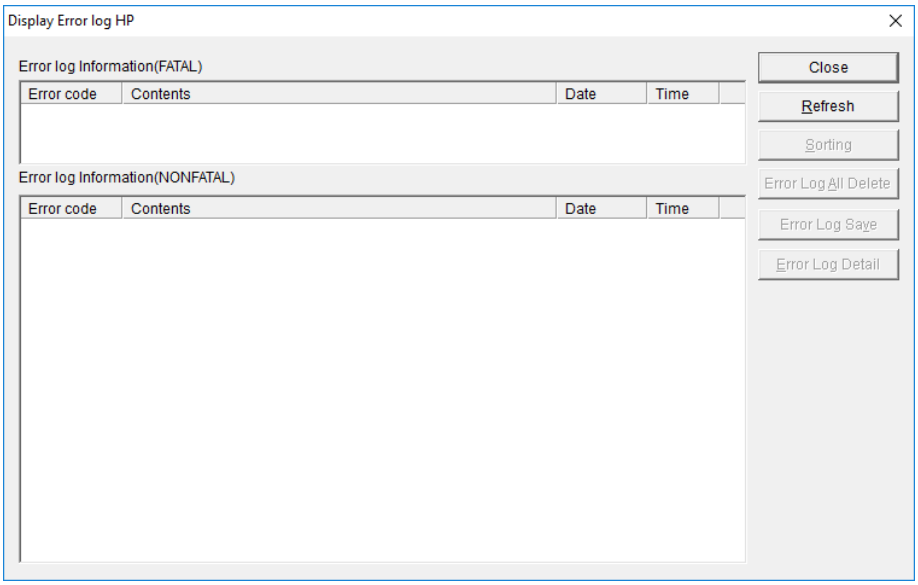
	Name	Description
0	mde_form	Format type (in this case, LOG_FORM_MODULERR)
4	mde_frsz	Data size after mde_ecd (in bytes)
8	mde_ecd	Error code
12	mde_slot	Slot number
16	mde_msw0	Module status word 0 (-1 is used if this word is invalid.)
20	mde_msw1	Module status word 1 (-1 is used if this word is invalid.)
24	mde_data[112]	Detailed module error format
472		

APPENDIX E GUIDELINES FOR DISPLAYING ERROR LOGS

You can display error log information by selecting a menu item from the main menu of BASE SYSTEM/S10VE.
To display error logs for the CP side, click **RAS**, **Error Log Display**, and then **CP Error Log Display**.
To display error logs for the HP side, click **RAS, Error Log Display**, and then **HP Error Log Display**.
The Display Error log CP or Display Error log HP window appears.



Display Error log CP window



Display Error log HP window

In the Display Error log CP window, you can also specify a CP for which to check log information from the **Module Name** drop-down list.
Depending on the installed modules, the following module names might appear in the drop-down list: **CPU**, **ET.NET (Main)**, and **ET.NET (Sub)**. From among these names, you can select the module that you want to check. For the HP, you can only check the error log information for the CPU.
Among the error information that is displayed when you select **ET.NET (Main)** or **ET.NET (Sub)**, `CPMS` indicates the firmware in ET.NET, not the CPMS operated by the CPU.

Error log information is displayed in the following formats.

Panic logs

```
[*] ******** (PC = ********, FADR = ********)
(1)    (2)            (3)              (4)
```

(1) Error severity type
[F]: Fatal error
[FU]: Built-in subroutine error
(2) Error message
(3) Program counter
(4) Fault address

Logs other than panic logs

(Pattern 1)

```
[*] ******** (UNO = **, DEV = ********) (TN = ***) (SLOT = **)
(1)    (2)              (3)                 (4)        (5)
```

(1) Error severity type
[F]: Fatal error [W]: Warning
[E]: Error [I]: Information
(2) Error message
(3) Unit number, device number
Unit range: 1 to 24
Device range: 0x00000000 to 0xFFFFFFFF
(4) Task number
Task range: 1 to 300
(5) Slot number
Slot range: 0 to 7

Note: Depending on the type of error, the information indicated in (3) to (5) of pattern 1 of the preceding *Logs other than panic logs* might not be displayed.

(Pattern 2)

```
%****-*-****-****
  (1) (2) (3)  (4)
```

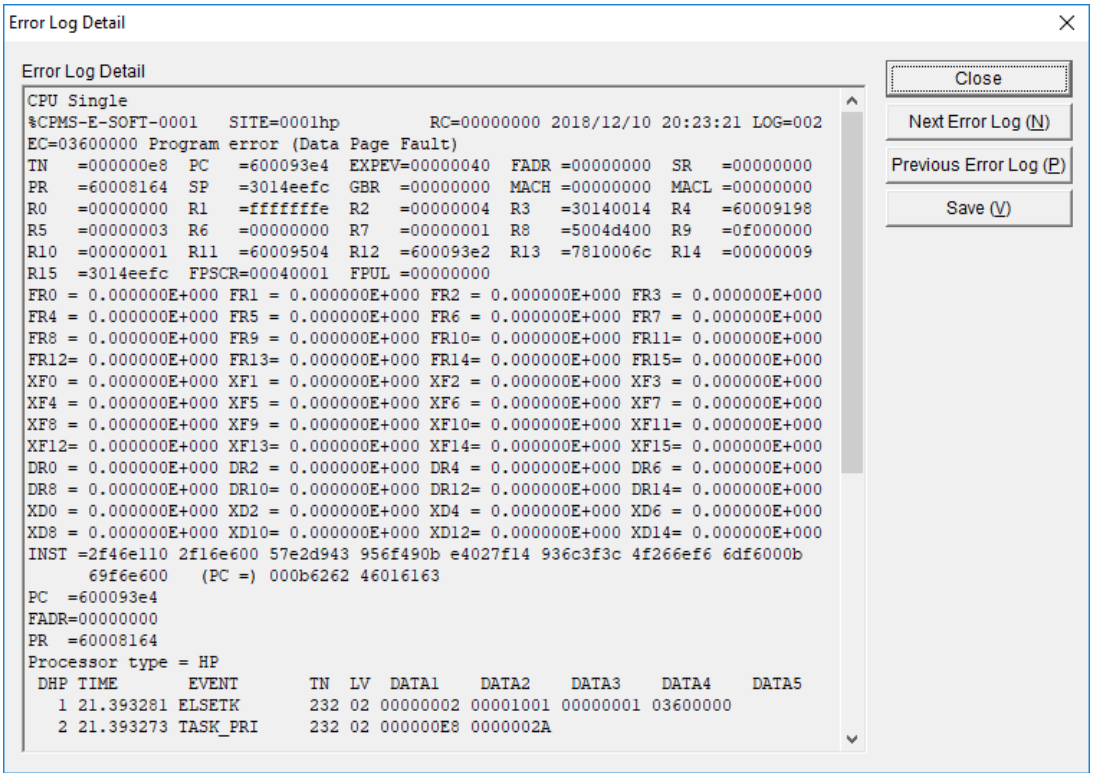
(1) System that detected an error
CPMS: CPMS (basic OS)
LNET: RCTLNET (network driver)
NX: NXACP (autonomous distributed platform)
MS*xx*: Middleware (*xx* is a number from 01 to 16.)
US*xx*: Application software (*xx* is a number from 01 to 16.)

(2) Error severity type
F: Fatal error E: Error
W: Warning I: Information
?: Other error

(3) Fault category
HARD: Hardware
CPMS: CPMS
SOFT: Software other than CPMS

(4) Code
A code consisting of a 4-digit hexadecimal number that represents the error log type

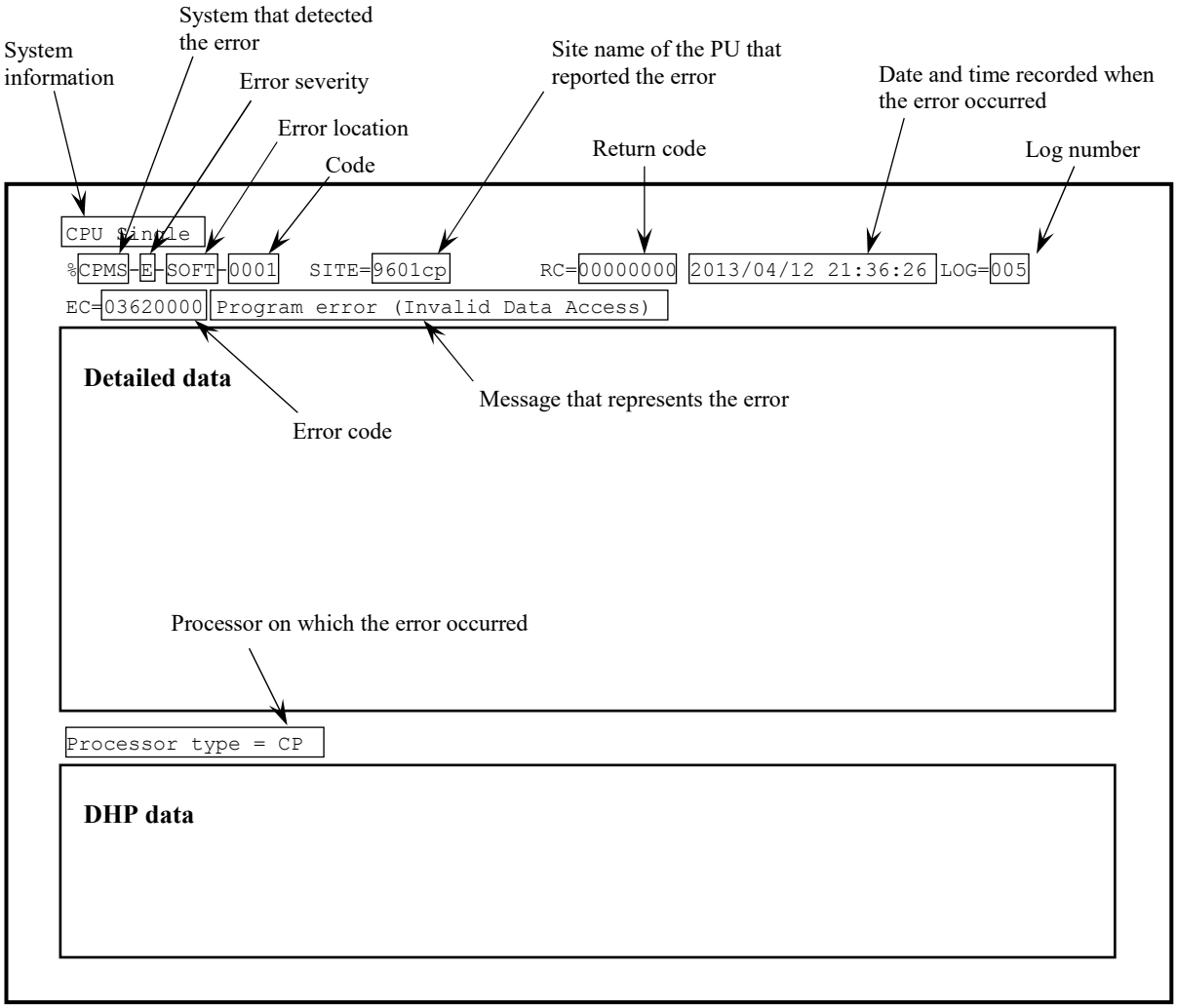
To display detailed information about the error, click the [Error Log Detail] button.
The Error Log Detail window appears.



This section describes the details displayed about error logs.

E.1 Reading Error Logs

This section describes how to read error logs by using the following displayed error log details as an example:



- System information
  CPU Single: Single CPU configuration

- System that detected the error
  CPMS: CPMS (basic OS)
  LNET: RCTLNET (network driver)
  NX: NXACP (autonomous distributed platform)
  MS*xx*: Middleware (*xx* is a number from 01 to 16.)
  US*xx*: Application software (*xx* is a number from 01 to 16)

- Message representing the error (See APPENDIX C LIST OF ERROR MESSAGES.)
  Program error: Error causing the failure in program execution
  Macro parameter check error: OS macro instruction parameter error
  WDT timeout error: WDT (watchdog timer) monitoring timeout error
  I/O error: Error related to I/O
  Module error: Error mainly in the hardware
  For details, see E.2 Types of Error Logs.

- Error severity
  F: Fatal error
  E: Error
  W: Warning
  I: Information

- Error location
  HARD: Hardware
  CPMS: CPMS
  SOFT: Software other than CPMS

- Code
  Code representing the error log type

- Processor on which the error occurred
  CP: The error occurred on the CP side.
  HP: The error occurred on the HP side.

E.2 Types of Error Logs

(1) Types of OS error logs
Table E-1 lists the types of OS error logs. For a list of error messages, see APPENDIX C.

Table E-1 Types of OS Error Logs

Code	Log format name	Error message	Main error information			
0001	Program error	Program error (subtitle)	EC	TN	PC	FADR
0002	Macro parameter error	Macro parameter error	EC	TN	SVC	
0004	I/O error	I/O error (subtitle)	EC	UNO	DEV	
0005	WDT timeout error	WDT timeout error	EC			
0006	Module error	Module error (subtitle)	EC	SLOT		
0007	Kernel warning	Kernel warning	EC			
0008	Kernel information	Kernel Information	EC			
0009	System down (system error)	System down (subtitle)	EC	TN	PC	FADR
000A	System down (kernel trap)	System down (kernel trap)	EC	FILE	LINE	
000B	System down (built-in subroutine error)	ULSUB down (subtitle)	EC	NEST	POINT	ENTRY
000C	System down (built-in subroutine stop)	System down (ULSUB stop)	EC	NEST	POINT	
0012	Memory error	Memory error	EC	HERST		
0010	Ladder program error	Ladder program error	EC	FADR		
0013	System bus error	System bus error	EC	HERST		

EC: Error code
TN: Task number
SVC: Macro code
PC: Instruction address
FADR: Field address
HERST: Serious fault register
UNO: I/O unit number
DEV: Device number
SLOT: Slot number
FILE: File name
LINE: Line number
NEST: Nest
POINT: Point
ENTRY: Entry

E.3 Error Log Details and Analysis Method

E.3.1 Program error

This error occurs when the program experiences a failure and attempts to access an abnormal address or execute an invalid instruction.

To analyze the cause of a failure, perform the following procedure:

(1) Check the error message text such as the error name for information about what happened. For error messages and their descriptions, see Table E-2 and Table E-3.

(2) Check the information in the error message such as the register and stack information to identify the failure location.

For a more detailed analysis procedure, see Figure E-1.

Table E-2 Program Error Message Format (1/2)

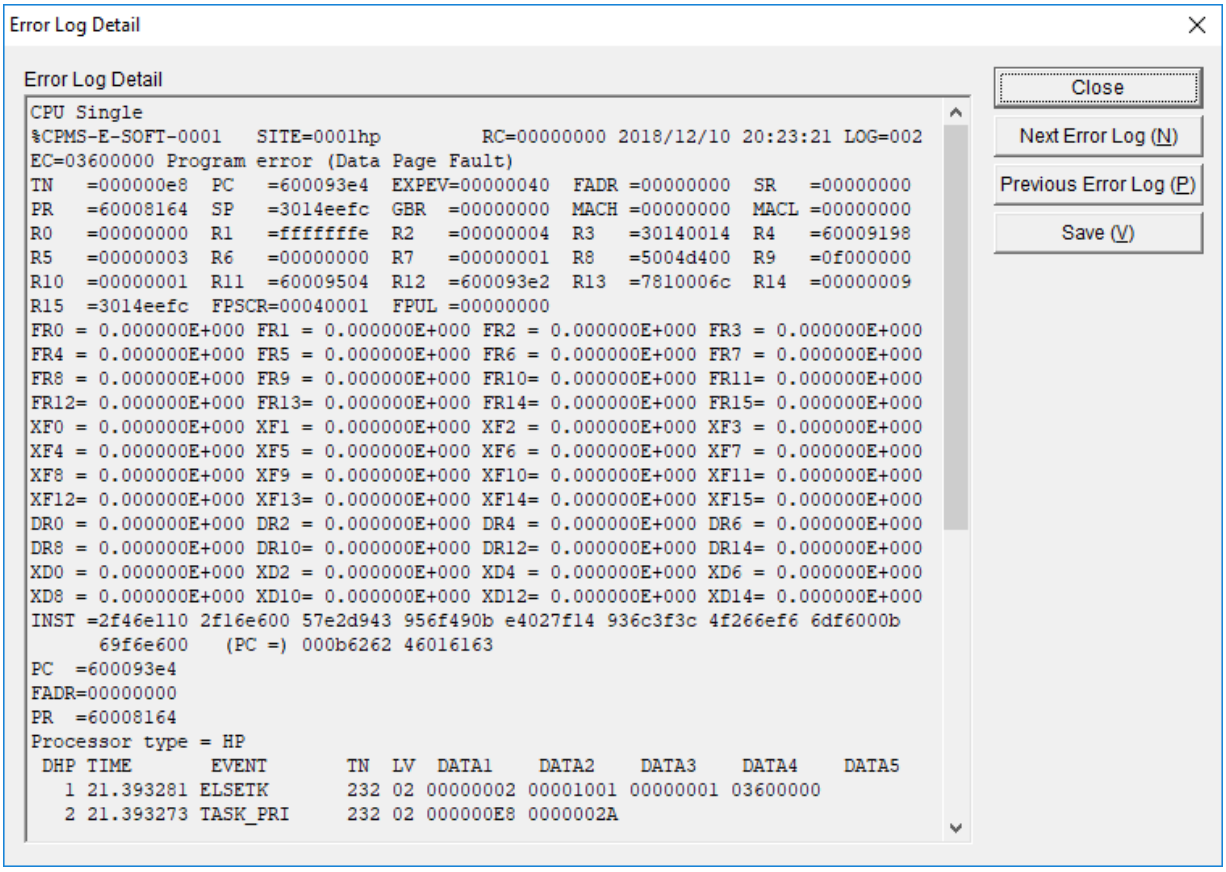


Table E-2 Program Error Message Format (2/2)

Item	Description
RC	Return code
EC	Error code (See Table E-3.)
TN	Task number of the task that caused the error
PC	Contents of the program counter
EXPEV	Contents of the exception code register. The exception code register is a 32-bit register defining the causes of data access exceptions and alignment exceptions.
FADR	Fault address (This indicates the address being accessed when the error occurred.)
SR	Status register
PR	Contents of the procedure register. The procedure register is used for subroutine calls. If the executed program was the last program in the subroutine calling sequence, this register contains the return address.
SP	Contents of the stack pointer (R15 is used as a stack pointer).
GBR	Contents of the global base register. This register contains the base addresses for GBR-indirect addressing with displacement and indexed GBR-indirect addressing.
MACH	MAC register. This register is used for storing additional values of MAC (multiply and accumulate operation) instructions, MAC instructions, and results of MUL instructions. If the calculated result is a value of 64 bits or more, this register stores the upper 32 bits.
MACL	MAC register. If the calculated result is a 64-bit value, this register stores the lower 32 bits. If the calculated result is a 32-bit value, this register stores 32 bits.
R*xx*	Contents of the general register represented by the value at *xx*
FPSCR	Contents of the floating-point status and control register
FPUL	Contents of the floating-point communication register. Data is transferred between a general register and floating-point register through this register.
FR*xx*	Contents of the 32-bit floating-point register represented by the value at *xx*. When FPSCR.FR (bit 21 of a 31-0-bit value) = 0, this is the value of FPR*xx*_BANK0. When FPSCR.FR = 1, this is the value of FPR*xx*_BANK1.
XF*xx*	Contents of the 32-bit floating-point register represented by the value at *xx*. When FPSCR.FR (bit 21 of a 31-0-bit value) = 0, this is the value of FPR*xx*_BANK1. When FPSCR.FR=1, this is the value of FPR*xx*_BANK0.
DR*xx*	Contents of the 64-bit floating-point register represented by the value at *xx*. When FPSCR.FR (bit 21 of a 31-0-bit value) = 0, this is the value of FPR*xx*_BANK0. When FPSCR.FR=1, this is the value of FPR*xx*_BANK1.
XD*xx*	Contents of 64-bit floating-point register *xx*. When FPSCR.FR (bit 21 of a 31-0-bit value) = 0, this is the value of FPR*xx*_BANK1. When FPSCR.FR=1, this is the value of FPR*xx*_BANK0.
INST	Instruction code
PC	Address information contained in a program counter is displayed in parentheses. When the address is for a program, the following information is displayed: name = *Program name*, type = *Program type (program location)*, raddr = *Relative address from the program*
FADR	Information about the fault address is displayed in parentheses.
PR	Information about the address contained in a procedure register is displayed in parentheses. When the address is for a program, the following information is displayed: name = *program name*, type = *program type (program location)*, raddr = *relative address from the program*

Table E-3 Error Codes, Subtitles, and their Descriptions (program errors)

No.	Error code	Subtitle	Description	Explanation
1	EC=03030000	Inst. Alignment Error	Instruction alignment error	Operands specified by the instruction are not word-aligned.
2	EC=03040000	Illegal Instruction	Illegal instruction error	An attempt was made to execute an illegal instruction.
3	EC=03080000	Privileged Instruction	Privileged instruction error	A privileged instruction was executed (an instruction that can be issued in system mode only).
4	EC=03390000	FP Program Error	Floating-point calculation error	An error occurred in a floating-point instruction.
5	EC=03400000	Instruction Page Fault	Instruction access page fault	An instruction access was made to a page whose address is not in the page table.
6	EC=03420000	Invalid Inst. Access	Instruction access error	The address area after 0x80000000 was accessed (instruction access error due to an error factor other than EC=03400000 and EC=03460000).
7	EC=03460000	Inst. Access Protection	Instruction access protection error	An instruction access infringed memory protection.
8	EC=03600000	Data Page Fault	Data access page fault	A data access was made to a page whose address is not in the page table.
9	EC=03620000	Invalid Data Access	Data access error	The address area after 0x80000000 was accessed. (A data access error due to an error factor other than EC=03600000 and EC=03660000)
10	EC=03660000	Data Access Protection	Data access protection error	A data access infringed memory protection.
11	EC=03470000	Data Alignment Error	Data alignment error	One of the following accesses was made: word data access from outside the word boundary (2n + 1), long word data access from outside the long word data boundary (4n + 1, 4n + 2, 4n + 3), quad word data access from outside the quad word data boundary.

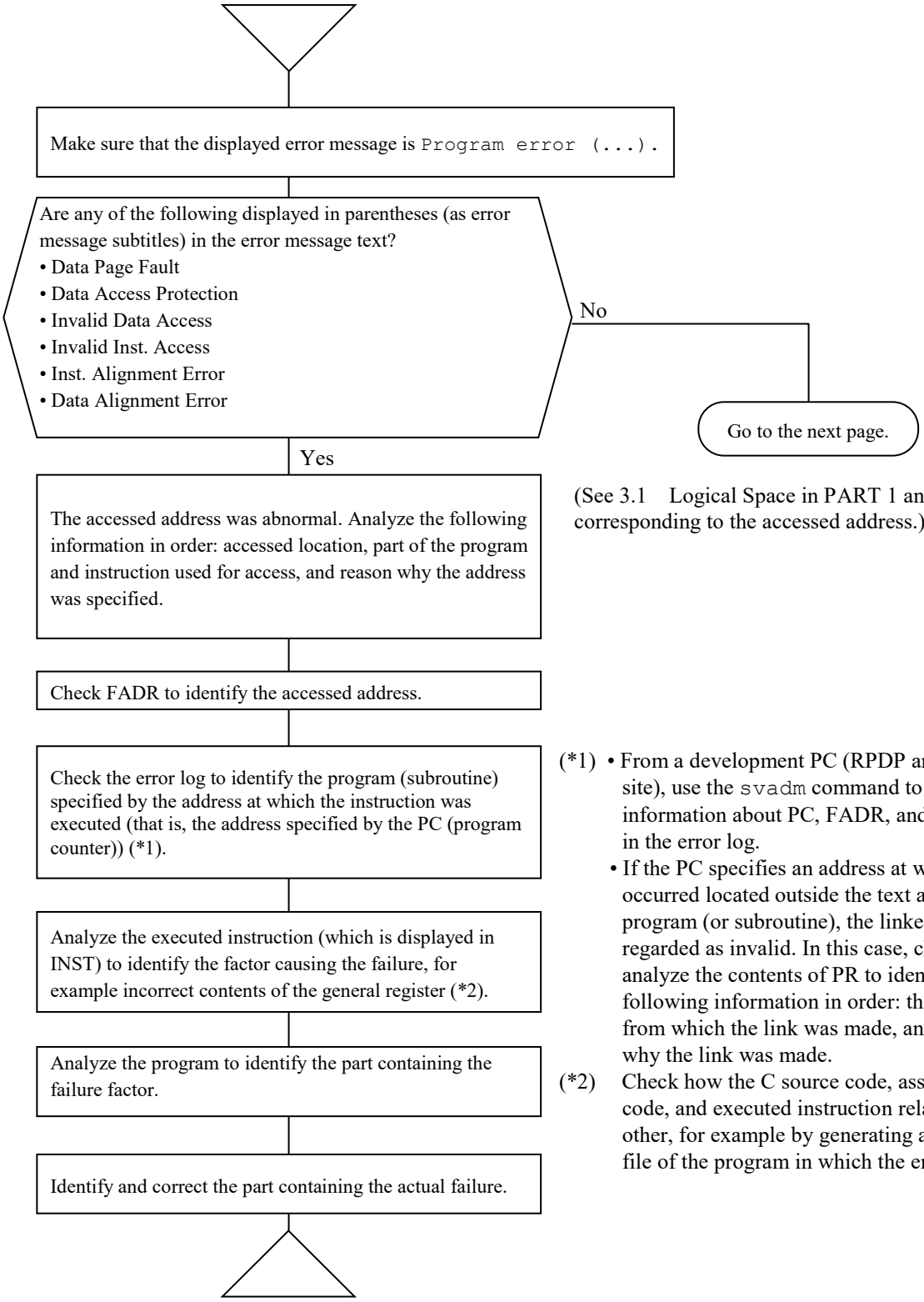


Figure E-1 Procedure for Analyzing Program Errors (1/2)

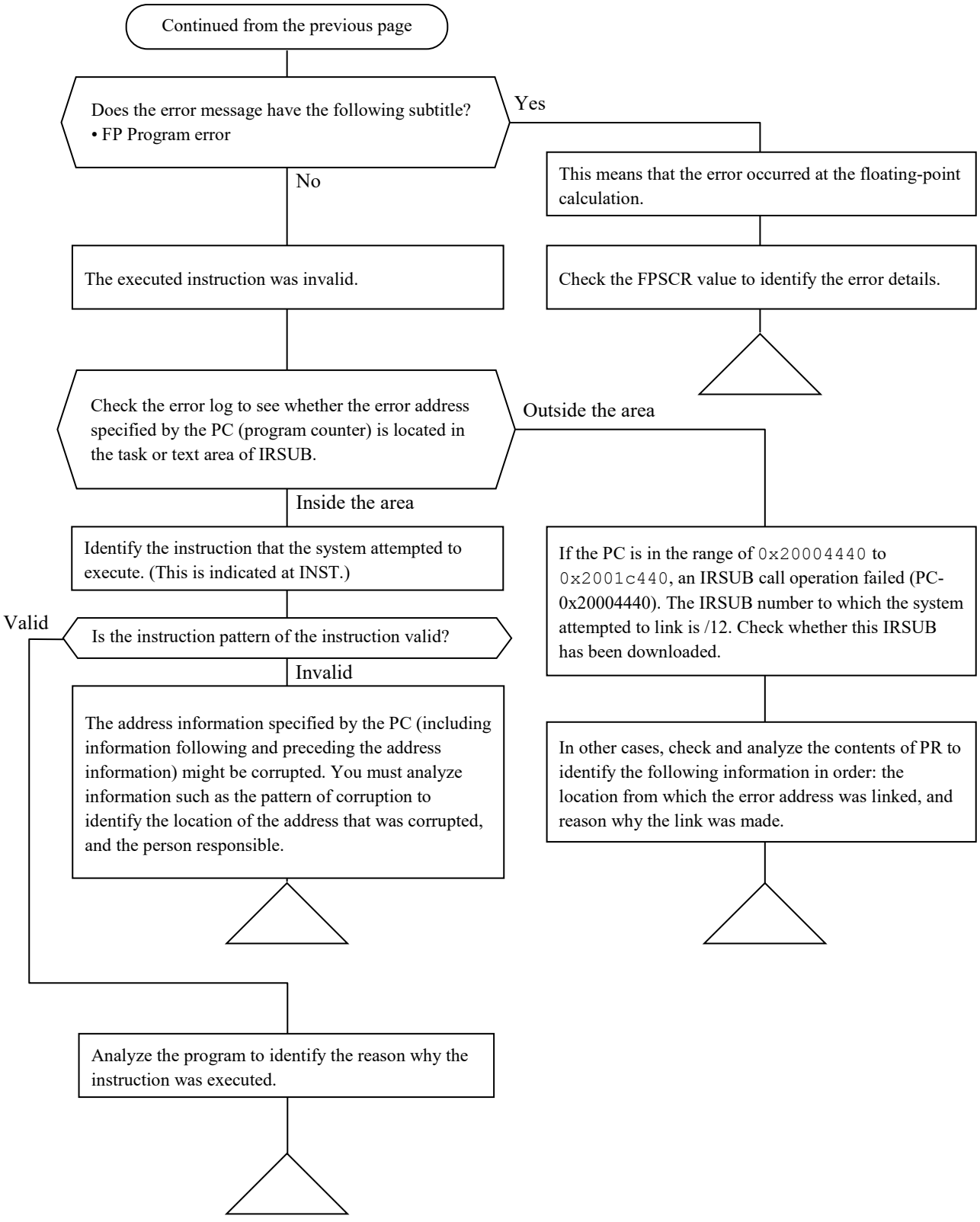


Figure E-1 Procedure for Analyzing Program Errors (2/2)

This section provides an example analysis using the following actual error message:

```
CPU Single
%CPMS-E-SOFT-0001   SITE=0001cp        RC=00000000 2013/04/12 14:36:46 LOG=006
EC=03600000 Program error (Data Page Fault)
TN   =00000067  PC   =300281a6  EXPEV=00000060  FADR =48000000  SR   =00008100
PR   =300281a4  SP   =3002d584  GBR  =00000000  MACH =00000000  MACL =00000000
R0   =ffffffff  R1   =0c650000  R2   =fc5b18f8  R3   =4d2c2032  R4   =300287ec
R5   =000018ca  R6   =0003d090  R7   =0000ca83  R8   =00059f46  R9   =300284ec
R10  =48000000  R11  =00000001  R12  =00000c38  R13  =300284bc  R14  =0000ca83
R15  =3002d584  FPSCR=00040e00  FPUL =00000000
FR0 = 0.000000E+000 FR1 = 0.000000E+000 FR2 = 0.000000E+000 FR3 = 0.000000E+000
FR4 = 0.000000E+000 FR5 = 0.000000E+000 FR6 = 0.000000E+000 FR7 = 0.000000E+000
FR8 = 0.000000E+000 FR9 = 0.000000E+000 FR10= 0.000000E+000 FR11= 0.000000E+000
FR12= 0.000000E+000 FR13= 0.000000E+000 FR14= 0.000000E+000 FR15= 0.000000E+000
XF0 = 0.000000E+000 XF1 = 0.000000E+000 XF2 = 0.000000E+000 XF3 = 0.000000E+000
XF4 = 0.000000E+000 XF5 = 0.000000E+000 XF6 = 0.000000E+000 XF7 = 0.000000E+000
XF8 = 0.000000E+000 XF9 = 0.000000E+000 XF10= 0.000000E+000 XF11= 0.000000E+000
XF12= 0.000000E+000 XF13= 0.000000E+000 XF14= 0.000000E+000 XF15= 0.000000E+000
DR0 = 0.000000E+000 DR2 = 0.000000E+000 DR4 = 0.000000E+000 DR6 = 0.000000E+000
DR8 = 0.000000E+000 DR10= 0.000000E+000 DR12= 0.000000E+000 DR14= 0.000000E+000
XD0 = 0.000000E+000 XD2 = 0.000000E+000 XD4 = 0.000000E+000 XD6 = 0.000000E+000
XD8 = 0.000000E+000 XD10= 0.000000E+000 XD12= 0.000000E+000 XD14= 0.000000E+000
INST =0cfeea48 3e88d45c d64f3ec8 d95c4d0b 67e3d15b 60e3d45b 490b4a18 65034d0b
      4a28eb01   (PC =) 2ab2930b 3f3c4f26
PC  =300281a6
FADR=48000000
PR  =300281a4
```

Step 1

The displayed error message is `Program error (Data Page Fault)`. This means that a data access was made to a page whose address is not in the page table.

Step2

`FADR=48000000` means that the error occurred when the address `0x48000000` was accessed.

Step 3

`PC=300281a6` means that the address of the instruction that caused the error was `0x300281a6`.

Step 4

From a development PC (RPDP and PC with site), use the `svadem` command.

```
C:\site0001>svadm -u 0001cp 0x300281a6
name =dry type = task(TEXT) raddr = 000001a6
```

Based on the result of the `svadm` command (`name` = *program name*, `type` = *program type* (*program location*), `raddr` = *relative address from the program*), identify the program and its instruction that caused the error.

(If the address `0x300281a6` is not located in the text area of the program (or subroutine), a link was made to an unfixed address. In this case, check the contents of PR to identify the location from which the link was made.

Step 5

INST displays the instruction that caused the error in addition to its preceding and following instructions. According to the displayed information, the instruction that caused the error was `0x2ab2`, which corresponds to the following hardware instruction:

MOV.L R11, @R10

This hardware instruction specifies to load the contents of general register 11 to the address in general register 10.

General register 10 contains `R10=48000000`, which is the same value as FADR. Therefore, the direct cause of the error was the value `48000000` that was specified for R10.

Step 6

Check the instructions that precede the error instruction. Because the hardware instructions are fixed at 2 bytes, you can check 18 instructions preceding the error instruction and 3 instructions following the error instruction. The following lists the instructions preceding the instruction in the PC:

PC-36	MOV.L	@(R0,R15),R12
PC-34	MOV	#72,R10
PC-32	SUB	R8,R14
PC-30	MOV.L	L70+58,R4
PC-28	MOV.L	L70+6,R6
PC-26	SUB	R12,R14
PC-24	MOV.L	L70+62,R9
PC-22	JSR	@R13
PC-20	MOV	R14,R7
PC-18	MOV.L	L70+66,R1
PC-16	MOV	R14,R0
PC-14	MOV.L	L70+70,R4
PC-12	JSR	@R9
PC-10	SHLL8	R10
PC-8:	MOV	R0,R5
PC-6:	JSR	@R13
PC-4:	SHLL16	R10
PC-2:	MOV	#1,R11
PC :	MOV.L	R11,@R10

General register 10 contains `R10=48000000`. Therefore, find the instruction that was assigned to general register 10.

You will find that 0x00000048 is assigned from PC-34 to general register 10, and general register 10 is shifted to the left by 24 bits according to the instructions at addresses PC-10 and PC-4. This is why the value was changed to 0x48000000.

Step 7

This information shows a high probability that the value 0x48000000 was directly assigned to a pointer variable or similar variable. Therefore, check whether the program contains an invalid process.

In other cases, if any invalid address is assigned from the memory to general register 10, check and identify the following information: the person who stored the invalid address value 0x48000000 in the memory, the location where the value was stored, and the reason why the value was stored.

E.3.2 Macro parameter check error

This error indicates that a CPMS macro instruction issued by the program contained an irrational parameter. In this case, the task that issued the macro is aborted. Check the error message to identify the irrational parameter and then correct it.

Table E-4 lists the error messages and their descriptions.

Table E-4 Format of a Macro Parameter Check Error Message

```
CPU xxxxxxx
%CPMS-E-SOFT-0002  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx Macro parameter error
TN  =xxxxxxxx  SVC  =xxxxxxxx
EPN  =xxxxxxxx  PARA1=xxxxxxxx  PARA2=xxxxxxxx  PARA3=xxxxxxxx  PARA4=xxxxxxxx
PARA5=xxxxxxxx  PARA6=xxxxxxxx  PARA7=xxxxxxxx
```

Item	Description
EC	Error code: This represents the error type. EC: 05130000 — An undefined macro instruction was issued (SVC is abnormal). EC: 05110000 — The contents of the parameter are abnormal. EPN indicates the number of the abnormal parameter and PARA*n* indicates the value of the parameter itself.
TN	Task number: This is the task number of the task that issued the macro instruction.
SVC	Supervisory macro code This code represents the type of the issued macro. For information on how the codes and macro names correspond to each other, see Table E-5.
EPN	Error parameter number This indicates the number of the irrational parameter. This is valid only when EC = 05110000.
PARA*n*	Parameter *n* This indicates the contents of the parameter in order. For example, PARA1 corresponds to the first parameter and PARA2 corresponds to the second parameter. <Example> When SVC = 0000000A (timer macro): PARA1: id PARA2: tn PARA3: fact PARA4: t PARA5: cyt } Correspondence between PARA*n* and parameters Therefore, if EPN = 00000004, PARA4(the value of t) is regarded as irrational.

Table E-5 SVCs (Supervisory Macro Codes) and their Corresponding Macro Names

SVC / i	0000000i	0000001i	0000002i	0000003i	0000004i	0000005i
0	–	prsrv	usrel		atmcas	
1	queue	pfree	Elset (*)	gtkmem	prog_start	
2	rleas	gfact	cpms_ginfo (*)	wrtmem	prog_switch	
3	sfact	gtime	Chml (*)	chkbmem	prog_exit	
4	abort	exit	taskenv (*)	chktaer	prog_call	
5	susp	asusp	printf (*)	getsysinfo		
6	rsum	arsum		gettaskinfo		
7	ctime	open (*)		save_env		
8	wait	close (*)	wdtset	resume_env		
9	post	read (*)		gettimebase		
A	timer	write (*)		atmswap		
B	delay	ioctl (*)		atmand		
C	stime	usrdhp		atmor	romread (*)	
D	chap	dhpset (*)		atmxor	romwrite (*)	
E	resrv	dhpctl (*)		atmadd		
F	free	dhpread (*)		atmtas		

SVC / i	0000006i	0000007i	0000008i	0000009i	000000Ai	000000Bi
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						
A						
B						
C						
D						
E						
F						

(*) Dedicated macro for a subsystem in the CPMS

E.3.3 I/O errors

(1) Network I/O error

This error indicates that an error occurred in the network hardware or on the network transmission line when a program executed the network access macro or when a library was used for network access. Table-E-6 lists the error messages and their descriptions.

Table E-6 Format of a Network I/O Error Message (1/2)

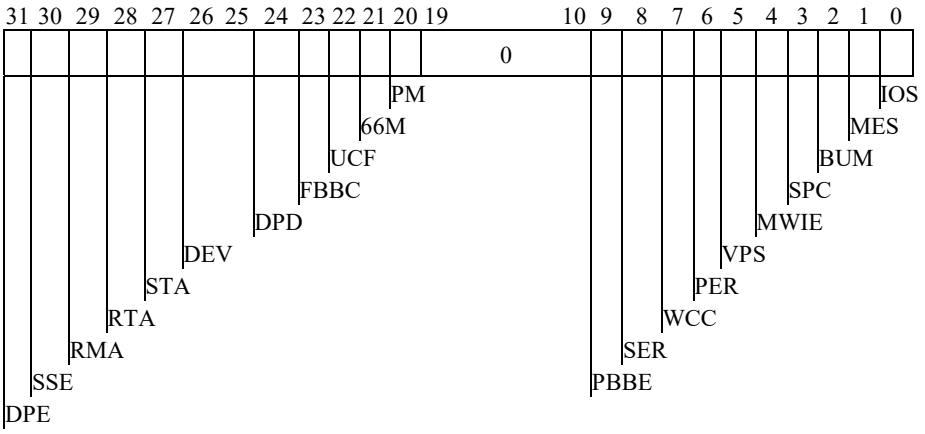
```
CPU xxxxxxxx
%LNET-x-xxxx-0004  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx I/O error (subtitle)
UNO  =xxxxxxxx  DEV  =xxxxxxxx  DVA  =xxxxxxxx  IOEC =xxxxxxxx
TN   =xxxxxxxx
DAT000=xxxxxxxx DAT001=xxxxxxxx DAT002=xxxxxxxx DAT003=xxxxxxxx DAT004=xxxxxxxx
DAT005=xxxxxxxx DAT006=xxxxxxxx DAT007=xxxxxxxx DAT008=xxxxxxxx DAT009=xxxxxxxx
DAT010=xxxxxxxx DAT011=xxxxxxxx DAT012=xxxxxxxx DAT013=xxxxxxxx DAT014=xxxxxxxx
DAT015=xxxxxxxx DAT016=xxxxxxxx DAT017=xxxxxxxx DAT018=xxxxxxxx DAT019=xxxxxxxx
DAT020=xxxxxxxx DAT021=xxxxxxxx DAT022=xxxxxxxx DAT023=xxxxxxxx DAT024=xxxxxxxx
DAT025=xxxxxxxx DAT026=xxxxxxxx DAT027=xxxxxxxx DAT028=xxxxxxxx DAT029=xxxxxxxx
DAT030=xxxxxxxx DAT031=xxxxxxxx DAT032=xxxxxxxx DAT033=xxxxxxxx DAT034=xxxxxxxx
DAT035=xxxxxxxx DAT036=xxxxxxxx DAT037=xxxxxxxx DAT038=xxxxxxxx DAT039=xxxxxxxx
DAT040=xxxxxxxx DAT041=xxxxxxxx DAT042=xxxxxxxx DAT043=xxxxxxxx DAT044=xxxxxxxx
DAT045=xxxxxxxx DAT046=xxxxxxxx DAT047=xxxxxxxx DAT048=xxxxxxxx DAT049=xxxxxxxx
DAT050=xxxxxxxx DAT051=xxxxxxxx DAT052=xxxxxxxx DAT053=xxxxxxxx DAT054=xxxxxxxx
DAT055=xxxxxxxx DAT056=xxxxxxxx DAT057=xxxxxxxx DAT058=xxxxxxxx DAT059=xxxxxxxx
DAT060=xxxxxxxx DAT061=xxxxxxxx DAT062=xxxxxxxx DAT063=xxxxxxxx DAT064=xxxxxxxx
DAT065=xxxxxxxx DAT066=xxxxxxxx DAT067=xxxxxxxx DAT068=xxxxxxxx DAT069=xxxxxxxx
DAT070=xxxxxxxx DAT071=xxxxxxxx DAT072=xxxxxxxx DAT073=xxxxxxxx DAT074=xxxxxxxx
DAT075=xxxxxxxx DAT076=xxxxxxxx DAT077=xxxxxxxx DAT078=xxxxxxxx DAT079=xxxxxxxx
DAT080=xxxxxxxx DAT081=xxxxxxxx DAT082=xxxxxxxx DAT083=xxxxxxxx DAT084=xxxxxxxx
DAT085=xxxxxxxx DAT086=xxxxxxxx DAT087=xxxxxxxx DAT088=xxxxxxxx DAT089=xxxxxxxx
DAT090=xxxxxxxx DAT091=xxxxxxxx DAT092=xxxxxxxx DAT093=xxxxxxxx DAT094=xxxxxxxx
DAT095=xxxxxxxx DAT096=xxxxxxxx DAT097=xxxxxxxx DAT098=xxxxxxxx DAT099=xxxxxxxx
DAT100=xxxxxxxx DAT101=xxxxxxxx DAT102=xxxxxxxx DAT103=xxxxxxxx DAT104=xxxxxxxx
DAT105=xxxxxxxx DAT106=xxxxxxxx DAT107=xxxxxxxx DAT108=xxxxxxxx DAT109=xxxxxxxx
```

Table E-6 Format of a Network I/O Error Message (2/2)

Item	Description
EC	Error code This represents the error type. See Table C-3 and Table C-4.
UNO	I/O unit number: This indicates the unit number.
DEV	Device number This indicates the type of I/O where the error occurred, and its mounted location. 31 16 15 12 11 8 7 0 Major number (=Driver ID) | SLOT | CH | Other Major number = 4: Built-in Ethernet/ET.NET SLOT: Slot number, CH: Channel (interface) number, Others: Depends on the device driver
DVA	Device address This indicates the mounted location of the device. This error is fixed to 0.
IOEC	I/O error code This indicates the detailed error code. 0x8xxxxxxx: Stopped due to an abnormal adapter 0x2xxxxxxx: Device was restarted by the CPMS (only for built-in Ethernet)
TN	Task number This indicates the task number at which the error occurred.
DAT*n*	Data *n* This indicates the detailed error data. The contents of this data vary depending on each EC. See Table E-7 and Table E-8.

Note: If the displayed value is 0xFFFFFFFF, the data is invalid.

Table E-7 Detailed Data for I/O Errors Detected in Built-in Ethernet/ET.NET (EC=0x078013XX) (1/3)

DATn	Description	Detailed data
DAT0	NB_PCICONF1	

Bit no.	Bit name	Read-in value	Description
31	DPE	0: No PCI parity error 1: PCI parity error detected	This indicates the parity error status.
30	SSE	0: No SERR 1: SERR# assertion detected	This indicates the SERR# status.
29	RMA	0: No master abort 1: Master abort detected	This indicates the master abort status when NBU is master.
28	RTA	0: No target abort 1: Target abort detected	This indicates the target abort status when NBU is master.
27	STA	0: No target abort 1: Target abort detected	This indicates the target abort status when NBU is target.
26, 25	DEV	01: Intermediate	This indicates the DEVDEL response timing. NBU is fixed to 01 (intermediate speed).
24	DPD	0: No PCI parity error 1: PCI parity error detected	This indicates the parity error status.
23	FBBC	1: Fixed	This indicates that fast back-to-back transfer is available. NBU is fixed to 1.
22	UCF	0: Not supported 1: Supported	This indicates whether user functions are supported.
21	66M	1: Fixed	This indicates that the system is operable at 66 MHz. NBU is fixed to 1.
20	PM	1: Fixed	This indicates whether power management is supported. NBU is fixed to 1.
19 to 10	–	0: Fixed	–
9	PBBE	0: Fixed	This sets whether to issue fast back-to-back when NBU is master. NBU is fixed to 0.
8	SER	0: Disabled 1: Enabled	This enables or disables SERR output.
7	WCC	0: Stepping prohibited 1: Stepping permitted	This permits or prohibits address data stepping.
6	PER	0: Response disabled 1: Response enabled	This enables or disables responses when a parity error occurs.

Table E-7 Detailed Data for I/O Errors Detected in Built-in Ethernet/ET.NET (EC=0x078013XX) (2/3)

<table>
<tr><th>DATn</th><th>Description</th><th>Detailed data</th></tr>
<tr><td>DAT0</td><td>NB_PCICONF1</td><td>
<table>
<tr><th>Bit no.</th><th>Bit name</th><th>Read-in value</th><th>Description</th></tr>
<tr><td>5</td><td>VPS</td><td>0: Fixed</td><td>This indicates that the device is not VGA-compatible.
NBU is fixed to 0.</td></tr>
<tr><td>4</td><td>MWIE</td><td>0: Fixed</td><td>This indicates that the memory write and invalidate command is not executed.
NBU is fixed to 0.</td></tr>
<tr><td>3</td><td>SPC</td><td>0: Fixed</td><td>This indicates that special cycle is not supported.
NBU is fixed to 0.</td></tr>
<tr><td>2</td><td>BUM</td><td>0: Master operation disabled
1: Master operation enabled</td><td>This enables or disables bus master operation.</td></tr>
<tr><td>1</td><td>MES</td><td>0: Access prohibited
1: Access permitted</td><td>This permits or prohibits access to memory space.</td></tr>
<tr><td>0</td><td>IOS</td><td>0: Access prohibited
1: Access permitted</td><td>This permits or prohibits access to I/O space.</td></tr>
</table>
</td></tr>
<tr><td>DAT1</td><td>ICR register</td><td>
<table>
<tr><th>Bit</th><th>15-10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>ICR</td><td>Rsv</td><td>MDAC</td><td>Rsv</td><td>RXT0</td><td>RXO</td><td>Rsv</td><td>RXDMT0</td><td>RXSEQ</td><td>LSC</td><td>TXQE</td><td>TXDW</td></tr>
</table>
(Bits 31 to 16 are reserved.)

ICR: Interrupt cause read register

Rsv: Unused
MDAC: MDI/O access complete
Rsv: Unused
RXT0: Reception timer interrupted
RXO: Reception overrun
Rsv: Unused
RXDMT0: Receive descriptor reached the minimum point
RXSEQ: Reception sequence error
LSC: Link status changed
TXQE: Transmission queue unused
TXDW: Transmission complete
</td></tr>
</table>

Table E-7 Detailed Data for I/O Errors Detected in Built-in Ethernet/ET.NET (EC=0x078013XX) (3/3)

DATn	Description	Detailed data
DAT2	Return value	Return value of the firmware reception utility when an error is detected
DAT3	Frame size	Reception frame size when an error is detected
DAT4	Status	Reception status error information when an error is detected
DAT5	Return value	Return value of the firmware transmission utility when an error is detected
DAT6	Unused	–
DAT7	Status	Transmission status information when an error is detected
DAT8	Station number	Station number of CPU or ET.NET
DAT9	Version number	Version number of CPU firmware or ET.NET firmware

DAT10 to DAT109 contain driver table information.

Table E-8 Detailed Data for I/O Errors Detected in a Driver (EC=0x078015XX)

DATn	Description	Detailed data
DAT0	Unused	
DAT1	Unused	
DAT2	Unused	
DAT3	IP address	IP address information setting, only when EC=0x07801512 (when a duplicate IP address is detected)
DAT4	MAC address 1	Information setting for the upper 4 bytes of a duplicate MAC address, when EC=0x07801512 (when a duplicate IP address is detected)
DAT5	MAC address 2	Information setting for the lower 2 bytes of a duplicate MAC address, when EC=0x07801512 (when a duplicate IP address is detected)
DAT6	Unused	
DAT7	Unused	
DAT8	Station number	Station number of CPU or ET.NET
DAT9	Version number	Version number of CPU firmware or ET.NET firmware

DAT10 to DAT109 contain driver table information.

(2) I/O error

This is an error detected during I/O transactions with a device. The details vary depending on the device.

Table E-9 Error Message Format

```
CPU xxxxxxxx
%CPMS-E-HARD-0004  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx I/O error (subtitle)
UNO =xxxxxxxx  DEV =xxxxxxxx  DVA =xxxxxxxx  IOEC =xxxxxxxx
TN  =xxxxxxxx
DAT0  =xxxxxxxx  DAT1  =xxxxxxxx  DAT2  =xxxxxxxx  DAT3  =xxxxxxxx  DAT4  =xxxxxxxx
DAT5  =xxxxxxxx  DAT6  =xxxxxxxx  DAT7  =xxxxxxxx  DAT8  =xxxxxxxx  DAT9  =xxxxxxxx
DAT10 =xxxxxxxx  DAT11 =xxxxxxxx  DAT12 =xxxxxxxx  DAT13 =xxxxxxxx  DAT14 =xxxxxxxx
DAT15 =xxxxxxxx  DAT16 =xxxxxxxx  DAT17 =xxxxxxxx  DAT18 =xxxxxxxx  DAT19 =xxxxxxxx
DAT20 =xxxxxxxx  DAT21 =xxxxxxxx  DAT22 =xxxxxxxx  DAT23 =xxxxxxxx  DAT24 =xxxxxxxx
DAT25 =xxxxxxxx  DAT26 =xxxxxxxx  DAT27 =xxxxxxxx  DAT28 =xxxxxxxx  DAT29 =xxxxxxxx
DAT30 =xxxxxxxx  DAT31 =xxxxxxxx  DAT32 =xxxxxxxx  DAT33 =xxxxxxxx  DAT34 =xxxxxxxx
DAT35 =xxxxxxxx  DAT36 =xxxxxxxx  DAT37 =xxxxxxxx  DAT38 =xxxxxxxx  DAT39 =xxxxxxxx
DAT40 =xxxxxxxx  DAT41 =xxxxxxxx  DAT42 =xxxxxxxx  DAT43 =xxxxxxxx  DAT44 =xxxxxxxx
DAT45 =xxxxxxxx  DAT46 =xxxxxxxx  DAT47 =xxxxxxxx  DAT48 =xxxxxxxx  DAT49 =xxxxxxxx
```

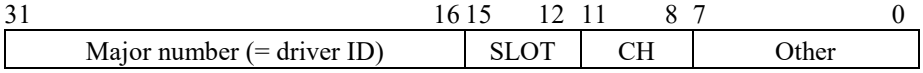
Item	Description
EC	Error code: This represents the error type. See Table E-10.
UNO	I/O unit number: This indicates a unit number.
DEV	Device number This indicates the type of I/O error and mount location of the device.  SLOT: Slot number; CH: Channel (interface) number; Other: Dependent on device driver
DVA	Device address This indicates the mount location of the device. This error is fixed to 0.
IOEC	I/O error code This indicates a detailed error code. Errors that do not have this data have an I/O error code of 0.
TN	Task number This indicates the task number of the task where an error occurred.
DAT*n*	Data *n* This indicates the error analysis data. See Table E-10 because the contents of this data vary depending on the error type.

Table E-10 Error Codes, Subtitles, and their Descriptions (I/O Error) (1/8)

No.	Error code	Subtitle	Description	Contents of DATn
1	EC=07395020	ROM (NANDF) Error	Accessing NAND flash memory resulted in an error. Try accessing the NAND flash memory again. If this error occurs often, replace the module.	For details about DATn, see Table E-11.
2	EC=50010100	OD.RING Module switch setting error	The setting of the MODU No. setting switch of the OD.RING module is incorrect. Check and, if necessary, correct the setting of the MODU No. setting switch.	For details about DATn, see Table E-13.
3	EC=50010101	OD.RING CPL switch setting error	The setting of the CPL No. setting switch of the OD.RING module is incorrect. Check and, if necessary, correct the setting of the CPL No. setting switch.	For details about DATn, see Table E-13.
4	EC=50010111	OD.RING Duplicate CPL No.	The setting of the CPL No. setting switch of the OD.RING module is the same as that of another OD.RING module connected to the same ring. Check and, if necessary, correct the setting of the CPL No. setting switch.	For details about DATn, see Table E-13.
5	EC=50010112	OD.RING Parameter type Mismatch/SUM err	The OD.RING module might have one or more incorrect parameters. Reconfigure the parameters. If this does not resolve the problem, the OD RING module might have failed. In this case, replace the OD.RING module.	For details about DATn, see Table E-13.

Table E-10 Error Codes, Subtitles, and their Descriptions (I/O Error) (2/8)

No.	Error code	Subtitle	Description	Contents of DATn
6	EC=5002010B	FL.NET Parameter type Mismatch/SUM error	The FL.NET module might have one or more incorrect parameters. Reconfigure the parameters. If this does not resolve the problem, the FL.NET module might have failed. In this case, replace the FL.NET module.	For details about DATn, see Table E-13.
7	EC=50020113	FL.NET IP address not registered	No IP address is registered for the FL.NET module. Register an IP address.	For details about DATn, see Table E-13.
8	EC=50020200	FL.NET NetWK participation not completed	The FL.NET module has not joined the network. (The FL.NET module is currently in the process of joining the network.)	For details about DATn, see Table E-13.
9	EC=50020201	FL.NET Duplicate common memory settings	The local node has the same common memory setting as another node. Compare the common memory settings of the local node with those of other nodes, and correct any duplicate setting. The common memory settings (area address and word count) of nodes containing duplicate common memory settings become 0.	For details about DATn, see Table E-13.
10	EC=50020202	FL.NET Duplicate node numbers	The node number set for the FL.NET module is already being used by another node on the network. Check the node number of other node, and set a different node number to avoid duplication. (According to the FA link protocol, any participating nodes with duplicate node numbers are forced to leave the network. To rejoin the network, power off and then power on the module, or send a join request from FL.NET SYSTEM/S10VE.)	For details about DATn, see Table E-13.
11	EC=50020203	FL.NET module setting error	The FL.NET module has one or more abnormal settings. From FL.NET SYSTEM/S10VE, check the settings, and correct any abnormal settings you find. If the error persists, the FL.NET module might have failed. In this case, replace the FL.NET module.	For details about DATn, see Table E-13.

Table E-10 Error Codes, Subtitles, and their Descriptions (I/O Error) (3/8)

No.	Error code	Subtitle	Description	Contents of DATn
12	EC=50020204	FL.NET Token hold timeout	The specified token hold time was exceeded three or more times in a row. The FL.NET module might have failed. Replace the FL.NET module. (According to the FA link protocol, any participating nodes where a token hold timeout occurred are forced to leave the network.)	For details about DATn, see Table E-13.
13	EC=50027310	FL.NET I/O CARRIER LOSS	A carrier loss error occurred on the transmission line. Check the transmission line.	For details about DATn, see Table E-13.
14	EC=50027311	FL.NET I/O RETRY	A retry error occurred on the transmission line. Check the transmission line.	For details about DATn, see Table E-13.
15	EC=50027312	FL.NET I/O LATE	A late collision occurred on the transmission line. Check the transmission line.	For details about DATn, see Table E-13.
16	EC=50027351	FL.NET I/O TX_ABORT	Abnormal transmission occurred on the transmission line. Check the transmission line.	For details about DATn, see Table E-13.
17	EC=50027353	FL.NET I/O TX_DEFER	A transmission error occurred due to a transmission delay on the transmission line. Check the transmission line.	For details about DATn, see Table E-13.
18	EC=50027375	FL.NET I/O RX_STAT_OVER	An overrun error occurred in the receive status FIFO. Check and, if necessary, correct the load on the line.	For details about DATn, see Table E-13.
19	EC=50027376	FL.NET I/O TX_DATA_UNDER	An underrun error occurred in the transmission data FIFO. Check and, if necessary, correct the load on the line.	For details about DATn, see Table E-13.
20	EC=50027377	FL.NET I/O RX_DATA_OVER	An overrun error occurred in the receive data FIFO. Check and, if necessary, correct the load on the line.	For details about DATn, see Table E-13.
21	EC=50027508	FL.NET I/O BUF_OVF	An overflow occurred in the transmission/reception management buffer. Check and, if necessary, correct the load on the line.	For details about DATn, see Table E-13.
22	EC=5002750F	FL.NET I/O SOCKET_OVF	An overflow occurred in the socket management buffer. If this error occurs, replace the FL.NET module.	For details about DATn, see Table E-13.
23	EC=50027512	FL.NET I/O IPADDR_DUPL	The FL.NET module has the same IP address as another device on the network. Check the IP address setting.	For details about DATn, see Table E-13.
24	EC=50027D10	FL.NET INVALID MAIN/SUB SWITCH SETTING	The MAIN/SUB setting switch has one or more incorrect settings. Check and, if necessary, correct the settings for the MAIN/SUB setting switch.	For details about DATn, see Table E-13.
25	EC=50027D12	FL.NET MAIN/SUB SW SETTING DUPLICATION	The MAIN/SUB setting switch has the same setting as another FL.NET module mounted on the same mount base. Check and, if necessary, correct the setting of the MAIN/SUB setting switch.	For details about DATn, see Table E-13.

Table E-10 Error Codes, Subtitles, and their Descriptions (I/O Error) (4/8)

No.	Error code	Subtitle	Description	Contents of DATn
26	EC=50030100	J.NET Module switch setting error	The MODU No. setting switch has one or more incorrect settings. Check and, if necessary, correct the settings of the MODU No. setting switch.	For details about DATn, see Table E-13.
27	EC=50030101	J.NET Baud rate switch setting error	The BIT RATE setting switch has one or more incorrect settings. Check, and if necessary, correct the settings of the BIT RATE setting switch.	For details about DATn, see Table E-13.
28	EC=50030112	J.NET Parameter type Mismatch/SUM error	The J.NET module might have one or more incorrect parameters. Reconfigure the parameters. If this does not resolve the problem, the J.NET module might have failed. In this case, replace the J.NET module.	For details about DATn, see Table E-13.
29	EC=50032010	J.NET CRC error	• Check whether the network line is in normal condition. • Check whether the parameter and station settings are correct. • If the error persists, replace the J.NET module.	For details about DATn, see Table E-13.
30	EC=50032020	J.NET Station No. error		For details about DATn, see Table E-13.
31	EC=50032030	J.NET Undefined service operated		For details about DATn, see Table E-13.
33	EC=50032040	J.NET I / UI - frame length error		For details about DATn, see Table E-13.
34	EC=50032041	J.NET I-frame format error (non Exist)		For details about DATn, see Table E-13.
35	EC=50032042	J.NET I-frame format error (Exist)		For details about DATn, see Table E-13.
36	EC=50032050	J.NET Data link sequence error		For details about DATn, see Table E-13.
37	EC=50032060	J.NET Slave response Timeout error	• Power off and then power on the station. • Check whether the switch settings are correct for the J.NET module and station. • If the error persists, replace the station.	For details about DATn, see Table E-13.
38	EC=50032061	J.NET recover not successful		For details about DATn, see Table E-13.

Table E-10 Error Codes, Subtitles, and their Descriptions (I/O Error) (5/8)

No.	Error code	Subtitle	Description	Contents of DATn
39	EC=50032070	J.NET Transmit /Receive error	• Check the connections for the network line and terminating register. • Check whether the parameter and station settings are correct. • Reset and restore the CPU. If the error persists, restart the system. • If the error persists, replace the J.NET module.	For details about DATn, see Table E-13.
40	EC=50032080	J.NET error occurred (.etc)	• Reset and restore the CPU. If the error persists, restart the system. • If the error persists, replace the module.	For details about DATn, see Table E-13.
41	EC=50037061	J.NET Waiting Input data	This does not indicate an error. The status returns to normal when loading of the input data is complete.	For details about DATn, see Table E-13.
42	EC=50037110	J.NET Undefined service operated	• Reset and restore the CPU. If the error persists, restart the system. • If the error persists, replace the module.	For details about DATn, see Table E-13.
42	EC=50037120	J.NET Transmission data length error		For details about DATn, see Table E-13.
43	EC=50037130	J.NET Transmission packet error		For details about DATn, see Table E-13.
44	EC=50038020	J.NET Initialize refused	• The parameter and station settings do not match. Reconfigure the parameters so that they match the station settings. • If the error persists, replace the station.	For details about DATn, see Table E-13.
45	EC=50038081	J.NET SVPT TX Bytes unmatched (Auto mode)		For details about DATn, see Table E-13.
46	EC=50038082	J.NET SVPT TX Bytes unmatched (Slot)		For details about DATn, see Table E-13.
47	EC=50039001	J.NET Station stopped	• Power off and then power on the station to reset the CPU. • If the error persists, replace the station.	For details about DATn, see Table E-13.
48	EC=50039002	J.NET Station error status detected		For details about DATn, see Table E-13.
49	EC=50039003	J.NET St.err status detected and Stopped		For details about DATn, see Table E-13.

Table E-10 Error Codes, Subtitles, and their Descriptions (I/O Error) (6/8)

No.	Error code	Subtitle	Description	Contents of DATn
50	EC=5003A020	J.NET PUT/GET (Insufficient address data)	Check and, if necessary, correct the PUT/GET service request sent from the station.	For details about DATn, see Table E-13.
51	EC=5003A021	J.NET PUT/GET (addr field number illegal)		For details about DATn, see Table E-13.
52	EC=5003A022	J.NET PUT/GET (addr field format error)		For details about DATn, see Table E-13.
53	EC=5003A040	J.NET PUT/GET (Slot setting)		For details about DATn, see Table E-13.

Table E-10 Error Codes, Subtitles, and their Descriptions (I/O Error) (7/8)

No.	Error code	Subtitle	Description	Contents of DATn
54	EC=5004140A	D.NET Invalid MODU No. switch setting	The MODU No. setting switch has one or more incorrect settings. Check and, if necessary, correct the settings of the MODU No. setting switch.	For details about DATn, see Table E-13.
55	EC=50045188	D.NET TX data size setting error	The setting of the transmission word count is incorrect. Check the parameter setting, and set it again.	For details about DATn, see Table E-13.
56	EC=50045189	D.NET Parameter type Mismatch/SUM error	The D.NET module might have one or more incorrect parameters. Reconfigure the parameters. If this does not resolve the problem, the D.NET module might have failed. In this case, replace the D.NET module.	For details about DATn, see Table E-13.
57	EC=50047082	D.NET Recover from Transmission Bus Off	This does not indicate an error. This message notifies you of recovery of the transmission line from the bus off status.	For details about DATn, see Table E-13.
58	EC=50047381	D.NET Transmission Bus Off	The transmission line is in bus off status. Check for loose connectors, and check the cable wiring, transmission speed, MODU No. setting switch setting, and MAC ID setting (set from D.NET SYSTEM/S10VE).	For details about DATn, see Table E-13.
59	EC=50048181	D.NET CAN Transmission Timeout Error.	Check for loose connectors, and check the cable wiring, transmission speed, MODU No. setting switch setting, and MAC ID setting (set from D.NET SYSTEM/S10VE). Note that although the *CAN transmission timeout error* also occurs in the following cases, the status of the D.NET module remains normal: • When the communication connector of the D.NET module is disconnected • When the other station does not exist or is turned off • When the transmission speed of the other station does not match that of the local station The *CAN transmission timeout error* does not occur if a station other than the D.NET module exists on the network. (This error does not occur even if the station does not communicate directly with the D.NET module.)	For details about DATn, see Table E-13.

Table E-10 Error Codes, Subtitles, and their Descriptions (I/O Error) (8/8)

No.	Error code	Subtitle	Description	Contents of DATn
60	EC=500E7510	ET.NET IFCONFIG_UP	The ET.NET module might have one or more incorrect parameters. Set the parameters again. If this does not resolve the problem, the ET.NET module might have failed. In this case, replace the ET.NET module.	For details about DATn, see Table E-13.
61	EC=500E7511	ET.NET NETADDR_DUPL	The ET.NET module might have one or more incorrect parameters. Set the parameters again. If this does not resolve the problem, the ET.NET module might have failed. In this case, replace the ET.NET module.	For details about DATn, see Table E-13.
62	EC=500E7512	ET.NET IPADDR_DUPL	The ET.NET module has the same IP address as another device on the network. Check the IP address setting.	For details about DATn, see Table E-13.
63	EC=500E7D12	ET.NET Invalid MAIN/SUB switch setting Duplication	The MODU No. setting switch has the same setting as another ET.NET module mounted on the same mount base. Check and, if necessary, correct the setting of the MODU. No. setting switch.	For details about DATn, see Table E-13.
64	EC=500E7D1A	ET.NET Invalid MAIN/SUB switch setting	The MODU No. setting switch has the incorrect setting. Check and, if necessary, correct the setting of the MODU. No. setting switch.	For details about DATn, see Table E-13.
65	EC=500E7D1B	ET.NET Invalid ST. No. switch setting	The ST.No. setting switch has the incorrect setting. Check and, if necessary, correct the setting of the ST.No. setting switch.	For details about DATn, see Table E-13.
66	EC=500E7D1C	ET.NET Invalid network setting	The communication settings of the ET.NET module have not been configured. Configure the communication settings.	For details about DATn, see Table E-13.

Table E-11 Detailed Data About the ROM (NANDF) Error (I/O Error)

<table>
<tr><th>DATn</th><th>Description</th><th>Detailed data</th></tr>
<tr><td>DAT0</td><td>Status</td><td>Process that caused the error
<table>
<tr><th>Status</th><th>Description</th></tr>
<tr><td>0x1011</td><td>Abnormal ROM ACCESS utility (romread)</td></tr>
<tr><td>0x1012</td><td>Abnormal ROM WORK utility, or access timeout (romread)</td></tr>
<tr><td>0x1021</td><td>Abnormal ROM ACCESS utility (romwrite)</td></tr>
<tr><td>0x1022</td><td>Abnormal ROM WORK utility, or access timeout (romwrite)</td></tr>
<tr><td>0x1031</td><td>Abnormal ROM ACCESS utility (cfread)</td></tr>
<tr><td>0x1032</td><td>Abnormal ROM WORK utility, or access timeout (cfread)</td></tr>
<tr><td>0x1041</td><td>Abnormal ROM ACCESS utility (cfwrite)</td></tr>
<tr><td>0x1042</td><td>Abnormal ROM WORK utility, or access timeout (cfwrite)</td></tr>
</table></td></tr>
<tr><td>DAT1</td><td>Access type</td><td>Firmware error information (for access type)
<table>
<tr><th>Error code</th><th>Description</th></tr>
<tr><td>0x00A0</td><td>Read</td></tr>
<tr><td>0x00A1</td><td>Write</td></tr>
</table></td></tr>
<tr><td>DAT2</td><td>Starting sector</td><td>Firmware error information (starting sector)
Starting sector when a firmware utility is called</td></tr>
<tr><td>DAT3</td><td>Number of sectors</td><td>Firmware error information (number of sectors)
Number of sectors when a firmware utility is called</td></tr>
<tr><td>DAT4</td><td>Main memory address</td><td>Firmware error information (main memory address)
Main memory address when a firmware utility is called</td></tr>
<tr><td>DAT5
|
DAT8</td><td>Detailed error information</td><td>See Table E-12 DAT5 to DAT8 Firmware Error Information (Detailed Error Data) of the ROM (NANDF) Error.</td></tr>
</table>

Table E-12 DAT5 to DAT8 Firmware Error Information (Detailed Error Data) of the ROM (NANDF) Error (1/2)

No.	Item	Trace code (DAT5)	DAT6	DAT7	DAT8	Description
1	Block management error	0x000000B0	Block number of assignment table	Block number of status table	0x00000000	Error caused by block deletion request
2		0x000000B1	Block number of assignment table	Block number of status table	0x00000000	Error caused by execution of block deletion
3		0x000000B2	Block number of assignment table	0x00000000	0x00000000	Error caused by new block assignment
4		0x000000B3	Block number of assignment table	Block number of status table	0x00000000	Error caused by unassigned block
5	Table management error	0x000000C0	0x00000001	Status of management table A	Status of management table B	Error caused by table selection
6			0x00000002	Access number of management table A	Access number of management table B	Error caused by table selection
7			0x00000003	0x00000000	0x00000000	Error caused by reference table selection
8		0x000000C1	0x00000001	Block number of assignment table	Usage status of block	Table mismatch 1 (comparison mismatch in assignment table)
9			0x00000002	Block number of assignment table	Block number of status table	Table mismatch 2 (specified block out of range)
10			0x00000003	Block number of assignment table	Usage status of block	Table mismatch 3 (comparison mismatch in status table)
11			0x00000004	Block number of status table	Usage status of block	Table mismatch 4 (unused)
12		0x000000C2	0x00000001	0x00000000	0x00000000	Error caused by DMA request managed by table
13			0x00000002	0x00000000	0x00000000	Error caused by DMA execution managed by table
14		0x000000C3	Block number	0x00000000	0x00000000	Error caused by BAD block check request
15		0x000000C4	Block number	0x00000000	0x00000000	Error caused by execution of BAD block check

Table E-12 DAT5 to DAT8 Firmware Error Information (Detailed Error Data) of the ROM (NANDF) Error (2/2)

No.	Item	Trace code (DAT5)	DAT6	DAT7	DAT8	Description
16	Read error	0x000000D0	Block number of status table	Page number	Serious fault factor or error factor	Error caused by request to read from NAND flash memory
17		0x000000D1	Block number of status table	Page number	Serious fault factor or error factor	Error caused by read execution from NAND flash memory
18		0x000000D2	Starting sector number in the page	Number of sectors in the page	Starting address of block buffer	Error caused by request to read from NASC1 buffer
19		0x000000D3	Starting sector number in the page	Number of sectors in the page	Starting address of block buffer	Error caused by request to execute read from NASC1 buffer
20		0x000000D4	Block number of status table	Process page number	Serious fault factor or error factor	RS uncorrectable
21	Write error	0x000000E0	Starting sector number in the page	Number of sectors in the page	Starting address of block buffer	Error caused by request to write to NASC1 buffer
22		0x000000E1	Starting sector number in the page	Number of sectors in the page	Starting address of block buffer	Error caused by write execution to NASC1 buffer
23		0x000000E2	Block number of assignment table	Block number of status table	Page number in the block	Error caused by request to write to NAND flash memory
24		0x000000E3	Block number of assignment table	Block number of status table	Page number in the block	Error caused by write execution to NAND flash memory
25	Other error	0x000000F0	0x00000000	0x00000000	0x00000000	Program error
26		0x000000F1	0x00000000	0x00000000	0x00000000	Abnormal number of blocks

Table E-13 Detailed Data of the Option Module Error (I/O Error)

<table>
<tr><th>DATn</th><th>Description</th><th>Detailed data</th></tr>
<tr><td>DAT0</td><td>Error code</td><td>Error code represented by a four-digit hexadecimal number output from an optional module</td></tr>
<tr><td>DAT1</td><td>Module no.</td><td>Module no.
<table>
<tr><th>Module no.</th><th>Description</th></tr>
<tr><td>0x0000</td><td>Module 0/channel 0</td></tr>
<tr><td>0x0001</td><td>Module 1/channel 1</td></tr>
<tr><td>0x0002</td><td>Module 2/channel 2</td></tr>
<tr><td>0x0003</td><td>Module 3/channel 3</td></tr>
<tr><td>0x0011</td><td>Main module</td></tr>
<tr><td>0x0012</td><td>Submodule</td></tr>
</table></td></tr>
<tr><td>DAT2</td><td>Time when error occurred</td><td>Time when the error occurred (seconds)</td></tr>
<tr><td>DAT3</td><td>Time when error occurred</td><td>Time when the error occurred (minutes)</td></tr>
<tr><td>DAT4</td><td>Time when error occurred</td><td>Time when the error occurred (hours)</td></tr>
<tr><td>DAT5</td><td>Time when error occurred</td><td>Time when the error occurred (date)</td></tr>
<tr><td>DAT6</td><td>Time when error occurred</td><td>Time when the error occurred (month)</td></tr>
<tr><td>DAT7</td><td>Time when error occurred</td><td>Time when the error occurred (year)</td></tr>
<tr><td>DAT8</td><td>Channel no.</td><td>Channel no. (set for D.NET and ET.NET)
<table>
<tr><th>Module no.</th><th>Description</th></tr>
<tr><td>0x0000</td><td>Channel 0</td></tr>
<tr><td>0x0001</td><td>Channel 1</td></tr>
</table></td></tr>
</table>

E.3.4 Watchdog timer timeout error

This error indicates that the watchdog timer has expired. This means that the user task that is supposed to periodically update the watchdog timer did not operate for some reason, resulting in the set time not being updated in time.
When this error occurs, a link is made to the built-in subroutine WDTES so that you can register the appropriate processing for each individual user.
Table E-14 shows and describes the error message.

Table E-14 Format of the Watchdog Timer Timeout Error Message

```
CPU xxxxxxxx
%CPMS-E-SOFT-0005    SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=05c70000 WDT timeout error
TIME =xxxxxxxx
```

Item	Description
EC	Error code: This indicates the error type (fixed to 05c70000).
TIME	WDT set time: This indicates the monitoring time that expired. The unit is milliseconds.

E.3.5 Module Error

This error indicates that an abnormality was detected in the hardware of a module.
Table E-15 shows and describes the error message.

Table E-15 Format of the Module Error Message

```
CPU xxxxxxxx
%CPMS-x-HARD-0006  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx Module Error (subtitle)
SLOT =xxxxxxxx  MSW0 =xxxxxxxx  MSW1 =xxxxxxxx
DAT0  =xxxxxxxx DAT1  =xxxxxxxx DAT2  =xxxxxxxx DAT3  =xxxxxxxx DAT4  =xxxxxxxx
DAT5  =xxxxxxxx DAT6  =xxxxxxxx DAT7  =xxxxxxxx
```

(Can be displayed up to DAT111)

Item	Description
EC	Error code: This indicates the error type. See Table E-16.
SLOT	Slot number: This indicates the slot number of the module where the error was detected.
MSW0	Module status word 0: This is one of registers indicating the module status. See Table E-16 because the contents of this register vary depending on the module.
MSW1	Module status word 1: This is one of registers indicating the module status. See Table E-16 for the contents of the register that vary depending on the module.
DAT*n*	Data *n*: Error analysis data See Table E-16 for the contents of the data that vary depending on the error type.

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (1/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
1	EC=03820001	Memory Error (MRAM)	An abnormal diagnostic error was detected in the MRAM.	DAT0: Execution result of the utility For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17.
2	EC=03B60000	RI/O-IF Module Error	An RI/O-IF module error was detected.	For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17. For details about DAT*n*, see Table E-18.
3	EC=03B90000	PCI Bus Error	A serious fault interruption was detected in the PCI bus system.	For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17. For details about DAT*n*, see Table E-25.
4	EC=03BD0000	LSI Internal Timeout Error	An LSI internal timeout was detected.	For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17. For details about DAT*n*, see Table E-19.
5	EC=03BE0000	SPU Error	An SPU error was detected.	For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17. For details about DAT*n*, see Table E-20.
6	EC=03BF0000	RI/O Error	An RI/O-IF RI/O error was detected.	For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17. For details about DAT*n*, see Table E-21.

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (2/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
7	EC=03E00000	System task error (Table not found)	An address fetch from a table failed.	For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17. DAT0: Task number DAT1: Table type `0`: IRGLB `1`: IRSUB DAT: IRGLB/IRSUB number
8	EC=03E00001	System task error (Task queue failed)	Task startup failed.	For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17. DAT0: Number of started task DAT1: Return value of `queue`
9	EC=05000000	Invalid Interrupt	An invalid interrupt was detected.	• For DIF(PI/O) For the contents of MSW0 and MSW1, see the description of *DIF* in Table E-20. • For anything other than DIF(PI/O) The contents of MSW0, MSW, and DATA0 were invalid.
10	EC=05000001	Undefined Interrupt	Interruption processing accepted an undefined interrupt code.	DAT0: Interrupt code when the interrupt occurred (INTEVT) The contents of MSW0 and MSW1 were invalid.
11	EC=05000002	INTEVT Invalid Interrupt	An invalid interrupt was accepted.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (3/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
12	EC=05001011	RI/O INTR Invalid Interrupt	An invalid RI/O-IF RI/O interrupt was detected.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0xFFFFFFFF MSW1: Fixed to 0xFFFFFFFF DAT0: Interrupt event code
13	EC=05003001	LV3 INTST Invalid Interrupt	An invalid LV3 interrupt was detected.	
14	EC=05003002	RQI6 INF Invalid Interrupt	An invalid RQI6 interrupt was detected.	
15	EC=05004001	RINTR Invalid Interrupt	The module that reported the I/O interrupt does not exist.	DAT0: Interrupt code when the interrupt occurred (INTEVT) The contents of MSW0 and MSW1 are invalid.
16	EC=05006001	SPU INTR Invalid Interrupt	An invalid SPU interrupt was detected.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0xFFFFFFFF MSW1: Fixed to 0xFFFFFFFF DAT0: Interrupt event code
17	EC=0500A001	NINTR Invalid Interrupt	The module that reported the transmission interrupt does not exist.	DAT0: Interrupt code when the interrupt occurred (INTEVT) The contents of MSW0 and MSW1 are invalid.
18	EC=0500B001	PUINTR Invalid Interrupt	The module that reported the interrupt between PUs does not exist.	
19	EC=0500F001	HERST Invalid Interrupt	A serious fault interrupt was detected, but the factor could not be identified. (Before master judgment)	
20	EC=0500F002	HERST Invalid Interrupt(2)	A serious fault interrupt was detected, but the factor could not be identified. (At master judgment)	
21	EC=0500F003	BUERRSTAT Invalid Interrupt	Although a serious fault interrupt was detected in the PCI bus system, a detailed factor was not recorded.	
22	EC=0500F004	P2NHERRQ Invalid Interrupt	Although the HP received a serious fault interrupt report from the CP, the CP did not request a serious fault interrupt report.	
23	EC=0500F005	N2PHERRQ Invalid Interrupt	Although the CP received a serious fault interrupt report from the HP, the HP did not request a serious fault interrupt report.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (4/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
24	EC=0500F006	MHPMCLG Invalid Interrupt	Although a serious fault interrupt was detected in the memory system, a detailed factor was not recorded.	DAT0: Interrupt code when the interrupt occurred (INTEVT) The contents of MSW0 and MSW1 are invalid.
25	EC=0500F007	ECC 2bit Master Invalid Interrupt	Although a serious fault interrupt report was detected in the memory system, the master could not be identified.	
26	EC=0500F008	RERRMST Invalid Interrupt	Although a serious fault interrupt was detected in the system bus system, the master information was not recorded.	
27	EC=0500F009	Invalid P2NHERR Interrupt (CP Alive)	Although the HP received a serious fault report from the CP, the CP did not go down.	
28	EC=0500F00B	NP_ERRLOGMP Invalid Interrupt	An invalid interrupt of an NPU serious fault interrupt was detected.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0xFFFFFFFF MSW1: Fixed to 0xFFFFFFFF DAT0: Interrupt event code
29	EC=0500F00C	SPU HERR Invalid Interrupt	An invalid interrupt of an SPU serious fault interrupt was detected.	
30	EC=0500F00D	RIO HERR Invalid Interrupt	An invalid interrupt of an RI/O-IF RI/O serious fault interrupt was detected.	
31	EC=05110000	Macro Parameter Error	An error was detected when an rleas or queue macro was issued to a task on another PU.	DAT0: Error factor DAT1: Request DAT2: Task number to which the macro was issued DAT3: Startup factor of the task to which the macro was issued • Error factor −1: The logical slot number of the task to be started was abnormal. 2: The other PU is down. 4: The other PU does not exist. The contents of MSW0 and MSW1 were invalid.

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (5/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
32	EC=0739D001	RQI6 Interrupt Received	An RQI6 interrupt error occurred. (A parity error occurred when the memory of an optional module was accessed.)	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0xFFFFFFFF MSW1: Fixed to 0xFFFFFFFF DAT0: Module ID code 0x0001: OD.RING 0x0002: FL.NET 0x0003: J.NET 0x0004: D.NET DAT1: I/F register address DAT2: NS_INTST register value DAT3: MSW register value DAT4: ISW6 register value DAT5: Number of optional modules that detected an RQI6 interrupt error in one interrupt DAT6: Number of optional modules that detected an RQI6 interrupt factor clear error in one interrupt DAT7: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule
33	EC=0739D002	RQI6 Interrupt Factor (ISW6) Clear Error	An RQI6 interrupt factor clear error occurred.	
34	EC=0D010000	Memory Alarm	A memory 1-bit error was detected five times in a row.	For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17. For details about DAT*n*, see Table E-23.
35	EC=0D010001	Memory Patrol Error	An error was detected during patrol scrubbing.	For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17. For details about DAT*n*, see Table E-22.

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (6/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
36	EC=0D300010	Primary Battery Error	A primary battery error was detected.	For the contents of MSW0 and MSW1, see the description of *CPU* in Table E-17. For details about DATn, see Table E-24.
37	EC=0D320000	Memory Error	A built-in processor detected an unrecoverable error (2-bit error) when accessing built-in memory.	DAT0: MSW2 of the module where the error occurred For the contents of MSW0, MSW1, and MSW2 of the module where the error occurred, see the description of the module in Table E-17.

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (7/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
38	EC=0D330000	Hardware WDT Timeout	A hardware watchdog timer timeout was detected.	DAT0: MSW2 of the module where the error occurred For the contents of MSW0, MSW1, and MSW2 of the module where the error occurred, see the description of the module in Table E-17.
39	EC=0D340000	Software WDT Timeout	A software watchdog timer timeout was detected.	DAT0: MSW2 of the module where the error occurred For the contents of MSW0, MSW1, and MSW2 of the module where the error occurred, see the description of the module in Table E-17.
40	EC=0D350000	RAM Sum Check Error	A RAM checksum error was detected.	DAT0: MSW2 of the module where the error occurred For the contents of MSW0, MSW1, and MSW2 of the module where the error occurred, see the description of the module in Table E-17.
41	EC=0D360000	ROM Sum Check Error	A ROM checksum error was detected.	DAT0: MSW2 of the module where the error occurred For the contents of MSW0, MSW1, and MSW2 of the module where the error occurred, see the description of the module in Table E-17.
42	EC=0D370000	External Error	A hardware error other than a battery error, watchdog timer timeout, checksum error, or clock stop was detected. (Logical sum of HWDT and CLKSTP for CPU)	DAT0: MSW2 of the module where the error occurred For the contents of MSW0, MSW1, and MSW2 of the module where the error occurred, see the description of the module in Table E-17.
43	EC=0D380000	OS Clear Error	The OS was cleared, or the OS has not been loaded.	DAT0: MSW2 of the module where the error occurred For the contents of MSW0, MSW1, and MSW2 of the module where the error occurred, see the description of the module in Table E-17.
44	EC=0D390000	Clock Stop Error	A hardware error (clock stop) was detected.	DAT0: MSW2 of the module where the error occurred For the contents of MSW0, MSW1, and MSW2 of the module where the error occurred, see the description of the module in Table E-17.

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (8/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
45	EC=0D800000	TOD Error	The data read from TOD was irrational.	DAT0: Year (last two digits) DAT1: Month DAT2: Date DAT3: Hours DAT4: Minutes DAT5: Seconds DAT6: HKP utility return value DAT7: Year (utility interface area) DAT8: Month (utility interface area)) DAT9: Date (utility interface area) DAT10: Hours (utility interface area) DAT11: Minutes (utility interface area) DAT12: Seconds (utility interface area) For the contents of MSW0 and MSW1, see description of *CPU* in Table E-17.

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (9/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
46	EC=50010010	OD.RING Bus error	A bus error occurred in the OD.RING module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when an error occurred (year)
47	EC=50010011	OD.RING Invalid address	An address error occurred in the OD.RING module.	
48	EC=50010012	OD.RING Invalid instruction	An invalid instruction error occurred in the OD.RING module.	
49	EC=50010013	OD.RING Division by zero	A division by zero error occurred in the OD.RING module.	
50	EC=50010014	OD.RING Privilege violation	A privilege violation error occurred in the OD.RING module.	
51	EC=50010015	OD.RING WDT timeout error	A watchdog timer timeout error occurred in the OD.RING module.	
52	EC=50010016	OD.RING Format error	A format error occurred in the OD.RING module.	
53	EC=50010017	OD.RING Spurious Interrupt	A spurious interrupt occurred in the OD.RING module.	
54	EC=50010018	OD.RING Unused exception	An unused exception occurred in the OD.RING module.	
55	EC=50010019	OD.RING Parity error	A parity error occurred in the OD.RING module.	
56	EC=5001001A	OD.RING Prepare for Grand Reset	A GR-warning occurred in the OD.RING module.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (10/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
57	EC=50010102	OD.RING ROM1 checksum error	A ROM1 checksum error occurred in the OD.RING module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year)
58	EC=50010103	OD.RING RAM1 compare error	A RAM1 comparison error occurred in the OD.RING module.	
59	EC=50010105	OD.RING RAM2 compare error	A RAM2 comparison error occurred in the OD.RING module.	
60	EC=5001010B	OD.RING ROM3 checksum error	A ROM3 checksum error occurred in the OD.RING module.	
61	EC=5001010C	OD.RING ROM erasing error (program)	A ROM (program) erase error occurred in the OD.RING module.	
62	EC=5001010D	OD.RING ROM writing error (program)	A ROM (program) write error occurred in the OD.RING module.	
63	EC=5001010E	OD.RING ROM erasing error (parameter)	A ROM (parameter) erase error occurred in the OD.RING module.	
64	EC=5001010F	OD.RING ROM writing error (parameter)	A ROM (parameter) write error occurred in the OD.RING module.	
65	EC=50010110	OD.RING ROM writing over 50000 times	The maximum number of ROM writes in the OD.RING module was exceeded.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (11/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
66	EC=50020114	FL.NET MAC address not registered	A MAC address unregistered error occurred in the FL.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year)
67	EC=50023031	FL.NET Inst. Alignment Error	An instruction alignment error occurred in the FL.NET module.	
68	EC=50023041	FL.NET Illegal Instruction	An illegal instruction error occurred in the FL.NET module.	
69	EC=50023081	FL.NET Privileged Instruction	A privileged instruction error occurred in the FL.NET module.	
70	EC=500230F9	FL.NET Illegal Exception	An illegal exception error occurred in the FL.NET module.	
71	EC=50023389	FL.NET FP Unavailable	A floating-point unavailable exception error occurred in the FL.NET module.	
72	EC=50023391	FL.NET FP Program Error	A floating-point error occurred in the FL.NET module.	
73	EC=50023401	FL.NET Instruction Page Fault	An instruction access page fault occurred in the FL.NET module.	
74	EC=50023421	FL.NET Invalid Inst. Access	An instruction access error occurred in the FL.NET module.	
75	EC=50023461	FL.NET Inst. Access Protection	An instruction access protection error occurred in the FL.NET module.	
76	EC=50023471	FL.NET Data Alignment Error	A data alignment error occurred in the FL.NET module.	
77	EC=50023601	FL.NET Data Page Fault	A data access page fault error occurred in the FL.NET module.	
78	EC=50023621	FL.NET Invalid Data Access	A data access error occurred in the FL.NET module.	
79	EC=50023661	FL.NET Data Access Protection	A data access protection error occurred in the FL.NET module.	
80	EC=50023820	FL.NET Memory Error	A memory error occurred in the FL.NET module.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (12/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
81	EC=500238A0	FL.NET Memory Access Error	A memory access error occurred in the FL.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year)
82	EC=500238B0	FL.NET Internal Bus Parity	An internal bus parity error occurred in the FL.NET module.	
83	EC=500238C0	FL.NET System Bus Parity	A system bus parity error occurred in the FL.NET module.	
84	EC=500238F0	FL.NET Undefined Machine Check	An undefined machine check error occurred in the FL.NET module.	
85	EC=50023B70	FL.NET Bus Target Abort	A bus target abort error occurred in the FL.NET module.	
86	EC=50025000	FL.NET Invalid Interrupt	An invalid interrupt error occurred in the FL.NET module.	
87	EC=50025001	FL.NET Undefined Invalid Interrupt	An undefined invalid interrupt error occurred in the FL.NET module.	
88	EC=50025002	FL.NET INTEVT Invalid Interrupt	An INTEVT invalid interrupt occurred in the FL.NET module.	
89	EC=50025011	FL.NET RQI3 INT Invalid Interrupt	A RQI3 invalid station error occurred in the FL.NET module.	
90	EC=50025012	FL.NET RQI3 Link Invalid Interrupt	A RQI3 invalid link status error occurred in the FL.NET module.	
91	EC=50025013	FL.NET RQI3 Module Invalid Interrupt	A RQI3 invalid module status error occurred in the FL.NET module.	
92	EC=50025031	FL.NET LV3 INTST Invalid Interrupt	A level 3 invalid interrupt status error occurred in the FL.NET module.	
93	EC=50025032	FL.NET RQI6 INF Invalid Interrupt	An RQI6 invalid status error occurred in the FL.NET module.	
94	EC=50025051	FL.NET RINTR Invalid Interrupt	A PUINT invalid status error occurred in the FL.NET module.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (13/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
95	EC=500250B1	FL.NET PUINTR Invalid Interrupt	A PUINT invalid status error occurred in the FL.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year)
96	EC=500250C1	FL.NET NINTR Invalid Interrupt	An NINT invalid status error occurred in the FL.NET module.	
97	EC=500250F1	FL.NET HERST Invalid Interrupt	A serious fault invalid interrupt occurred in the FL.NET module.	
98	EC=500250F2	FL.NET HERST2 Invalid Interrupt	A serious fault invalid interrupt 2 occurred in the FL.NET module.	
99	EC=500250F3	FL.NET BUERRSTAT Invalid Interrupt	A bus serious-error interrupt status invalid error occurred in the FL.NET module.	
100	EC=500250F6	FL.NET NHPMCLG Invalid Interrupt	A memory serious fault invalid interrupt status error occurred.	
101	EC=500250F7	FL.NET ECC 2bit Master Invalid Interrupt	A serious fault invalid status error occurred due to a memory ECC2 bit error in the FL.NET module.	
102	EC=500250F8	FL.NET RERRMST Invalid Interrupt	An RERR invalid interrupt status error occurred in the FL.NET module.	
103	EC=50025110	FL.NET Macro parameter error	An abnormal macro parameter error occurred in the FL.NET module.	
104	EC=50025130	FL.NET Undefined Macro	An undefined macro issuance error occurred in the FL.NET module.	
105	EC=50025700	FL.NET System Error	The system went down (a system error occurred) in the FL.NET module.	
106	EC=50025800	FL.NET Kernel Trap	The system went down (a kernel trap occurred) in the FL.NET module.	
107	EC=50025C70	FL.NET WDT timeout error	A watchdog timer timeout error occurred in the FL.NET module.	
108	EC=50027308	FL.NET I/O SEND_TIMEOUT	A transmission timeout error occurred in the FL.NET module.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (14/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
109	EC=5002730A	FL.NET I/O RESET_ERROR	An I/O hardware reset error occurred in the FL.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year)
110	EC=5002730E	FL.NET I/O MEMORY	An I/O memory error occurred in the FL.NET module.	
111	EC=50027370	FL.NET I/O EC_PCI_ERROR	An I/O PCI error occurred in the FL.NET module.	
112	EC=50027400	FL.NET I/O PCI_BUS_ERR	An I/O PCI bus error occurred in the FL.NET module.	
113	EC=50027505	FL.NET I/O INV_INTR	An error due to an invalid interrupt from I/O occurred in the FL.NET module.	
114	EC=50027510	FL.NET I/O IFCONFIG_UP	A network I/F initialization error occurred in the FL.NET module.	
115	EC=50027D01	FL.NET INVALID EXCEPTION	An invalid exception error occurred in the FL.NET module.	
116	EC=50027D13	FL.NET ETHERNET LSI CHECK ERROR	An Ethernet LSI check error occurred in the FL.NET module.	
117	EC=50027D14	FL.NET SDRAM CHECK ERROR	An SDRAM check error occurred in the FL.NET module.	
118	EC=50027D15	FL.NET OS-ROM CHECKSUM ERROR	A ROM checksum error occurred in the FL.NET module.	
119	EC=50027D18	FL.NET TASK-ROM CHECKSUM ERROR	A ROM (task) checksum error occurred in the FL.NET module.	
120	EC=5002D010	FL.NET Memory Alarm	An I/O memory alarm error occurred in the FL.NET module.	
121	EC=5002D330	FL.NET Hardware WDT timeout	A hardware watchdog timer timeout error occurred in the FL.NET module.	
122	EC=5002D340	FL.NET Software WDT Timeout	A software watchdog timer timeout error occurred in the FL.NET module	
123	EC=5002D810	FL.NET BPU Error	A BPU error occurred in the FL.NET module.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (15/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
124	EC=50030010	J.NET Bus error	A bus error occurred in the J.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year)
125	EC=50030011	J.NET Invalid address	An address error occurred in the J.NET module.	
126	EC=50030012	J.NET Invalid instruction	An invalid instruction error occurred in the J.NET module.	
127	EC=50030013	J.NET Division by zero	A division by zero error occurred in the J.NET module.	
128	EC=50030014	J.NET Privilege violation	A privilege violation error occurred in the J.NET module.	
129	EC=50030015	J.NET WDT timeout error	A watchdog timer timeout error occurred in the J.NET module.	
130	EC=50030016	J.NET Format error	A format error occurred in the J.NET module.	
131	EC=50030017	J.NET Spurious Interrupt	A spurious interrupt occurred in the J.NET module.	
132	EC=50030018	J.NET Unused exception	An unused exception occurred in the J.NET module.	
133	EC=50030019	J.NET Parity error	A parity error occurred in the J.NET module.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (16/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
134	EC=50030102	J.NET ROM1 checksum error	A ROM1 checksum error occurred in the J.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Sub module DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year)
135	EC=50030103	J.NET RAM1 compare error	A RAM1 comparison error occurred in the J.NET module.	
136	EC=50030105	J.NET RAM2 compare error	A RAM2 comparison error occurred in the J.NET module.	
137	EC=50030107	J.NET DMA1 send error	A DMA1 transfer (transmission) error occurred in the J.NET module.	
138	EC=50030108	J.NET DMA2 send error	A DMA2 transfer (transmission) error occurred in the J.NET module.	
139	EC=50030109	J.NET DMA1 receive error	A DMA1 transfer (reception) error occurred in the J.NET module.	
140	EC=5003010A	J.NET DMA2 receive error	A DMA2 transfer (reception) error occurred in the J.NET module.	
141	EC=5003010B	J.NET ROM3 checksum error	A ROM3 checksum error occurred in the J.NET module.	
142	EC=5003010C	J.NET ROM erasing error (program)	A ROM (program) erase error occurred in the J.NET module.	
143	EC=5003010D	J.NET ROM writing error (program)	A ROM (program) write error occurred in the J.NET module.	
144	EC=5003010E	J.NET ROM erasing error (parameter)	A ROM (parameter) erase error occurred in the J.NET module.	
145	EC=5003010F	J.NET ROM writing error (parameter)	A ROM (parameter) write error occurred in the J.NET module.	
146	EC=50030110	J.NET ROM writing error (writing over)	The maximum number of ROM writes in the J.NET module was exceeded.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (17/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
147	EC=50041401	D.NET MPU Register Compare Error	An MPU register comparison error occurred in the D.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Sub module DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year) DAT8: Channel no. (D.NET, ET.NET)
148	EC=50041402	D.NET MPU Operation Check Error	An MPU operation check error occurred in the D.NET module.	
149	EC=50041403	D.NET CAN Register Compare Error	A CAN register comparison error occurred in the D.NET module.	
150	EC=50041405	D.NET FROM Compare Check Error	A ROM comparison error occurred in the D.NET module.	
151	EC=50041406	D.NET FROM Checksum Error (microprogram)	A ROM sum error (program) occurred in the D.NET module.	
152	EC=50041407	D.NET SRAM Compare Check Error	A RAM comparison error occurred in the D.NET module.	
153	EC=50041409	D.NET MPU Built-in Timer Diagnosis Error	A diagnosis error occurred in the MPU built-in timer of the D.NET module.	
154	EC=5004140D	D.NET FROM Checksum Error (parameter)	A ROM sum error (parameter) occurred in the D.NET module.	
155	EC=50042403	D.NET Parity Error	A parity error occurred in the D.NET module.	
156	EC=50042404	D.NET Watch-Dog-Timer Timeout Error	A watchdog timer timeout error occurred in the D.NET module.	
157	EC=50043400	D.NET Undefined interrupt	An undefined interrupt occurred in the D.NET module.	
158	EC=50043404	D.NET General Invalid Instruction	A general invalid instruction error occurred in the D.NET module	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (18/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
159	EC=50043406	D.NET Slot Invalid Instruction	An invalid slot instruction error occurred in the D.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year) DAT8: Channel no. (D.NET, ET.NET)
160	EC=50043409	D.NET Address Error	An address error occurred in the D.NET module	
161	EC=50044181	D.NET Duplicated MAC ID(Other-Node Stop)	A duplicate MAC ID error (that stopped another node's communication) occurred in the D.NET module.	
162	EC=50044281	D.NET Duplicated MAC ID(Self-Node Stop)	A duplicate MAC ID error (that stopped the local node's communication) occurred in the D.NET module.	
163	EC=50044401	D.NET I/O Transmission Stop Error (ch0))	An I/O transmission stop error occurred at channel 0 in the D.NET module.	
164	EC=50044402	D.NET I/O Transmission Stop Error (ch1))	An I/O transmission stop error occurred at channel 1 in the D.NET module.	
165	EC=50049001	D.NET T/M Error	An error occurred on execution of a built-in T/M in the D.NET module.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (19/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
166	EC=500E 3031	ET.NET Inst. Alignment Error	An instruction alignment error occurred in the ET.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year) DAT8: Channel no. (D.NET, ET.NET)
167	EC=500E 3041	ET.NET Illegal Instruction	An illegal instruction error occurred in the ET.NET module.	
168	EC=500E 3081	ET.NET Privileged Instruction	A privileged instruction violation error occurred in the ET.NET module.	
169	EC=500E 30F9	ET.NET Illegal Exception	An illegal exception error occurred in the ET.NET module.	
170	EC=500E 3389	ET.NET FP Unavailable	An error due to an unavailable floating-point exception occurred in the ET.NET module.	
171	EC=500E 3391	ET.NET FP Program Error	A floating-point error occurred in the ET.NET module.	
172	EC=500E 3401	ET.NET Instruction Page Fault	An instruction access page fault occurred in the ET.NET module.	
173	EC=500E 3421	ET.NET Invalid Inst. Access	An instruction access error occurred in the ET.NET module.	
174	EC=500E 3461	ET.NET Inst. Access Protection	An instruction access protection error occurred in the ET.NET module.	
175	EC=500E 3471	ET.NET Data Alignment Error	A data alignment error occurred in the ET.NET module.	
176	EC=500E 3601	ET.NET Data Page Fault	A data access page fault error occurred in the ET.NET module.	
177	EC=500E 3621	ET.NET Invalid Data Access	A data access error occurred in the ET.NET module.	
178	EC=500E 3661	ET.NET Data Access Protection	A data access protection error occurred in the ET.NET module.	
179	EC=500E 3820	ET.NET Memory Error	A memory error occurred in the ET.NET module.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (20/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
180	EC=500E 3B70	ET.NET Bus Target Abort	A bus target abort error occurred in the ET.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year) DAT8: Channel no. (D.NET, ET.NET)
181	EC=500E 3B81	ET.NET System Bus Error CPU Master	A system bus error occurred during access from the ET.NET module.	
182	EC=500E 3B82	ET.NET System Bus Error CPU Target	A system bus error occurred during access to the ET.NET module.	
183	EC=500E 3B90	ET.NET PCI_BUS_ERR	An I/O PCI bus error occurred in the ET.NET module.	
184	EC=500E 5001	ET.NET Undefined Invalid Interrupt	An undefined invalid interrupt error occurred in the ET.NET module.	
185	EC=500E 5002	ET.NET INTEVT Invalid Interrupt	An INTEVT invalid interrupt occurred in the ET.NET module.	
186	EC=500E 50F1	ET.NET HERST Invalid Interrupt	A serious fault invalid interrupt occurred in the ET.NET module.	
187	EC=500E 50F2	ET.NET HERST2 Invalid Interrupt	A serious fault invalid interrupt 2 occurred in the ET.NET module.	
188	EC=500E 50F3	ET.NET BUERRSTAT Invalid Interrupt	A bus serious-error interrupt status invalid error occurred in the ET.NET module.	
189	EC=500E 50F6	ET.NET NHPMCLG Invalid Interrupt	A memory serious fault invalid interrupt status error occurred in the ET.NET module.	
190	EC=500E 50F7	ET.NET ECC 2bit Master Invalid Interrupt	A memory ECC2 bit error invalid serious fault status error occurred in the ET.NET module.	
191	EC=500E 50F8	ET.NET RERRMST Invalid Interrupt	An RERR invalid interrupt status error occurred in the ET.NET module.	
192	EC=500E 5110	ET.NET Macro parameter error	An abnormal macro parameter error occurred in the ET.NET module.	
193	EC=500E 5130	ET.NET Macro parameter error	An undefined macro issuance error occurred in the ET.NET module.	

Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (21/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
194	EC=500E 5700	ET.NET System Error	The system went down (a system error occurred) in the ET.NET module.	SLOT: Fixed to 0xFFFFFFFF MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Error code output from the module DAT1: Module no. 0x0000: Module 0/Channel 0 0x0001: Module 1/Channel 1 0x0002: Module 2/Channel 2 0x0003: Module 3/Channel 3 0x0011: Main module 0x0012: Submodule DAT2: Time when the error occurred (seconds) DAT3: Time when the error occurred (minutes) DAT4: Time when the error occurred (hours) DAT5: Time when the error occurred (date) DAT6: Time when the error occurred (month) DAT7: Time when the error occurred (year) DAT8: Channel no. (D.NET, ET.NET)
195	EC=500E 5800	ET.NET Kernel Trap	The system went down (a kernel trap occurred) in the ET.NET module.	
196	EC=500E 5C70	ET.NET WDT timeout error	A watchdog timer timeout error occurred in the ET.NET module.	
197	EC=500E 7308	ET.NET SEND_TIMEOUT	A transmission timeout error occurred in the ET.NET module.	
198	EC=500E 730A	ET.NET RESET_ERROR	A hardware reset error occurred in the ET.NET module.	
199	EC=500E 7505	ET.NET INV_INTR	An invalid interrupt error from a line occurred in the ET.NET module.	
200	EC=500E 7D01	ET.NET INVALID EXCEPTION	An invalid exception error occurred in the ET.NET module.	
201	EC=500E 7D11	ET.NET Invalid MAC ADDRESS	An abnormal MAC address error occurred in the ET.NET module.	
202	EC=500E 7D13	ET.NET ETHERNET LSI CHECK ERROR	An Ethernet LSI check error occurred in the ET.NET module.	
203	EC=500E 7D14	ET.NET SDRAM CHECK ERROR	An SDRAM check error occurred in the ET.NET module.	
204	EC=500E 7D18	ET.NET ROM CHECKSUM ERROR	A ROM checksum error occurred in the ET.NET module.	
205	EC=500E D010	ET.NET Memory Alarm	A memory alarm error occurred in the ET.NET module.	
206	EC=500E D810	ET.NET BPU Error	A BPU error occurred in the ET.NET module.	

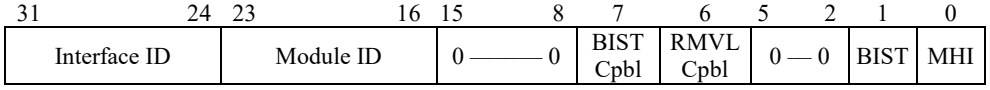
Table E-16 Error Codes, Subtitles, and their Descriptions (Module Error) (22/22)

No.	Error code	Subtitle	Description	Contents of MSW0, MSW1, and DATn
207	EC=51000000	Optional Module startup check error	An error occurred in the startup monitoring check for an optional module.	SLOT: Fixed to 0 MSW0: Fixed to 0 MSW1: Fixed to 0
208	EC=51000001	System Register Clear Time Out	An error occurred in the startup monitoring check for the HP.	
209	EC=51000002	Optional Parameter size Error	An optional module parameter size error occurred.	SLOT: Fixed to 0 MSW0: Fixed to 0 MSW1: Fixed to 0 DAT0: Area number of setting parameter

Table E-17 Detailed MSW Data for Each Module (1/3)

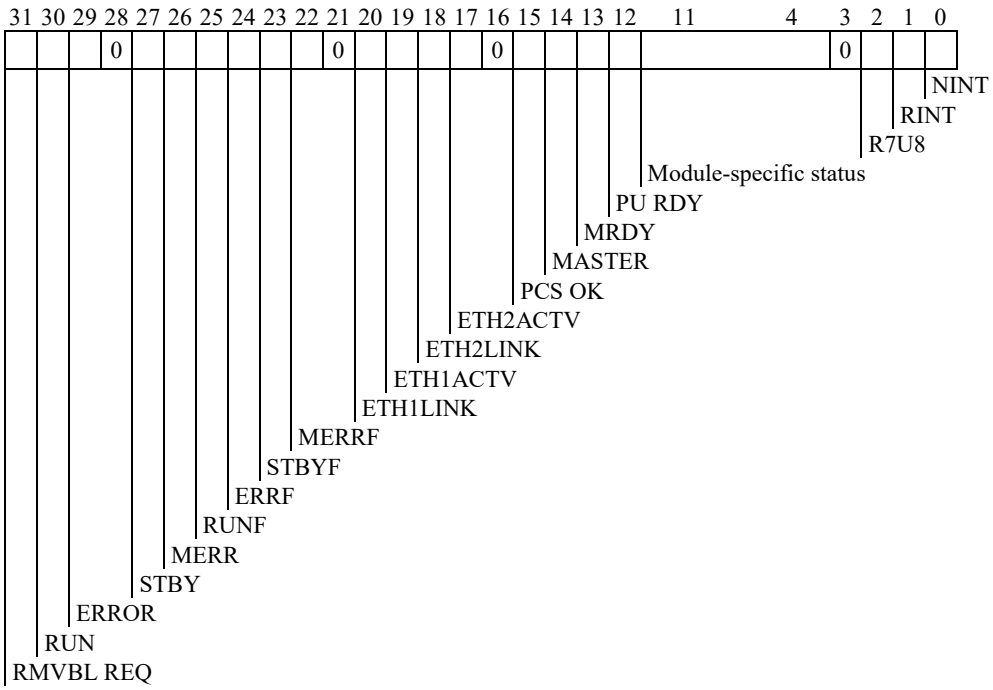
Module name	MSW	Detailed data

CPU — MSW0



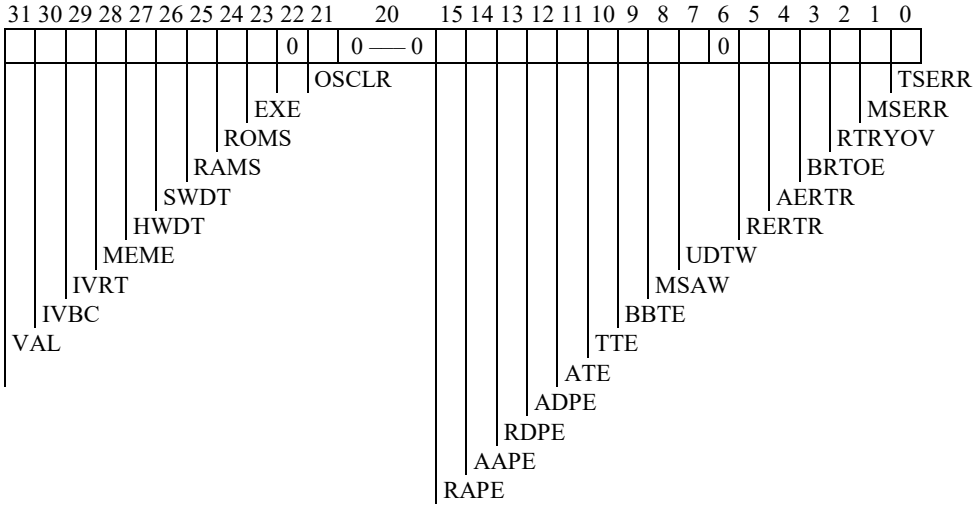
Bit no.	Bit name	Read-in value	Description
31 to 24	Interface ID	Interface ID	Module ID code for software
23 to 16	Module ID	Module ID	Module ID code for hardware
15 to 8	–	Fixed to 0	–
7	BIST Cpbl	0: Not equipped 1: Equipped	Whether the module is equipped with a self-diagnostic function (optional)
6	RMVL Cpbl	0: Cannot be hot swapped 1: Can be hot swapped	Whether the module can be hot swapped (for future use for PU)
5 to 2	–	Fixed to 0	–
1	BIST	0: Self-diagnosis ended 1: Self-diagnosis running	Operation status of the self-diagnostic function of the module (optional)
0	MHI	0: Initialization process ended 1: Initialization process running	Initialization status of the module hardware

MSW1



Bit no.	Bit name	Read-in value	Description
31	RMVBL REQ	0: Not requested 1: Requested	Whether hot swapping is requested
30	RUN	1: RUN setting process running	Module status (RUN LED turns on or flashes)
29	ERROR	0: No error 1: Error	Error status (ERR LED turns on or flashes)
28	–	Fixed to 0	–
27	STBY	1: STANDBY setting process running	Module status (STBY LED turns on or flashes)
26	MERR	1: Module error	Module error status (MERR LED turns on or flashes)

Table E-17 Detailed MSW Data for Each Module (2/3)

<table>
<tr><th>Module name</th><th>MSW</th><th>Detailed data</th></tr>
<tr><td>CPU
(continued from the preceding page)</td><td>MSW1</td><td>
<table>
<tr><th>Bit no.</th><th>Bit name</th><th>Read-in value</th><th>Description</th></tr>
<tr><td>25</td><td>RUNF</td><td>1: RUN flashing</td><td>Module state (RUN LED flashes)</td></tr>
<tr><td>24</td><td>ERRF</td><td>1: ERR flashing</td><td>Module state (ERR LED flashes)</td></tr>
<tr><td>23</td><td>STBYF</td><td>1: STBY flashing</td><td>Module state (STBY LED flashes)</td></tr>
<tr><td>22</td><td>MERRF</td><td>1: MERR flashing</td><td>Module state (MERR LED flashes)</td></tr>
<tr><td>21</td><td>–</td><td>Fixed to 0</td><td>–</td></tr>
<tr><td>20</td><td>ETH1LINK</td><td>1: Transmitting successfully</td><td>LINK establishment state for port Ether1</td></tr>
<tr><td>19</td><td>ETH1ACTV</td><td>1: Receiving successfully</td><td>TX and RX state for port Ether1</td></tr>
<tr><td>18</td><td>ETH2LINK</td><td>1: Transmitting successfully</td><td>LINK establishment state for port Ether2</td></tr>
<tr><td>17</td><td>ETH2ACTV</td><td>1: Receiving successfully</td><td>TX and RX state for port Ether2</td></tr>
<tr><td>16</td><td>–</td><td>Fixed to 0</td><td>–</td></tr>
<tr><td>15</td><td>PCS OK</td><td>1: PCS OK</td><td>CPU: PCS OK status (output)</td></tr>
<tr><td>14</td><td>MASTER</td><td>1: MASTER</td><td>CPU: MASTER status (output)</td></tr>
<tr><td>13</td><td>MRDY</td><td>1: MRDY</td><td>Master ready status (valid only for CPU)</td></tr>
<tr><td>12</td><td>PU RDY</td><td>1: PU ready</td><td>PU ready status (valid only for PU)</td></tr>
<tr><td>11 to 4</td><td>Module-specific status</td><td>Module-specific status</td><td>Module-specific status</td></tr>
<tr><td>3</td><td>–</td><td>Fixed to 0</td><td>–</td></tr>
<tr><td>2</td><td>R7U8</td><td>0: No error
1: Error</td><td>Whether the following error occurred: when the DMA controller of R700 is performing a burst read, TRDY is sent from the slave less than eight times.</td></tr>
<tr><td>1</td><td>RINT</td><td>0: No factor
1: Factor detected</td><td>Whether a general interrupt other than network related interrupts occurred</td></tr>
<tr><td>0</td><td>NINT</td><td>0: No factor
1: Factor detected</td><td>Whether a network interrupt occurred</td></tr>
</table>
</td></tr>
<tr><td></td><td>MSW2</td><td>


<table>
<tr><th>Bit no.</th><th>Bit name</th><th>Read-in value</th><th>Description</th></tr>
<tr><td>31</td><td>VAL</td><td>1: Valid
0: Invalid</td><td>Whether this register is valid</td></tr>
<tr><td>30</td><td>IVBC</td><td>1: Battery voltage dropped
0: Normal</td><td>Voltage dropped, capacity decreased, or unmounted</td></tr>
<tr><td>29</td><td>IVRT</td><td>1: Backup failed
0: Backup succeeded</td><td>Whether data retention succeeded or failed during a power failure</td></tr>
</table>
</td></tr>
</table>

Table E-17 Detailed MSW Data for Each Module (3/3)

Module name	MSW	Detailed data
CPU (continued from the preceding page)	MSW2	

Bit no.	Bit name	Read-in value	Description
28	MEME	1: Error 0: Normal	An unrecoverable error occurred when the CPU was accessing built-in memory.
27	HWDT	1: Error 0: Normal	A hardware watchdog timer timeout occurred (CPU is included in EXE).
26	SWDT	1: Error 0: Normal	A software watchdog timer timeout occurred.
25	RAMS	1: Error 0: Normal	A RAM checksum error occurred.
24	ROMS	1: Error 0: Normal	A ROM checksum error occurred.
23	EXE	1: Error 0: Normal	Hardware error (CPU stopped HWDT and SWDT)
22	–	Fixed to 0	–
21	OSCLR	1: OS clear 0: Normal	OS clear (Only modules such as the CPU and XPU that use the OS are valid.)
20 to 16	–	Fixed to 0	–
15	RAPE	1: Error 0: Normal	Received address parity error (This error occurs when the CPU is the master.)
14	AAPE	1: Error 0: Normal	Asserted address parity error (This error occurs when the CPU is a target.)
13	RDPE	1: Error 0: Normal	Received data parity error (This error occurs when the master is performing a write operation, or when the target is performing a read operation.)
12	ADPE	1: Error 0: Normal	Asserted data parity error (This error occurs when the master is performing a read, or when the target is performing a write.)
11	ATE	1: Error 0: Normal	Address cycle timeout error (Only the master is valid.)
10	TTE	1: Error 0: Normal	Transaction timeout error (Only the master is valid.)
9	BBTE	1: Error 0: Normal	BGACK busy timeout error (Only the CPU is valid.)
8	MSAW	1: Error 0: Normal	Invalid misalign access
7	UDTW	1: Error 0: Normal	Unsupported transaction error (The PU is not supported.)
6	–	Fixed to 0	–
5	RERTR	1: Error 0: Normal	Received error transaction (Only the master is valid.)
4	AERTR	1: Error 0: Normal	Asserted error transaction
3	BRTOE	1: Error 0: Normal	Bus request timeout (Only the master is valid.)
2	RTRYOV	1: Exceeded 0: Not exceeded	The retry frequency limit was exceeded when the CPU was the bus master.
1	MSERR	1: Received 0: Not received	The CPU received SERROR when it was the master.
0	TSERR	1: Received 0: Not received	The CPU received SERROR when it was operated as a target.

Table E-18 Detailed Data of the RI/O-IF Module Error (Module Error)

DATn	Description	Detailed data
DAT0 DAT1	NASC1 register value	DAT0: NS_INTST DAT1: NS_I2RSVI

Table E-19 Detailed Data of the LSI Internal Timeout Error (Module Error)

DATn	Description	Detailed data
DAT0 | DAT35	NASC1 register value	DAT0: NS_HERST register value DAT1: NS_HERENB register value DAT2: NP_ERRLOG register value DAT3: NP_ERRENB register value DAT4: NP_ACCSTOVAL register value DAT5: NM_HEREB register value DAT6: NM_HERST register value DAT7: ND_DCMST register value DAT8: ND_DCMINT_EN register value DAT9: ND_NDUTOC register value DAT10: NE_ERRST register value DAT11: NE_ERRENB register value DAT12: NE_ACCSTOVAL register value DAT13: NF_ERRST register value DAT14: NF_ERRINTENB register value DAT15: NB_PCIHERINT register value DAT16: NB_PCIHEREB register value DAT17: NB_PCIAHERINT register value DAT18: NB_PCIAHEREB register value DAT19: NB_PCIMZTOTH register value DAT20: SP_ERRS register value DAT21: SP_ERRM register value DAT22: SP_MATO register value DAT23: NZ_ERRST register value DAT24: NZ_ERRENB register value DAT25: NZ_ACCSTOVAL register value DAT26: NO_ERRST register value DAT27: NO_ERRENB register value DAT28: NO_ACCSTOVAL register value DAT29: NL_NLUSTS register value DAT30: NL_NLUENB register value DAT31: NL_NLUTOC register value DAT32: NR_RERRLOGMST register value DAT33: NR_RERRLOGSLV register value DAT34: NR_RERREN register value DAT35: NR_NRUTMR register value

Table E-20 Detailed Data of the SPU Error (Module Error) (1/2)

DATn	Description	Detailed data
DAT0	Context number	Context number of the running ladder program
DAT1	Task number	Number of the task that activated the ladder program
DAT2	State	State of the ladder program
DAT3	Identification flag	Flag that identifies whether a break handling program is running
DAT4	Error code	Error code
DAT5 | DAT35	NASC1 register value	DAT5: SP_C1ERRS register value DAT6: SP_C1ERRM register value DAT7: SP_C1INTS register value DAT8: SP_C1INTM register value DAT9: SP_INTSTS register value DAT10: SP_ERRSTS register value DAT11: SP_STS register value DAT12: SP_RUN register value DAT13: SP_CNTC register value DAT14: SP_ERTRI register value DAT15: SP_DPEA register value DAT16: SP_ATRP register value DAT17: SP_ITRP register value DAT18: SP_DSOVA register value DAT19: SP_MMR register value DAT20: SP_MATO register value DAT21: SP_MERS register value DAT22: SP_MERAX register value DAT23: SP_MERDX register value DAT24: SP_MERSX register value DAT25: SP_MEC1X register value DAT26: SP_MEC1C register value DAT27: SP_MERAC register value DAT28: SP_MODE register value DAT29: SP_CAREA0 register value DAT30: SP_CAREA1 register value DAT31: SP_CAREA2 register value DAT32: SP_CAREA3 register value DAT33: SP_STAREA register value DAT34: SP_PC register value DAT35: SP_EXDIA register value

Table E-20 Detailed Data of the SPU Error (Module Error) (2/2)

DATn	Description	Detailed data
DAT36 | DAT56	NASC1 register value	DAT36: SP_DBT0 register value DAT37: SP_DBT1 register value DAT38: SP_DPLS register value DAT39: SP_DCSTR register value DAT40: SP_DIFAA register value DAT41: SP_DDA register value DAT42: SP_DFI register value DAT43: SP_DRA register value DAT44: SP_DOFA register value DAT45: SP_DRD register value DAT46: SP_DEXA register value DAT47: SP_DALUC register value DAT48: SP_DALUO register value DAT49: SP_DALUS register value DAT50: SP_DWD register value DAT51: SP_DFPUCS register value DAT52: SP_DMCS register value DAT53: SP_DEACS register value DAT54: SP_SEQ_SN2 register value DAT55: SP_SEQ_SN1 register value DAT56: SP_SEQ_SN register value

Table E-21 Detailed Data of the RI/O-IF RI/O Error (Module Error)

DATn	Description	Detailed data
DAT0	Error code	Error code (134)
DAT0 | DAT21	NASC1 register value	DAT1: NO_RIOMODE DAT2: NO_RIOSTART DAT3: NO_RIOSTATUS DAT4: NO_C0ERRST DAT5: NO_C1ERRST DAT6: NO_SPERRST DAT7: NO_C0ERRENB DAT8: NO_C1ERRENB DAT9: NO_SPERRENB DAT10: NO_C0INTST DAT11: NO_C1INTST DAT12: NO_SPINTST DAT13: NO_C0INTENB DAT14: NO_C1INTENB DAT15: NO_SPINTENB DAT16: NO_NGUERRINJ DAT17: NO_PTYERINJ DAT18: NO_ACCSTOVAL DAT19: NO_PRTYERRINFO DAT20: NO_PRTYERRADR DAT21: NO_PRTYERRDAT

Table E-22 Detailed Data of the Memory Patrol Error (Module Error)

DATn	Description	Detailed data
DAT0 | DAT3	NASC1 register value	DAT0: NM_DMASRCADR1 DAT1: NM_DMADSTADR1 DAT2: NM_DMACNT1 DAT3: NM_DMAMODE_SET1
DAT4	Status	DMA status
DAT5 | DAT7	NASC1 register value	DAT5: NM_DMACHKSUM1 DAT6: NM_DMACHKERRADR1 DAT7: NM_DMACHKERRDAT1

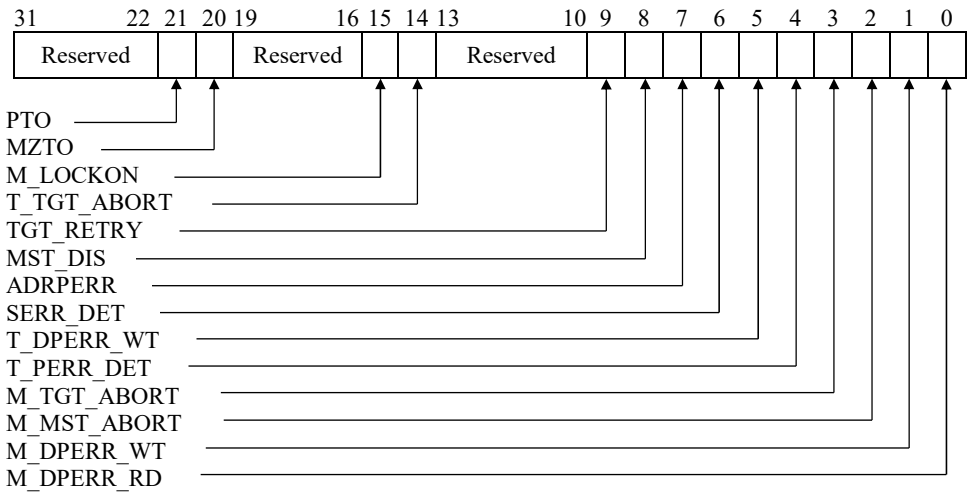
Table E-23 Detailed Data of the Memory Alarm (Module Error)

DATn	Description	Detailed data
DAT0	Timer base value	Timer base value (at the first error detection)
DAT1 | DAT5	NASC1register value	NASC1 register value at the first error detection DAT1: NM_ECC3STATUS DAT2: NM_ECC3CORADR DAT3: NM_ECC3CORDAT DAT4: NM_ECC3CORDAT_UL DAT5: NM_ECC3CORSUM
DAT6 | DAT11	Details of the second error	Value at the second error detection The contents are the same as those of DAT0 to 5.
DAT12 | DAT17	Details of the third error	Value at the third error detection The contents are the same as those of DAT0 to 5
DAT18 | DAT23	Details of the fourth error	Value at the fourth error detection The contents are the same as those of DAT0 to 5.
DAT24 | DAT29	Details of the fifth error	Value at the fifth error detection The contents are the same as those of DAT0 to 5.

Table E-24 Detailed Data of the Primary Battery Error (Module Error)

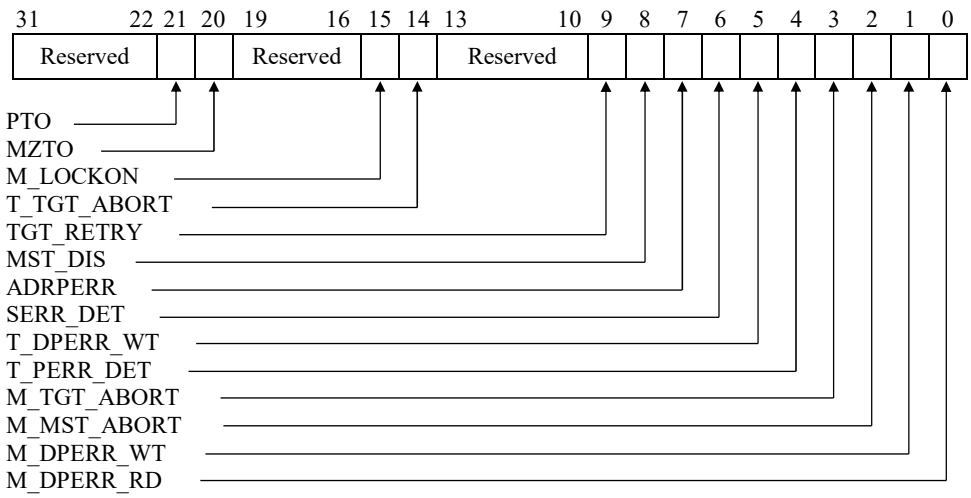
DATn	Description	Detailed data
DAT0	Primary battery state	Primary battery state  Time elapsed since power from the primary battery stopped Low voltage detected in the primary battery
DAT1	Cumulative time that power was not provided	Cumulative time that the primary battery did not provide power (in seconds)

Table E-25 Detailed Data of the PCI Bus Error (1/7)

DATn	Description	Detailed data
DAT0	NB_PCIHERINTC0	

Bit	Name	Set value	Description
31 to 22	Reserved	–	–
21	PTO	1: Error 0: Normal	This detects the following error: NBU could not obtain the bus right within the specified period after NBU asserted a bus request to the PCI host.
20	MZTO	1: Error 0: Normal	This indicates that the following error occurred: ACK was not asserted within the specified period after NBU asserted REQ to an internal block (NMU, NZU).
19 to 16	Reserved	–	–
15	M_LOCKON	1: Error 0: Normal	This indicates that the following error occurred: when NBU was initiator, it unlocked and transferred a specific target that had been locked. Exclusive control by LOCK
14	T_TGT_ABORT	1: Error 0: Normal	This indicates that the following error occurred: when NBU was a target, it terminated a transaction via a target abort. A target abort is issued if the combination of the last 2 bits of the address and byte enable signal is invalid.
13 to 10	Reserved	–	–
9	TGT_RETRY	1: Error 0: Normal	This indicates that the following error occurred: when NBU was a target, the initiator did not perform retry processing within the 2^{15} PCI clock limit after the target gave a retry response. This error is detected only for memory-read transfers.
8	MST_DIS	1: Error 0: Normal	This indicates that the following error occurred: although bit 2 of PCICONF1 was set to 0 and NBU was not the initiator, NBU acted as initiator (performed PI/O transfer).
7	ADRPERR	1: Error 0: Normal	This indicates that an address parity error occurred. This error is detected only when both bits 6 and 8 of PCICONF1 are set to 0.

Table E-25 Detailed Data of the PCI Bus Error (2/7)

<table>
<tr><th>DATn</th><th>Description</th><th>Detailed data</th></tr>
<tr><td>DAT0</td><td>NB_PCIHERINTC0 (continued from the preceding page)</td><td>
<table>
<tr><th>Bit</th><th>Name</th><th>Set value</th><th>Description</th></tr>
<tr><td>6</td><td>SERR_DET</td><td>1: Error
0: Normal</td><td>This indicates that SERR_N was asserted when NBU was a host.</td></tr>
<tr><td>5</td><td>T_DPERR_WT</td><td>1: Error
0: Normal</td><td>This indicates that the following error was detected: when NBU was a target, NBU detected a parity error during a target-write. This error is detected only when bit 6 of PCICONF1is set to 6.</td></tr>
<tr><td>4</td><td>T_PERR_DET</td><td>1: Error
0: Normal</td><td>This indicates that the following error occurred: when NBU was a target, it accepted a PERR_N assert during a target-read. This error is detected only when bit 6 of PCICONF1 is set to 1.</td></tr>
<tr><td>3</td><td>M_TGT_ABORT</td><td>1: Error
0: Normal</td><td>This indicates that the following error occurred: when NBU was initiator, the transaction was terminated via a target abort.</td></tr>
<tr><td>2</td><td>M_MST_ABORT</td><td>1: Error
0: Normal</td><td>This indicates that the following error occurred: when NBU was initiator, the transaction was terminated via a master abort.</td></tr>
<tr><td>1</td><td>M_DPERR_WT</td><td>1: Error
0: Normal</td><td>This indicates that the following error occurred: when NBU was initiator, it accepted a PERR_N assert while writing data to a target. This error is detected only when bit 6 of PCICONF1 is set to 0.</td></tr>
<tr><td>0</td><td>M_DPERR_RD</td><td>1: Error
0: Normal</td><td>This indicates that the following error occurred: when NBU was initiator, it detected a parity error while reading data from a target. This error is detected when bit 6 of PCICONF1 is set to 0.</td></tr>
</table>
</td></tr>
<tr><td>DAT1</td><td>NB_PCIHEREBC0</td><td>


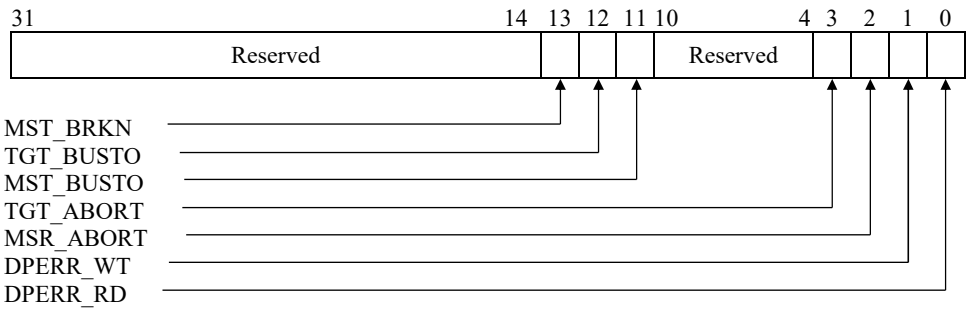
</td></tr>
</table>

Table E-25 Detailed Data of the PCI Bus Error (3/7)

DATn	Description	Detailed data
DAT1	NB_PCIHEREBC0 (continued from the preceding page)	

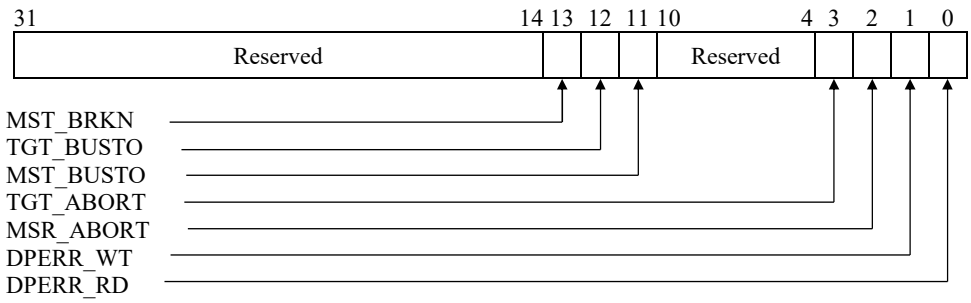
Bit	Name	Set value	Description
31 to 22	Reserved	–	–
21	PTO	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via PTO of NB_PCIHERINTC0/C1.
20	MZTO	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via MZTI of NB_PCIHERINTC0/C1.
19 to 16	Reserved	–	–
15	M_LOCKON	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via M_LOCKON of NB_PCIHERINTC0/C1.
14	T_TGT_ABORT	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via T_TGT_ABORT of NB_PCIHERINTC0/C1.
13 to 10	Reserved	–	–
9	TGT_RETRY	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via TGT_RETRY of NB_PCIHERINTC0/C1.
8	MST_DIS	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via MST_DIS of NB_PCIHERINTC0/C1.
7	ADRPERR	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via ADRPERR of NB_PCIHERINTC0/C1.
6	SERR_DET	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via SERR_DET of NB_PCIHERINTC0/C1.
5	T_DPERR_WT	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via T_DPERR_WT of NB_PCIHERINTC0/C1.
4	T_PERR_DET	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via T_PERR_DET of NB_PCIHERINTC0/C1.
3	M_TGT_ABORT	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via M_TGT_ABORT of NB_PCIHERINTC0/C1.
2	M_MST_ABORT	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via M_MST_ABORT of NB_PCIHERINTC0/C1
1	M_DPERR_WT	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via M_DPERR_WT of NB_PCIHERINTC0/C1.
0	M_DPERR_RD	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via M_DPERR_RD of NB_PCIHERINTC0/C1.

Table E-25 Detailed Data of the PCI Bus Error (4/7)

DATn	Description	Detailed data
DAT2	NB_PCIAHERINTC0	

Bit	Name	Set value	Description
31 to 14	Reserved	–	–
13	MST_BRKN	1: Error 0: Normal	This indicates that an initiator that had the bus right did not perform a FRAME_N assertion within the specified time limit.
12	TGT_BUSTO	1: Error 0: Normal	This indicates that a target did not perform a TRDY_N or STOP_N assertion within the specified time limit at the first data transfer.
11	MST_BUSTO	1: Error 0: Normal	This indicates that an initiator did not receive an IRDY_N assertion within the time limit specified in ITOTH at data transfer.
10 to 4	Reserved	–	–
3	TGT_ABORT	1: Error 0: Normal	This indicates that a target abort occurred when a device other than the local NBU was initiator.
2	MSR_ABORT	1: Error 0: Normal	This indicates that a master abort occurred when a device other than the local NBU was initiator.
1	DPERR_WT	1: Error 0: Normal	This indicates that PERR_N was asserted at a data write when a device other than the local NBU was initiator.
0	DPERR_RD	1: Error 0: Normal	This indicates that PERR_N was asserted at a data read when a device other than the local NBU was initiator.

Table E-25 Detailed Data of the PCI Bus Error (5/7)

DATn	Description	Detailed data
DAT3	NB_PCIAHERBC0	

Bit	Name	Set value	Description
31 to 14	Reserved	–	–
13	MST_BRKN	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via MST_BRKN of NB_PCIAHERINTC0/C1.
12	TGT_BUSTO	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via TGT_BUSTO of NB_PCIAHERINTC0/C1.
11	MST_BUSTO	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via MST_BUSTO of NB_PCIAHERINTC0/C1.
10 to 4	Reserved	–	–
3	TGT_ABORT	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via TGT_ABORT of NB_PCIAHERINTC0/C1.
2	MSR_ABORT	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via MSR_ABORT of NB_PCIAHERINTC0/C1.
1	DPERR_WT	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via DPERR_WT of NB_PCIAHERINTC0/C1.
0	DPERR_RD	1: Interrupt permitted 0: Interrupt prohibited	Permit a serious fault interrupt via DPERR_RD of NB_PCIAHERINTC0/C1.

Table E-25 Detailed Data of the PCI Bus Error (6/7)

DATn	Description	Detailed data
DAT4	NB_PCIALR	31 … 0 ALOG Bit: 31 to 0 / Name: ALOG / Set value: Address of PCI bus when the error occurred / Description: These bits contain address information from when an error occurred in the PCI bus. The address of the PCI bus is stored here.
DAT5	NB_PCIDLR	31 … 0 DLOG Bit: 31 to 0 / Name: DLOG / Set value: Data on PCI bus when the error occurred / Description: These bits contain data information from when an error occurred in the PCI bus.
DAT6	NB_PCICLR	31 30 … 27 26 25 24 … 4 3 … 0 Reserved | Reserved | CBE MSTMP, TGT, MSTSP Bit: 31 / Name: MSTMP / Set value: 1: Error, 0: Normal / Description: This indicates that an error occurred during a PCI access when MP was initiator. Bit: 30 to 27 / Name: Reserved / Set value: 0 / Description: Reserved Bit: 26 / Name: TGT / Set value: 1: Error, 0: Normal / Description: This indicates that an error occurred during data transfer from (read) or to (write) a target. Bit: 25 / Name: MSTSP / Set value: 1: Error, 0: Normal / Description: This indicates that an error occurred during a PCI access when SP was initiator. Bit: 24 to 4 / Name: Reserved / Set value: 0 / Description: Reserved Bit: 3 to 0 / Name: CBE / Set value: 1: Error, 0: Normal / Description: This bit contains the values of PCI transfer command information (C/BE_N) from when an error occurred.
DAT7	NB_PCIPLR	31 … 1 0 … 0 | PRTY Bit no.: 31 to 1 / Bit name: – / Bit value: Fixed to 0 / Description: – Bit no.: 0 / Bit name: PRTY / Bit value: 0: PRTY OFF, 1: PRTY ON / Description: This bit contains the PRTY signal value.

Table E-25 Detailed Data of the PCI Bus Error (7/7)

DATn	Description	Detailed data
DAT8	NB_PCIBMLR	

31	4	3	2	1	0
0	0	REQ3ID	REQ2ID	REQ1ID	REQ0ID

Bit no.	Bit name	Bit value	Description
31 to 4	–	Fixed to 0	–
3	REQ3ID	1: Error 0: Normal	This indicates that an error occurred when device 3 (REQ2) was initiator.
2	REQ2ID	1: Error 0: Normal	This indicates that an error occurred when device 2 (REQ1) was initiator.
1	REQ1ID	1: Error 0: Normal	This indicates that an error occurred when device 1 (REQ0) was initiator.
0	REQ0ID	1: Error 0: Normal	This indicates that an error occurred when device 0 (local NBU) was initiator.

E.3.6 Kernel warning

This message does not indicate the occurrence of an error, but the occurrence of an abnormality affecting user programs.
Table E-26 shows and describes the error message.

Table E-26 Format of the Kernel Warning Message

```
CPU xxxxxxxx
%CPMS-W-xxxx-0007  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx Kernel Warning
TN  =xxxxxxxx
DAT0 =xxxxxxxx  DAT1 =xxxxxxxx  DAT2 =xxxxxxxx  DAT3 =xxxxxxxx  DAT4 =xxxxxxxx
DAT5 =xxxxxxxx  DAT6 =xxxxxxxx  DAT7 =xxxxxxxx
MESSAGE=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Item	Description
EC	Error code: This indicates the warning type. See Table E-27.
TN	Task number: This indicates the task number. See Table E-27.
DAT*n*	This is the error analysis data. Because the contents of this data vary depending on the error type, see Table E-27.
MESSAGE	This is the message to be displayed. Because the contents of this message vary depending on the error type, see Table E-27.

Table E-27 Error Codes and their Descriptions (Kernel Warning)

No.	Error code	Message	Description	Contents of TN and DATn
1	EC=05A00001	No message is displayed.	At system time synchronization, there was a difference of 15 seconds or more compared to the TOD (time of day).	TN: The task number of the task that was running at system time synchronization DAT0: Difference value in seconds DAT1: Difference value in nanoseconds DAT2: Time value in seconds of the time when synchronization was attempted DAT3: Difference value in nanoseconds, of the time when synchronization was attempted DAT4: TOD value in seconds DAT5: TOD value in nanoseconds
2	EC=05A00006	No message is displayed.	The task to which pupost was issued was not in the WAIT state.	TN: Current task number DAT0: CM data [0] DAT1: CM data [1] DAT2: CM data [2] DAT3: Return value DAT4: 0 DAT5: 0 DAT6: 0 DAT7: 0

E.3.7 Kernel information

This error indicates that a temporary abnormality, which has no negative effects on the execution of user programs, occurred in the CPMS.
Table E-28 shows and describes the error message.

Table E-28 Format of the Kernel Information Message

```
CPU xxxxxxxx
%CPMS-I-xxxx-0008   SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx Kernel information
TN   =xxxxxxxx
DAT0 =xxxxxxxx  DAT1 =xxxxxxxx  DAT2 =xxxxxxxx  DAT3 =xxxxxxxx  DAT4 =xxxxxxxx
DAT5 =xxxxxxxx  DAT6 =xxxxxxxx  DAT7 =xxxxxxxx
MESSAGE=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Item	Description
EC	Error code: This indicates the information type. The system does not currently output this error.

E.3.8 System down (system error)

This error message indicates that the CPMS detected an error that prevented the CPMS from continuing processing.
Table E-29 shows and describes the error message.

Table E-29 Format of the System Down (system error) Message

```
CPU xxxxxxxx
%CPMS-F-CPMS-0009  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx System down (subtitle)
TN  =xxxxxxxx  PC  =xxxxxxxx  EXPEV=xxxxxxxx  FADR =xxxxxxxx  SR  =xxxxxxxx
  EXECD=xxxxxxxx
PR  =xxxxxxxx  SP  =xxxxxxxx  GBR  =xxxxxxxx  MACH =xxxxxxxx  MACL =xxxxxxxx
R0  =xxxxxxxx  R1  =xxxxxxxx  R2  =xxxxxxxx  R3  =xxxxxxxx  R4  =xxxxxxxx
R5  =xxxxxxxx  R6  =xxxxxxxx  R7  =xxxxxxxx  R8  =xxxxxxxx  R9  =xxxxxxxx
R10 =xxxxxxxx  R11 =xxxxxxxx  R12 =xxxxxxxx  R13 =xxxxxxxx  R14 =xxxxxxxx
R15 =xxxxxxxx
INST =xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
(PC =)xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
STACK=xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
(SP =)xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
PC  =xxxxxxxx
FADR=xxxxxxxx
```

Item	Description
EC	Error code (See Table E-30.)
TN	Task number of the task at which an error occurred
PC	Contents of the program counter
EXPEV	Contents of the exception code register. The exception code register is a 32-bit register defining the causes of data access exceptions and alignment exceptions.
FADR	Fault address
SR	Status register. This register defines the state of the processor.
EXECD	Instruction that was executed when an error occurred (error factor code)
PR	Content of the procedure register. The procedure register is used for subroutine calls. If the executed program was the last program in the subroutine calling sequence, this register contains the return address.
SP	Contents of the stack pointer (R15 is used as a stack pointer).
GBR	Contents of the global base register. This register contains the base addresses for GBR-indirect addressing with displacement and indexed GBR-indirect addressing.
MACH	MAC register. This register is used for storing additional values of MAC (multiply and accumulate operation) instructions, MAC instructions, and results of MUL instructions. If the calculated result is a value of 64 bits or more, this register stores the upper 32 bits.
MACL	MAC register. If the calculated result is a 64-bit value, this register stores the lower 32 bits. If the calculated result is a 32-bit value, the register stores the 32 bits.
R*xx*	Contents of the general register represented by the value at *xx*.
INST	Eight instructions preceding and following the PC address
PC	Instruction of the PC
STACK	Eight long-word data items preceding and following the SP address
SP	Contents of the SP address (contents of the stack)
PC	Information about the address contained in the program counter is displayed in parentheses. When the address is for a program, the following information is displayed: `name` = *program name*, `type` = *program type (program location)*, `raddr` = *relative address from the program*
FADR	Information about the fault address is displayed in parentheses.

Table E-30 Error Codes, Subtitles, and their Descriptions (System Error) (1/2)

No.	Error code	Subtitle	Description	Explanation
1	EC=03030000	Inst. Alignment Error	Instruction alignment error	Operands specified by the instruction are not word aligned.
2	EC=03040000	Illegal Instruction	Illegal instruction error	An attempt was made to execute an illegal instruction.
3	EC=030F0000	Illegal Exception	Illegal exception	An undefined exception was reported.
4	EC=03380000	FP Unavailable	Floating-point unavailable	When the CPMS was in operation, an attempt was made to execute a floating-point instruction.
5	EC=03390000	FP System Down	Floating-point calculation error	An error occurred in a floating-point instruction.
6	EC=03400000	Instruction Page Fault	Instruction access page fault	An instruction access was made to a page whose address is not in the page table.
7	EC=03470000	Data Alignment Error	Data alignment error	The accessed data was not word aligned nor long word aligned.
8	EC=03600000	Data Page Fault	Data access page fault	A data access was made to a page whose address is not in the page table.
9	EC=03660000	Data Access Protection	Data access protection error	A data access infringed memory protection.
10	EC=03820000	Memory Error	Memory error	The hardware detected a memory error.
11	EC=03820001	Memory Error (MRAM)	MRAM memory error	The hardware detected an MRAM memory error.
12	EC=03B60000	RI/O-IF Module Error	An RI/O-IF module error was detected	A serious fault interrupt was detected in the RI/O-IF module.
13	EC=03B80000	R700/S10 Bus Error	System bus error	An invalid serious fault interrupt was detected in the system bus 10 times in a row.
14	EC=03B80001	System Bus Error (CPU Master)	System bus error (PU is the master)	A serious fault interrupt was detected in the system bus system (PU is the master).
15	EC=03B90000	PCI Bus Error	PCI bus error	A serious fault interrupt was detected in the PCI bus system.
16	EC=03BD0000	LSI Internal Timeout Error	LSI internal timeout error	The hardware detected an LSI internal timeout.
17	EC=03BE0000	SPU Error	SPU error	The hardware detected an SPU error.
18	EC=03BF0000	RI/O Error	RI/O error	The hardware detected an RI/O error.
19	EC=0500F001	HERST Invalid Interrupt	Invalid serious fault interrupt detected (10 times) in a row	An invalid serious fault interrupt was detected 10 times in a row.
20	EC=0500F003	BUERRSTAT Invalid Interrupt	Invalid serious fault interrupt detected on PCI bus (two times) in a row	An invalid serious fault interrupt was detected on the PCI bus two times in a row.

Table E-30 Error Codes, Subtitles, and their Descriptions (System Error) (2/2)

No.	Error code	Subtitle	Description	Explanation
21	EC=0500F004	P2NHERRQ Invalid Interrupt	Invalid serious fault interrupt detected between the CP and NB (two times) in a row	An invalid serious fault interrupt was detected between the CP and NP two times in a row.
22	EC=0500F005	N2PHERRQ Invalid Interrupt	Invalid serious fault interrupt detected between the NP and CP (two times) in a row	An invalid serious fault interrupt was detected between the NP and CP two times in a row.
23	EC=0500F00B	NP_ERRLOGMP Invalid Interrupt	NPU invalid serious fault interrupt detected	An invalid serious fault interrupt was detected in the NPU two times in a row.
24	EC=0500F00E	DCM HERR Invalid Interrupt	DCM invalid serious fault interrupt	An invalid serious fault interrupt was detected in the DCM two times in a row.
25	EC=05700000	System Error	CPMS error	An error preventing the CPMS from continuing operation was detected.
26	EC=05700001	CP Inloop Detect	OS infinite loop detected for the CP	The OS on the CP side terminated due to a detected infinite loop.
27	EC=05700002	HP Inloop Detect	OS infinite loop detected for the HP	The OS on the HP side terminated due to a detected infinite loop.
28	EC=05900000	CP Down	OS down detected for the CP	The OS on the HP side detected the termination of the OS on the CP side and terminated.
29	EC=05900001	HP Down	OS down detected for the HP	The OS on the CP side detected the termination of the OS on the HP side and terminated.
30	EC=0D010001	Memory Patrol Error	Memory patrol error	An error was detected in the memory patrol.
31	EC=0D810000	BPU Error	BPU miscalculation detected	The CPMS detected a BPU miscalculation.

E.3.9 System down (kernel trap)

This error indicates that an irrationality was detected in the CPMS while the CPMS was in operation.

Table E-31 shows and describes the error message.

Table E-31 Format of the Kernel Trap Message

```
CPU xxxxxxxx
%CPMS-F-xxxx-000a  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=05800000 System down (Kernel Trap)
FILE =xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
LINE =xxxxxxxx
ERROR=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Item	Description
FILE	Name of the internal file of the CPMS in which an irrationality was detected
LINE	Location of the internal file of the CPMS in which an irrationality was detected
ERROR	Conditional expression that leads to irrationality

E.3.10 System down (built-in subroutine error)

This error indicates that an abnormality was detected that prevented the CPMS from continuing processing while a built-in subroutine was running. Table E-32 shows and describes the error message.

Table E-32 Format of the System Down (Built-in Subroutine Error) Message (1/2)

```
CPU xxxxxxxx
%CPMS-F-SOFT-000b  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx ULSUB down (subtitle)
NEST =xxxxxxxx  POINT=xxxxxxxx  ENTRY=xxxxxxxx
PC   =xxxxxxxx  EXPEV=xxxxxxxx  FADR =xxxxxxxx  SR   =xxxxxxxx
PR   =xxxxxxxx  SP   =xxxxxxxx  GBR  =xxxxxxxx  MACH =xxxxxxxx  MACL =xxxxxxxx
R0   =xxxxxxxx  R1   =xxxxxxxx  R2   =xxxxxxxx  R3   =xxxxxxxx  R4   =xxxxxxxx
R5   =xxxxxxxx  R6   =xxxxxxxx  R7   =xxxxxxxx  R8   =xxxxxxxx  R9   =xxxxxxxx
R10  =xxxxxxxx  R11  =xxxxxxxx  R12  =xxxxxxxx  R13  =xxxxxxxx  R14  =xxxxxxxx
R15  =xxxxxxxx
INST =xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
(PC =)xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
STACK=xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
(SP =)xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
PC  =xxxxxxxx
FADR=xxxxxxxx
PR  =xxxxxxxx
```

Item	Description
EC	Error code (see Table E-33)
NEST	Nest count of the built-in subroutine
POINT	Point number of the built-in subroutine
ENTRY	Entry number of the built-in subroutine
PC	Contents of the program counter
EXPEV	Contents of the exception code register. The exception code register is a 32-bit register defining the causes of data access exceptions and data alignment exceptions.
FADR	Fault address
SR	Status register. This register defines the state of the processor.
PR	Contents of the procedure register. The procedure register is used for subroutine calls. If the executed program was the last program in the subroutine calling sequence, this register contains the return address.
SP	Contents of the stack pointer (R15 is used as a stack pointer).
GBR	Contents of the global base register. This register contains the base addresses for GBR-indirect addressing with displacement and indexed GBR-indirect addressing.

Table E-32 Format of the System Down (Built-in Subroutine Error) Message (2/2)

Item	Description
MACH	MAC register. This register is used for storing additional values of MAC (multiply and accumulate operation) instructions, MAC instructions, and results of MUL instructions. If the calculated result is a value of 64 bits or more, this register stores the upper 32 bits.
MACL	MAC register. If the calculated result is a 64-bit value, this register stores the lower 32 bits. If the calculated result was a 32-bit value, this register stores the 32 bits.
R*xx*	Contents of the general register represented by the value at *xx*
INST	Eight instructions preceding and following the PC address
PC	Instruction of the PC
STACK	Eight long word data items preceding and following the SP address
SP	Contents of the SP address (contents of the stack)
PC	Address information contained in the program counter is displayed in parentheses. When the address is for a program, the following information is displayed: name = *program name*, type = *program type (program location)*, raddr = *relative address from the program*
FADR	Information about the fault address is displayed in parentheses.
PR	Information about the address contained in a procedure register is displayed in parentheses. When the address is for a program, the following information is displayed: name = *program name*, type = *program type (program location)*, raddr = *relative address from the program*

Table E-33 Error Codes, Subtitles, and their Descriptions (Built-in Subroutine Error)

No.	Error code	Subtitle	Description	Explanation
1	EC=03030000	Inst. Alignment Error	Instruction alignment error	Operands specified by the instruction are not word-aligned.
2	EC=03040000	Illegal Instruction	Illegal instruction error	An attempt was made to execute an illegal instruction.
3	EC=030F0000	Illegal Exception	Illegal instruction	An undefined exception was reported.
4	EC=03380000	FP Unavailable	Floating-point unavailable	An attempt was made to execute a floating-point instruction when a built-in subroutine was running.
5	EC=03390000	FP System Down	Floating-point calculation error	An error occurred in a floating-point instruction.
6	EC=03400000	Instruction Page Fault	Instruction access page fault	An instruction access was made to a page whose address is not in the page table.
7	EC=03470000	Data Alignment Error	Data alignment error	The accessed data was neither word-aligned nor long word-aligned.
8	EC=03600000	Data Page Fault	Data access page fault	A data access was made to a page whose address is not in the page table.
9	EC=03660000	Data Access Protection	Data access protection error	A data access infringed memory protection.

E.3.11 System down (built-in subroutine stop)

This error indicates that a built-in subroutine terminated with the return value of the CPU termination request.

Table E-34 shows and describes the error message.

Table E-34 Format of the System Down (Built-in Subroutine Stop) Message

```
CPU xxxxxxxx
%CPMS-F-SOFT-000c  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=05140000 System down (ULSUB Stop)
NEST =xxxxxxxx  POINT=xxxxxxxx
RET  =xxxxxxxx  RET0 =xxxxxxxx  RET1 =xxxxxxxx  RET2 =xxxxxxxx  RET3 =xxxxxxxx
SUBFM=xxxxxxxx  SUBSZ=xxxxxxxx  SUBEC=xxxxxxxx
DAT0  =xxxxxxxx DAT1  =xxxxxxxx DAT2  =xxxxxxxx DAT3  =xxxxxxxx DAT4  =xxxxxxxx
DAT5  =xxxxxxxx DAT6  =xxxxxxxx DAT7  =xxxxxxxx DAT8  =xxxxxxxx DAT9  =xxxxxxxx
DAT10 =xxxxxxxx DAT11 =xxxxxxxx DAT12 =xxxxxxxx DAT13 =xxxxxxxx DAT14 =xxxxxxxx
DAT15 =xxxxxxxx DAT16 =xxxxxxxx DAT17 =xxxxxxxx DAT18 =xxxxxxxx DAT19 =xxxxxxxx
DAT20 =xxxxxxxx DAT21 =xxxxxxxx DAT22 =xxxxxxxx DAT23 =xxxxxxxx DAT24 =xxxxxxxx
DAT25 =xxxxxxxx DAT26 =xxxxxxxx DAT27 =xxxxxxxx DAT28 =xxxxxxxx DAT29 =xxxxxxxx
DAT30 =xxxxxxxx DAT31 =xxxxxxxx DAT32 =xxxxxxxx DAT33 =xxxxxxxx DAT34 =xxxxxxxx
DAT35 =xxxxxxxx DAT36 =xxxxxxxx DAT37 =xxxxxxxx DAT38 =xxxxxxxx DAT39 =xxxxxxxx
```

(Other data might be displayed depending on the error.)

Item	Description
NEST	Nest counter for the built-in subroutine
POINT	Point number of the built-in subroutine
RET	Termination information for the built-in subroutine
RET*n*	Termination information for the built-in subroutine entries numbered from 0 to *n*
SUBFM	Format type that was passed to the built-in subroutine as an argument
SUBSZ	Byte count of data that was passed to the built-in subroutine as an argument
SUBEC	Sub error code: An error code that identifies the factor causing the built-in subroutine error
DAT*n*	Data for each type indicated by SUBFM

E.3.12 Memory error

This error indicates that a serious fault (memory multibit error) occurred in the memory system.

Table E-35 shows and describes the error message.

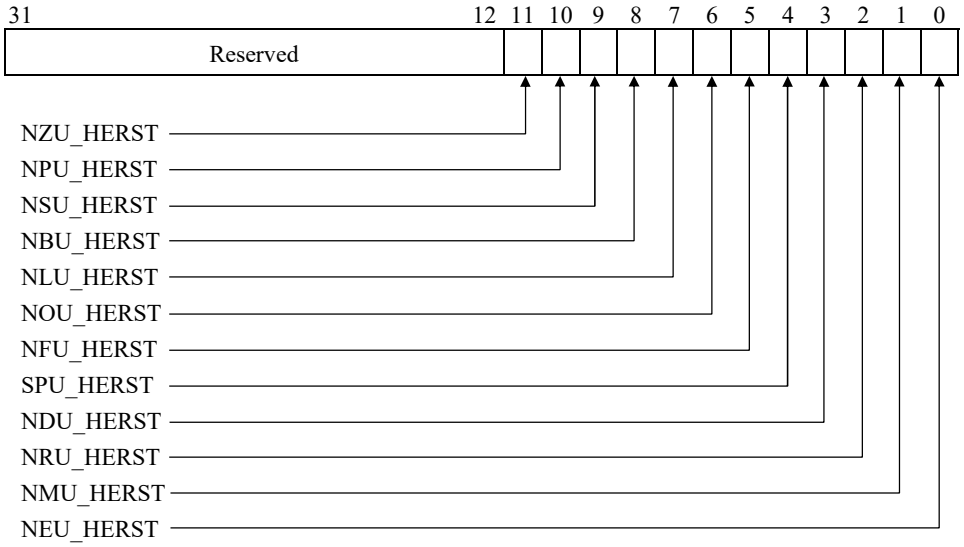
Table E-35 Format of the Memory Error Message (1/21)

```
CPU xxxxxxxx
%CPMS-E-HARD-0012  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=03820000 Memory Error
TN  =xxxxxxxx  PC  =xxxxxxxx  EXPEV=xxxxxxxx  SR  =xxxxxxxx
PR  =xxxxxxxx  SP  =xxxxxxxx  GBR  =xxxxxxxx  MACH =xxxxxxxx  MACL =xxxxxxxx
R0  =xxxxxxxx  R1  =xxxxxxxx  R2  =xxxxxxxx  R3  =xxxxxxxx  R4  =xxxxxxxx
R5  =xxxxxxxx  R6  =xxxxxxxx  R7  =xxxxxxxx  R8  =xxxxxxxx  R9  =xxxxxxxx
R10 =xxxxxxxx  R11 =xxxxxxxx  R12 =xxxxxxxx  R13 =xxxxxxxx  R14 =xxxxxxxx
R15 =xxxxxxxx  FPSCR=xxxxxxxx  FPUL =xxxxxxxx
FR0 = x.xxxxxxE+xxx FR1 = x.xxxxxxE+xxx FR2 = x.xxxxxxE+xxx FR3 = x.xxxxxxE+xxx
FR4 = x.xxxxxxE+xxx FR5 = x.xxxxxxE+xxx FR6 = x.xxxxxxE+xxx FR4 = x.xxxxxxE+xxx
FR8 = x.xxxxxxE+xxx FR9 = x.xxxxxxE+xxx FR10= x.xxxxxxE+xxx FR11= x.xxxxxxE+xxx
FR12= x.xxxxxxE+xxx FR13= x.xxxxxxE+xxx FR14= x.xxxxxxE+xxx FR15= x.xxxxxxE+xxx
XF0 = x.xxxxxxE+xxx XF1 = x.xxxxxxE+xxx XF2 = x.xxxxxxE+xxx XF3 = x.xxxxxxE+xxx
XF4 = x.xxxxxxE+xxx XF5 = x.xxxxxxE+xxx XF6 = x.xxxxxxE+xxx XF7 = x.xxxxxxE+xxx
XF8 = x.xxxxxxE+xxx XF9 = x.xxxxxxE+xxx XF10= x.xxxxxxE+xxx XF11= x.xxxxxxE+xxx
XF12= x.xxxxxxE+xxx XF13= x.xxxxxxE+xxx XF14= x.xxxxxxE+xxx XF14= x.xxxxxxE+xxx
INST =xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
      xxxxxxxx   (PC =) xxxxxxxx xxxxxxxx
HERST  =xxxxxxxx  STATUS  =xxxxxxxx  DETADR  =xxxxxxxx  DETDAT  =xxxxxxxx
STAT   =xxxxxxxx  LOGMST  =xxxxxxxx  LOGSLV  =xxxxxxxx  RERRADR =xxxxxxxx
RERRDAT =xxxxxxxx  RERRMST =xxxxxxxx  RERRCMD =xxxxxxxx  NLUSTS  =xxxxxxxx
NOERRST =xxxxxxxx  NZERRST =xxxxxxxx  PCIHER  =xxxxxxxx  PCIAHER =xxxxxxxx
PCIBMLR =xxxxxxxx  DCMSTS  =xxxxxxxx  SPERRS  =xxxxxxxx  NEERRST =xxxxxxxx
NPERRLOG=xxxxxxxx  NMHERST =xxxxxxxx  MSW2   =xxxxxxxx
PC  =xxxxxxxx
SR  =xxxxxxxx
PR  =xxxxxxxx
```

Table E-35 Format of the Memory Error Message (2/21)

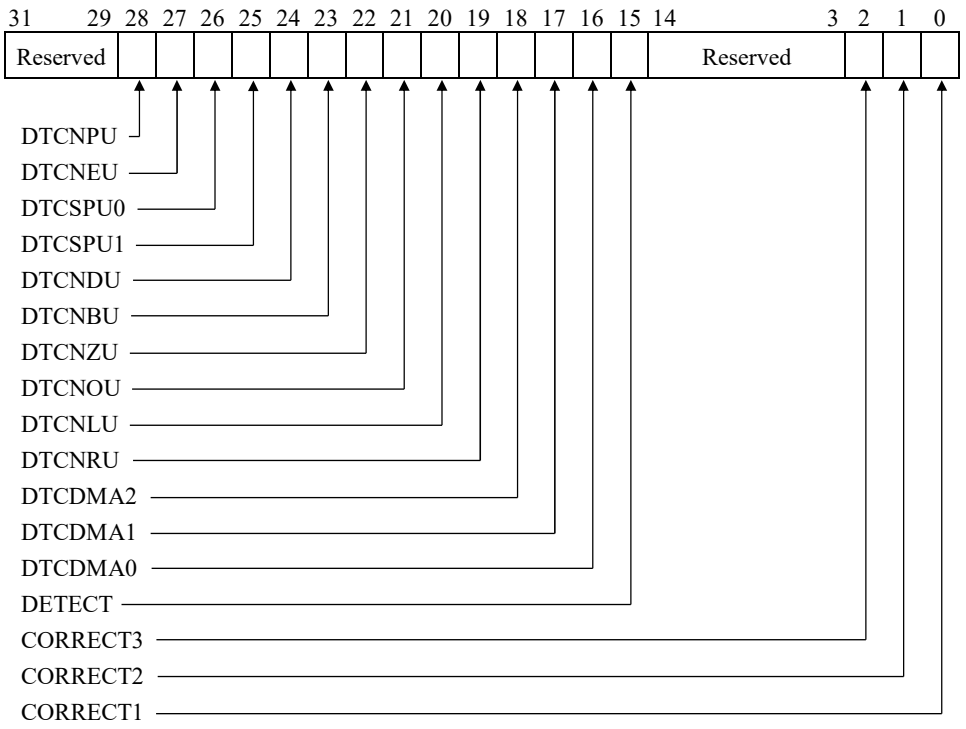
Item	Description
TN	Task number of the task at which an error occurred
PC	Contents of the program counter
EXPEV	Contents of the exception code register
SR	Contents of the status register
PR	Content of the procedure register. The procedure register is used for subroutine calls. If the executed program was the last program in the subroutine calling sequence, this register contains the return address.
SP	Contents of the stack pointer (R15 is used as a stack pointer).
GBR	Contents of the global base register. This register is used for storing the base addresses for GBR-indirect addressing with displacement and indexed GBR-indirect addressing.
MACH	MAC register. This register is used for storing additional values of MAC (multiply and accumulate operation) instructions, MAC instructions, and results of MUL instructions. If the calculated result is a value of 64 bits, this register stores the upper 32 bits.
MACL	MAC register. If the calculated result is a 64-bit value, this register stores the lower 32 bits. If the calculated result was a 32-bit value, this register stores the 32 bits.
R*xx*	Contents of the general register represented by the value at *xx*
FPSCR	Contents of the floating-point status and control register
FPUL	Contents of the floating-point communication register. Data is transferred between the general register and floating-point register through this register.
FR*xx*	Contents of the 32-bit floating-point register represented by the value at *xx*. When FPSCR.FR (bit 21 of a 31-0 bit value) is 0, this is the value of FPR*xx*_BANK0. When FPSCR.FR is 1, this is the value of FPR*xx*_BANK1.
XF*xx*	Contents of the 32-bit floating-point register *xx*. When FPSCR.FR (bit 21 of 31-0 bit value) is 0, this is the value of FPR*xx*_BANK1. When FPSCR.FR is 1, this is the value of FPR*xx*_BANK0.
INST	Instruction code

Table E-35 Format of the Memory Error Message (3/21)

Item	Description
HERST	

Bit	Name	Set value	Explanation
31 to 12	Reserved	–	–
11	NZU_HERST	1: Error 0: Normal	This indicates that an NZU serious fault occurred relating to MP core 0 (core1).
10	NPU_HERST	1: Error 0: Normal	This indicates that an NPU serious fault occurred relating to MP core 0 (core1).
9	NSU_HERST	1: Error 0: Normal	This indicates that an NSU serious fault occurred relating to MP core 0 (core1).
8	NBU_HERST	1: Error 0: Normal	This indicates that an NBU serious fault occurred relating to MP core 0 (core 1)
7	NLU_HERST	1: Error 0: Normal	This indicates that an NLU serious fault occurred relating to MP core 0 (core1).
6	NOU_HERST	1: Error 0: Normal	This indicates that an NOU serious fault occurred relating to MP core 0 (core1).
5	NFU_HERST	1: Error 0: Normal	This indicates that an NFU serious fault occurred relating to MP core 0 (core1).
4	SPU_HERST	1: Error 0: Normal	This indicates that an SPU serious fault occurred relating to MP core 0 (core1).
3	NDU_HERST	1: Error 0: Normal	This indicates that an NDU serious fault occurred relating to MP core 0 (core1).
2	NRU_HERST	1: Error 0: Normal	This indicates that an NRU serious fault occurred relating to MP core 0 (core1).
1	NMU_HERST	1: Error 0: Normal	This indicates that an NMU serious fault occurred relating to MP core 0 (core1).
0	NEU_HERST	1: Error 0: Normal	This indicates that an NEU serious fault occurred relating to MP core 0.

Table E-35 Format of the Memory Error Message (4/21)

Item	Description
STATUS	

Bit	Name	Set value	Explanation
31 to 29	Reserved	–	–
28	DTCNPU	1: Error 0: Normal	This indicates that a 4-bit error due to an NPU read access was detected
27	DTCNEU	1: Error 0: Normal	This indicates that a 4-bit error due to an NEU read access was detected.
26	DTCSPU0	1: Error 0: Normal	This indicates that a 4-bit error due to an SPU0 read access was detected.
25	DTCSPU1	1: Error 0: Normal	This indicates that a 4-bit error due to an SPU1 read access was detected.
24	DTCNDU	1: Error 0: Normal	This indicates that a 4-bit error due to an NDU read access was detected.
23	DTCNBU	1: Error 0: Normal	This indicates that a 4-bit error due to an NBU read access was detected.
22	DTCNZU	1: Error 0: Normal	This indicates that a 4-bit error due to an NZU read access was detected.
21	DTCNOU	1: Error 0: Normal	This indicates that a 4-bit error was detected due to an NPU read access.
20	DTCNLU	1: Error 0: Normal	This indicates that a 4-bit error due to an NLU read access was detected.
19	DTCNRU	1: Error 0: Normal	This indicates that a 4-bit error due to an NRU read access was detected.
18	DTCDMA2	1: Error 0: Normal	This indicates that a 4-bit error due to a DMA (ch2) read access was detected.
17	DTCDMA1	1: Error 0: Normal	This indicates that a 4-bit error due to a DMA (ch1) read access was detected.

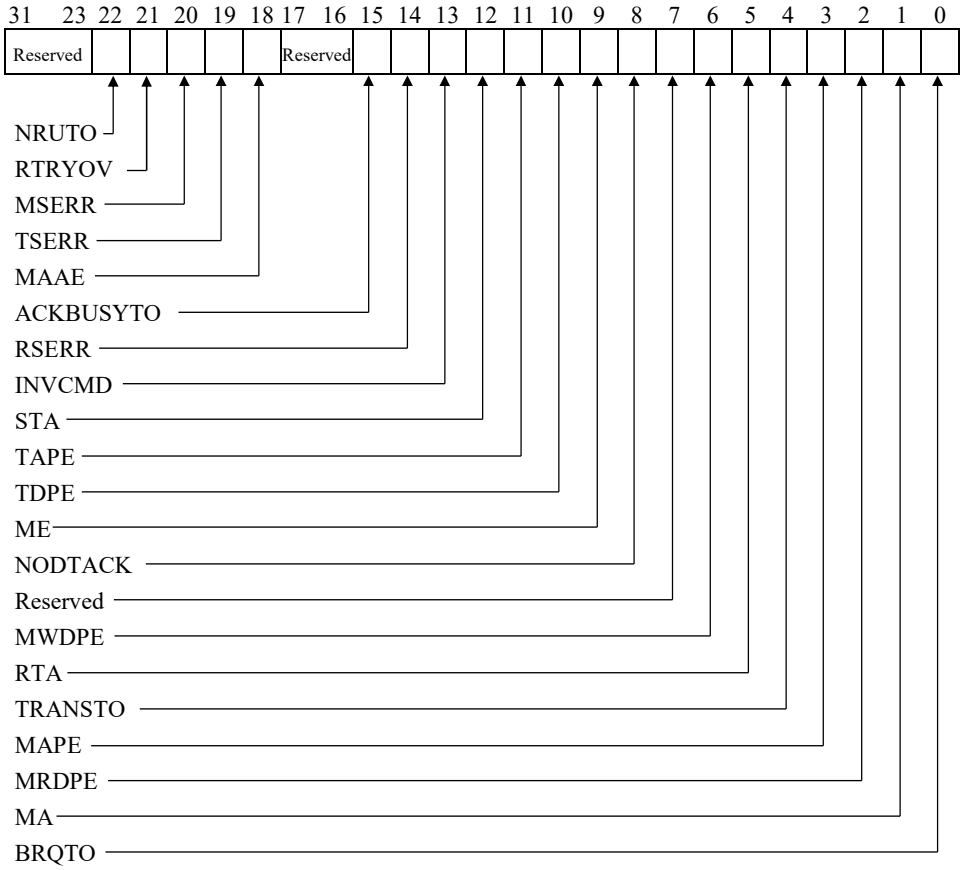
Table E-35 Format of the Memory Error Message (5/21)

<table>
<tr><th>Item</th><th>Description</th></tr>
<tr><td>STATUS
(continued from the preceding page)</td><td>
<table>
<tr><th>Bit</th><th>Name</th><th>Set value</th><th>Explanation</th></tr>
<tr><td>16</td><td>DTCDMA0</td><td>1: Error
0: Normal</td><td>This indicates that a 4-bit error due to a DMA (ch0) read access was detected.</td></tr>
<tr><td>15</td><td>DETECT</td><td>1: Error
0: Normal</td><td>This indicates that a 4-bit error was detected.</td></tr>
<tr><td>14 to 3</td><td>Reserved</td><td>–</td><td>–</td></tr>
<tr><td>2</td><td>CORRECT3</td><td>1: Error
0: Normal</td><td>This indicates that a 3-bit error correction occurred.</td></tr>
<tr><td>1</td><td>CORRECT2</td><td>1: Error
0: Normal</td><td>This indicates that a 2-bit error correction occurred.</td></tr>
<tr><td>0</td><td>CORRECT1</td><td>1: Error
0: Normal</td><td>This indicates that a 1-bit error correction occurred.</td></tr>
</table>
</td></tr>
<tr><td>DETADR</td><td>


<table>
<tr><th>Bit no.</th><th>Bit name</th><th>Read-in value</th><th>Explanation</th></tr>
<tr><td>31 to 0</td><td>DET_ADDR</td><td>Option</td><td>Address when a serious fault occurred in the memory system</td></tr>
</table>
</td></tr>
<tr><td>DETDAT</td><td>


<table>
<tr><th>Bit no.</th><th>Bit name</th><th>Read-in value</th><th>Explanation</th></tr>
<tr><td>31 to 0</td><td>DET_DATA</td><td>Option</td><td>Address when a serious fault occurred in the memory system</td></tr>
</table>
</td></tr>
</table>

Table E-35 Format of the Memory Error Message (6/21)

Item	Description
STAT	

Bit	Name	Set value	Explanation
31 to 23	Reserved	–	–
22	NRUTO	1: Error 0: Normal	Internal TO error
21	RTRYOV	1: Error 0: Normal	The retry limit was exceeded. This indicates that the device acting as bus master exceeded the retry frequency limit specified by the NR_RRTRYMAX register.
20	MSERR	1: Error 0: Normal	The device acting as bus master received an SERROR. This indicates that the device acting as bus master received an SERROR signal.
19	TSERR	1: Error 0: Normal	The R700 acting as a target received an SERROR. This indicates that when an R700 was acting as a target, it received an SERROR signal.
18	MAAE	1: Error 0: Normal	A misalignment access error was detected. The following error was detected: The R700 acting as a target detected an invalid byte-enable (BE) in a writing process.
17 to 16	Reserved	–	–

Table E-35 Format of the Memory Error Message (7/21)

Item	Description

STAT (continued from the preceding page)

Bit	Name	Set value	Explanation
15	ACKBUSYTO	1: Error 0: Normal	A BGACK busy TO was detected. This indicates that the arbiter detected a BGACK assertion timeout. (It detected that the assertion did not end.)
14	RSERR	1: Error 0: Normal	An SERROR was received.
13	INVCMD	1: Error 0: Normal	An invalid command was detected. This indicates that the R700 acting as a target terminated a bus operation due to an error transaction.
12	STA	1: Error 0: Normal	An error transaction was transmitted. This indicates that the R700 acting as a bus target transmitted an error transaction as a response.
11	TAPE	1: Error 0: Normal	An address parity error was detected. This indicates that the R700 acting as a target detected an address parity error (APE).
10	TDPE	1: Error 0: Normal	A write parity error was detected. This indicates that the R700 acting as a target detected a write data parity error (WDPE).
9	ME	1: Error 0: Normal	A memory error was detected. This indicates that the target performing read transaction through the R700/S10 bus detected a 4bitECC error in the memory.
8	NODTACK	1: Error 0: Normal	A data ready signal was not detected. This indicates that a timeout was detected during monitoring of the period between when the S10 bus started and when the data became ready for communication (DTACK assertion). This resulted in the CPU asserting DTACK instead.
7	Reserved	–	–
6	MWDPE	1: Error 0: Normal	The device acting as bus master received a WDPE. This indicates that when the device was acting as bus master, a target detected a write data parity error (WDPE) and the master received the error notification from the target.
5	RTA	1: Error 0: Normal	An error transaction was received. This indicates that the device acting as bus master stopped its bus operation due to an error transaction.
4	TRANSTO	1: Error 0: Normal	A transaction TO was detected. This indicates that the device acting as bus master detected a transaction timeout (TO) and terminated its bus operation.
3	MAPE	1: Error 0: Normal	The device acting as bus master received an APE. This indicates that the device acting as bus master received an address parity error (APE) notification.

Table E-35 Format of the Memory Error Message (8/21)

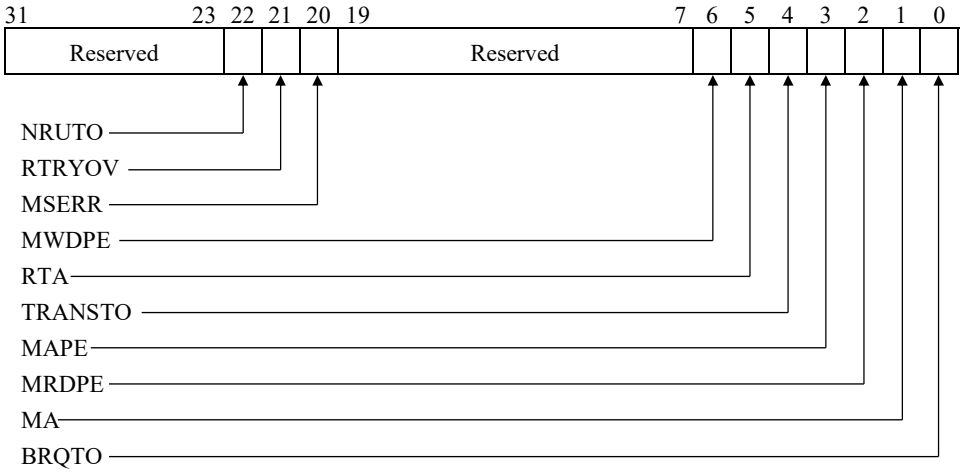
<table>
<tr><th>Item</th><th>Description</th></tr>
<tr><td>STAT
(continued from the preceding page)</td><td>
<table>
<tr><th>Bit</th><th>Name</th><th>Set value</th><th>Explanation</th></tr>
<tr><td>2</td><td>MRDPE</td><td>1: Error
0: Normal</td><td>The device acting as bus master received an RDPE.
This indicates that the device acting as bus master detected a read-data parity error (RDPE).</td></tr>
<tr><td>1</td><td>MA</td><td>1: Error
0: Normal</td><td>An address cycle TO was detected.
This indicates that the device acting as bus master detected an address cycle timeout (TO) and terminated its bus operation.</td></tr>
<tr><td>0</td><td>BRQTO</td><td>1: Error
0: Normal</td><td>A bus request TO was detected.
This indicates that the device acting as bus master detected a bus request timeout (TO) and terminated its bus operation.</td></tr>
</table>
</td></tr>
<tr><td>LOGMST</td><td> 
</td></tr>
</table>

Table E-35 Format of the Memory Error Message (9/21)

Item	Description
LOGMST (continued from the preceding page)	

Bit	Name	Set value	Explanation
31 to 23	Reserved	–	–
22	NRUTO	1: Error 0: Normal	Internal TO error
21	RTRYOV	1: Error 0: Normal	The retry limit was exceeded. This indicates that the device acting as bus master exceeded the retry frequency limit specified by the NR_RRTRYMAX register.
20	MSERR	1: Error 0: Normal	The device acting as bus master received an SERROR. This indicates that the device acting as bus master received an SERROR signal.
19 to 7	Reserved	–	–
6	MWDPE	1: Error 0: Normal	The device acting as bus master received a WDPE. This indicates that the following error occurred: when the device was acting as bus master, a target detected a write data parity error (WDPE) and the master received an error notification from the target.
5	RTA	1: Error 0: Normal	An error transaction was received. This indicates that the device acting as bus master terminated its bus operation due to an error transaction.
4	TRANSTO	1: Error 0: Normal	A transaction TO was detected. This indicates that the device acting as bus master detected a transaction timeout (TO) and terminated its bus operation.
3	MAPE	1: Error 0: Normal	The device acting as bus master received APE. This indicates that the device acting as bus master received an address parity error (APE) notification.
2	MRDPE	1: Error 0: Normal	The device acting as bus master received RDPE. This indicates that the device acting as bus master detected a read data parity error (RDPE).
1	MA	1: Error 0: Normal	An address cycle TO was detected. This indicates that the device acting as bus master detected an address cycle timeout (TO) and terminated its bus operation.
0	BRQTO	1: Error 0: Normal	A bus request TO was detected. This indicates that the device acting as bus master detected a bus request timeout (TO), and terminated its bus operation.

Table E-35 Format of the Memory Error Message (10/21)

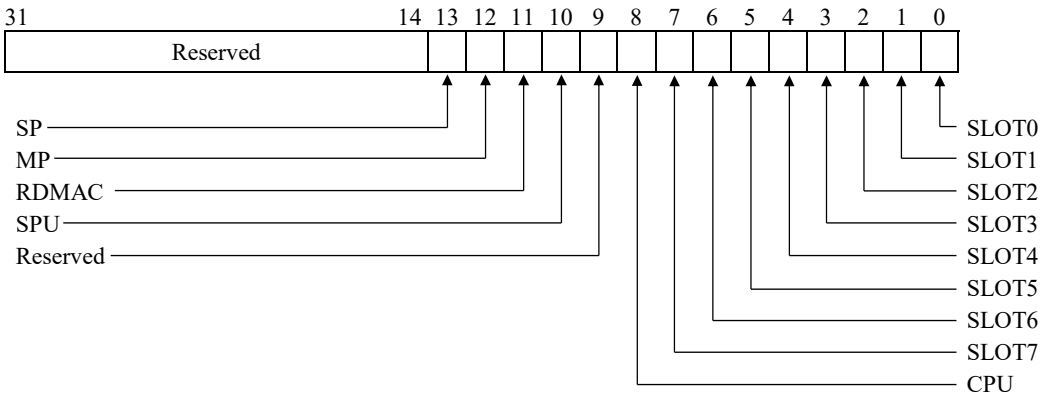
RERRADR

Bit no.	Bit name	Read-in value	Explanation
31 to 0	ADDR	Option	Address when the bus error occurred

RERRDAT

Bit no.	Bit name	Read-in value	Explanation
31 to 0	DAT	Option	Address when the bus error occurred

RERRMST

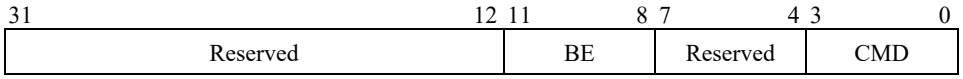


Bit	Name	Set value	Explanation
31 to 14	Reserved	–	–
13	SP	1: Master was SP when error occurred 0: Master was not SP when error occurred	SP master error log This indicates that the error factor recorded in the error log occurred when the SP was the master.
12	MP	1: Master was MP when error occurred 0: Master was not MP when error occurred	MP master error log This indicates that the error factor recorded in the error log occurred when the MP was the master.
11	RDMAC	1: Master was RDMAC when error occurred 0: Master was not RDMAC when error occurred	RDMAC master error log This indicates that the error factor recorded in the error log occurred when the RDMAC was the master.
10	SPU	1: Master was SPU when error occurred 0: Master was not SPU when error occurred	SPU master error log This indicates that the error factor recorded in the error log occurred when the SPU was the master.
9	Reserved	–	–
8	CPU	1: Master was CPU when error occurred 0: Master was not CPU when error occurred	CPU master error log This indicates that the error factor recorded in the error log occurred when the CPU was the master.
7 to 0	SLOT (7 to 0)	1: Master was SLOT (7 to 0) when error occurred 0: Master was not SLOT (7 to 0) when error occurred	SLOT (7 to 0) master error log This indicates that the error factor recorded in the error log occurred when the SLOT (7 to 0) was the master.

Table E-35 Format of the Memory Error Message (11/21)

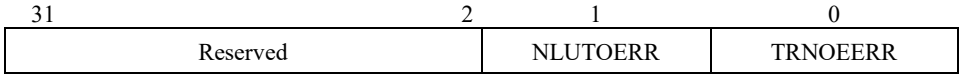
Item	Description

RERRCMD



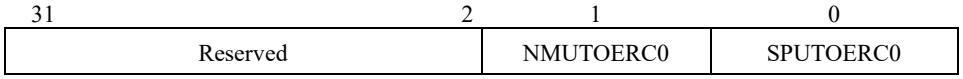
Bit	Name	Set value	Explanation
31 to 12	Reserved	–	–
11 to 8	BE	1: Error occurred when NR_RERRLOGC0 register was set 0: Other	Byte-enable error log This indicates the byte-enable that was used when the error factor recorded in the error log occurred. When the S10 bus operates, this value is a concatenation of *the 2-bit 0, UDS, and LDS*.
7 to 4	Reserved	–	–
3 to 0	CMD	1: Error occurred when NR_RERRLOGC0 register was set 0: Other	Bus command error log This indicates that the bus command that was used when the error factor recorded in the error log occurred. When the S10 bus operates, this value is a concatenation of *the 3-bit 0 (1 for writing/0 for reading)*.

NLUSTS



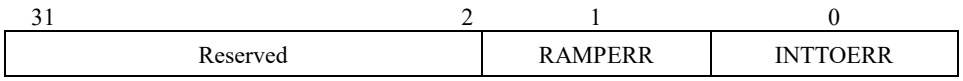
Bit	Name	Set value	Explanation
31 to 2	Reserved	–	–
1	NLUTOERR	1: Error 0: Normal	This indicates whether there is a factor that caused the internal timeout error.
0	TRNOEERR	1: Error 0: Normal	This indicates whether the transceiver OE error occurred in system 1 or 2.

NOERRST



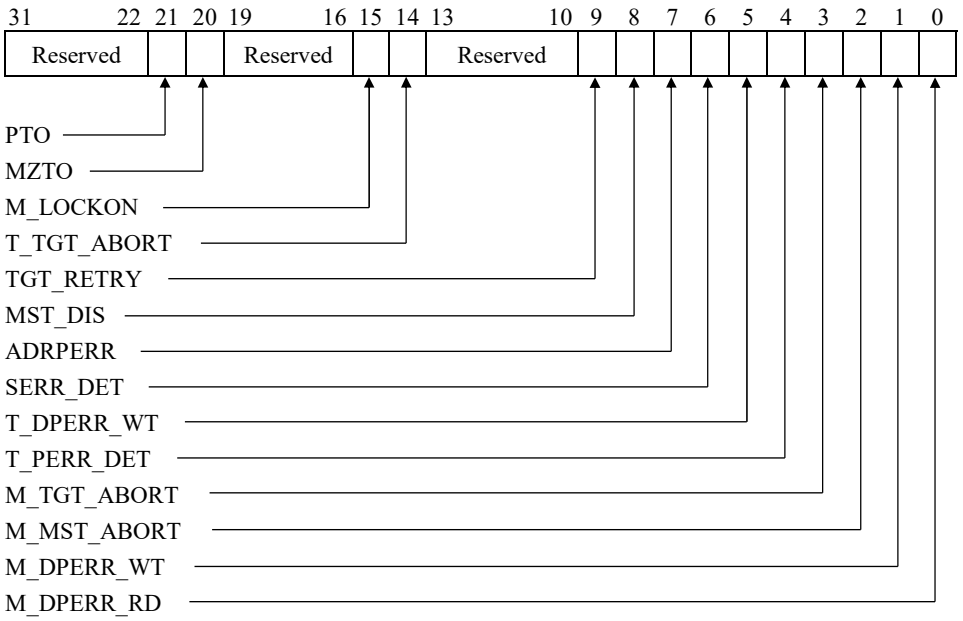
Bit	Name	Set value	Explanation
31 to 2	Reserved	–	–
1	NMUTOERC0	1: Error 0: Normal	This indicates that an NMU access caused the TO error.
0	SPUTOERC0	1: Error 0: Normal	This indicates that the cache access in the SPU caused the timeout error.

NZERRST



Bit	Name	Set value	Explanation
31 to 2	Reserved	–	–
1	RAMPERR	1: Error 0: Normal	This sets a RAM parity error.
0	INTTOERR	1: Error 0: Normal	This sets an internal timeout error.

Table E-35 Format of the Memory Error Message (12/21)

Item	Description
PCIHER	

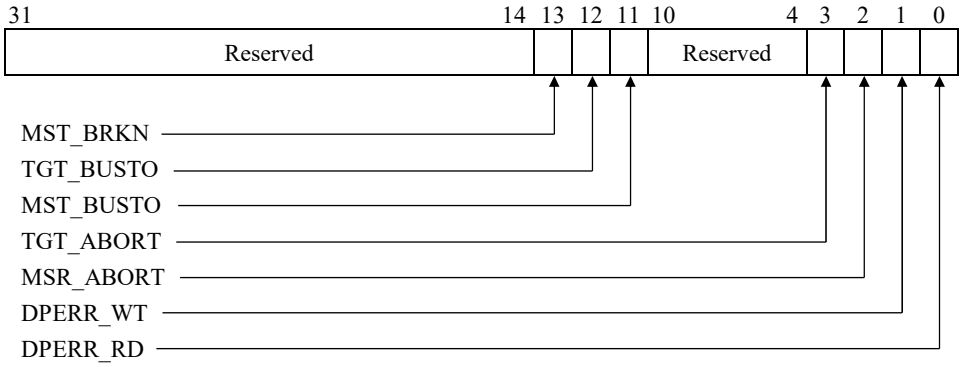
Bit	Name	Set value	Explanation
31 to 22	Reserved	–	–
21	PTO	1: Error 0: Normal	This detects that the NBU could not get bus ownership within the specified period after the NBU asserted a bus request to the PCI host.
20	MZTO	1: Error 0: Normal	This indicates that an ACK was not asserted within the specified period after the NBU asserted an REQ to the internal block (NMU and NZU).
19 to 16	Reserved	–	–
15	M_LOCKON	1: Error 0: Normal	This indicates that the following error occurred: when the NBU was an initiator, it unlocked and transferred a specific target that had been locked. Exclusive control by using LOCK.
14	T_TGT_ABORT	1: Error 0: Normal	This indicates that the following error occurred: When the NBU was a target, it terminated a transaction by using a target abort. A target abort is issued if during I/O transfer, the combination of the last 2 bits of the address and the byte-enable was invalid.
13 to 10	Reserved	–	–
9	TGT_RETRY	1: Error 0: Normal	This indicates that the following error occurred: When the NBU was a target, the initiator did not perform retry processing within the 2^{15} PCI clock limit after the target gave a retry response. This error is detected only for memory-read transfers.
8	MST_DIS	1: Error 0: Normal	This indicates that the following error occurred: The NBU acted as initiator (performed PI/O transfer) even though it did not have initiator functionality and bit 2 of PCICONF1 was set to 0.
7	ADRPERR	1: Error 0: Normal	This indicates that an address parity error occurred. This error is detected only when both bits 6 and 8 of PCICONF1 are set to 1.

Table E-35 Format of the Memory Error Message (13/21)

Item	Description
PCIHER (continued from the preceding page	

Bit	Name	Set value	Explanation
6	SERR_DET	1: Error 0: Normal	This indicates that when the NBU was the host, an SERR_N was asserted.
5	T_DPERR_WT	1: Error 0: Normal	This indicates that when the NBU was a target, it detected a parity error during a target-write process. This error is detected only when bit 6 of PCICONF1 is set to 1.
4	T_PERR_DET	1: Error 0: Normal	This indicates that when the NBU was a target, it accepted a PERR_N assert during a target-read process. This error is detected only when bit 6 of PCICONF1 is set to 1.
3	M_TGT_ABORT	1: Error 0: Normal	This indicates that the following error occurred: When the NBU was an initiator, the transaction was terminated by using a target abort.
2	M_MST_ABORT	1: Error 0: Normal	This indicates that the following error occurred: When the NBU was an initiator, it terminated the transaction by using a master abort.
1	M_DPERR_WT	1: Error 0: Normal	This indicates that the NBU acting as initiator accepted a PERR_N assert while writing data to a target. This error is detected only when bit 6 of PCICONF1 is set to 1.
0	M_DPERR_RD	1: Error 0: Normal	This indicates that the NBU acting as initiator detected a parity error while reading data from a target. This error is detected only when bit 6 of PCICONF1 is set to 1.

Table E-35 Format of the Memory Error Message (14/21)

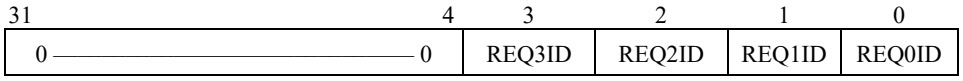
Item	Description
PCIAHER	

Bit	Name	Set value	Explanation
31 to 14	Reserved	–	–
13	MST_BRKN	1: Error 0: Normal	This indicates that the initiator taking ownership of the bus did not perform a FRAME_N assertion within the specified time limit.
12	TGT_BUSTO	1: Error 0: Normal	This indicates that a target did not perform a TRDY_N or STOP_N assertion within the time limit specified in TTOTH at the first data transfer.
11	MST_BUSTO	1: Error 0: Normal	This indicates that the initiator did not receive an IRDY_N assertion within the time limit specified in ITOTH at data transfer.
10 to 4	Reserved	–	–
3	TGT_ABORT	1: Error 0: Normal	This indicates that a target abort occurred when a device other than the local NBU was the initiator.
2	MSR_ABORT	1: Error 0: Normal	This indicates that a master abort occurred when a device other than the local NBU was the initiator.
1	DPERR_WT	1: Error 0: Normal	This indicates that a PERR_N was asserted during a data write when a device other than the local NBU was the initiator.
0	DPERR_RD	1: Error 0: Normal	This indicates that a PERR_N was asserted during a data read when a device other than the local NBU was the initiator.

Table E-35 Format of the Memory Error Message (15/21)

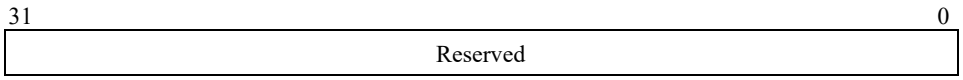
Item	Description

PCIBMLR



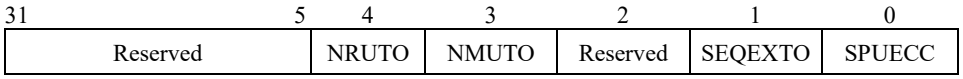
Bit no.	Bit name	Bit value	Explanation
31 to 4	–	Fixed to 0	–
3	REQ3ID	1: Error 0: Normal	This indicates that an error occurred when device 3 (REQ2) was the initiator.
2	REQ2ID	1: Error 0: Normal	This indicates that an error occurred when device 2 (REQ1) was the initiator.
1	REQ1ID	1: Error 0: Normal	This indicates that an error occurred when device 1 (REQ0) was the initiator.
0	REQ0ID	1: Error 0: Normal	This indicates that an error occurred when device 0 (local NBU) was the initiator.

DCMSTS



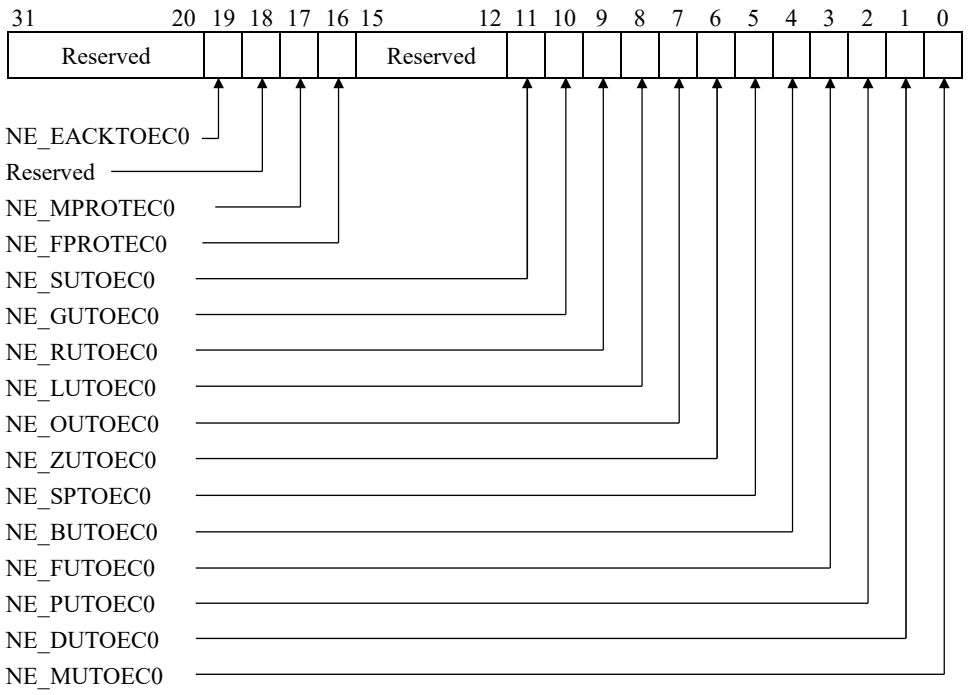
Bit	Name	Set value	Explanation
31 to 0	Reserved	–	–

SPERRS



Bit	Name	Set value	Explanation
31 to 5	Reserved	–	–
4	NRUTO	1: Error 0: Normal	This indicates that the following error occurred: An NRU timeout occurred when the SPU read the system bus as an operand fetch.
3	NMUTO	1: Error 0: Normal	This indicates that the NMU timeout occurred in the following cases: (1) The SPU read the DDR2 memory as an instruction fetch or operand fetch. (2) The SPU read the DDR2 memory as a result of the cache access from the ladder timer or another unit.
2	Reserved	–	–
1	SEQEXTO	1: Error 0: Normal	This indicates that a ladder execution timeout error occurred.
0	SPUECC	1: Error 0: Normal	This indicates that an ECC multiple-bit error occurred when the cache memory or context memory of the SPU was read.

Table E-35 Format of the Memory Error Message (16/21)

Item	Description
NEERRST	

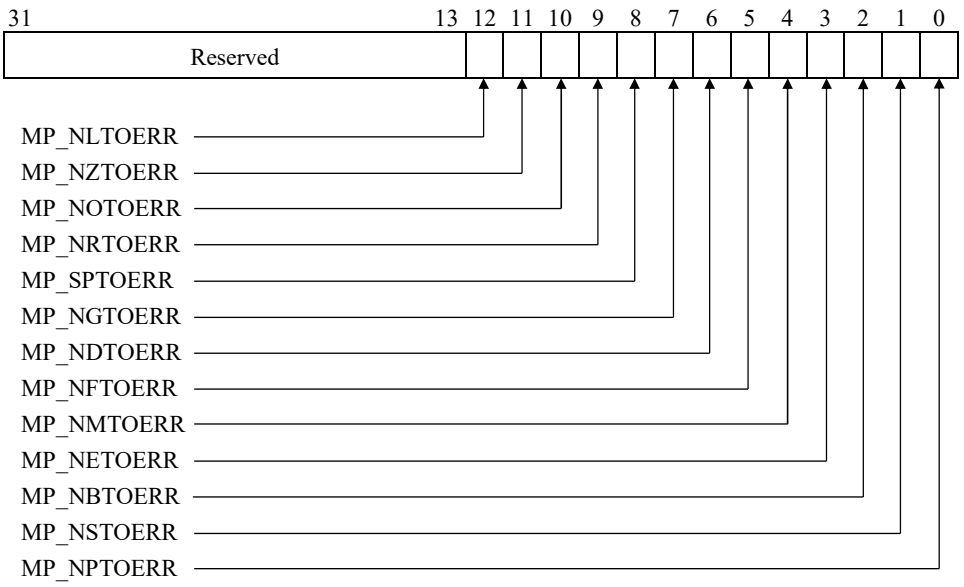
Bit	Name	Set value	Explanation
31 to 20	Reserved	–	–
19	NE_EACKTOEC0	1: Error 0: Normal	This indicates that the following error occurred: although a bus request (EREQ_N) was output to the SP, no bus response (EACK_N) was returned within the period specified in the NE_EACKTOVAL register. (A basic bus TO error occurred.)
18	Reserved	–	–
17	NE_MPROTEC0	1: Error 0: Normal	This indicates that a protection error occurred in the MRAM.

Table E-35 Format of the Memory Error Message (17/21)

Item	Description
NEERRST (continued from the preceding page)	

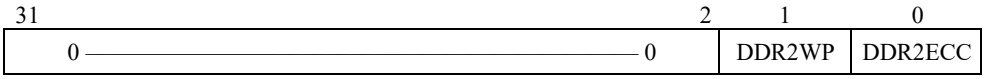
Bit	Name	Set value	Explanation
16	NE_FPROTPEC0	1: Error 0: Normal	This indicates that a protection error occurred due to FROM.
15 to 12	Reserved	–	–
11	NE_SUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NSU access TO* error.
10	NE_GUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NGU access TO* error.
9	NE_RUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NRU access TO* error.
8	NE_LUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NLU access TO* error.
7	NE_OUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NOU access TO* error.
6	NE_ZUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NZU access TO* error.
5	NE_SPTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *SPU access TO* error.
4	NE_BUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NBU access TO* error.
3	NE_FUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NFU access TO* error.
2	NE_PUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NPU access TO* error.
1	NE_DUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NDU access TO* error.
0	NE_MUTOEC0	1: Error 0: Normal	This indicates that an SP access caused an *NMU access TO* error.

Table E-35 Format of the Memory Error Message (18/21)

Item	Description
NPERRLOG	

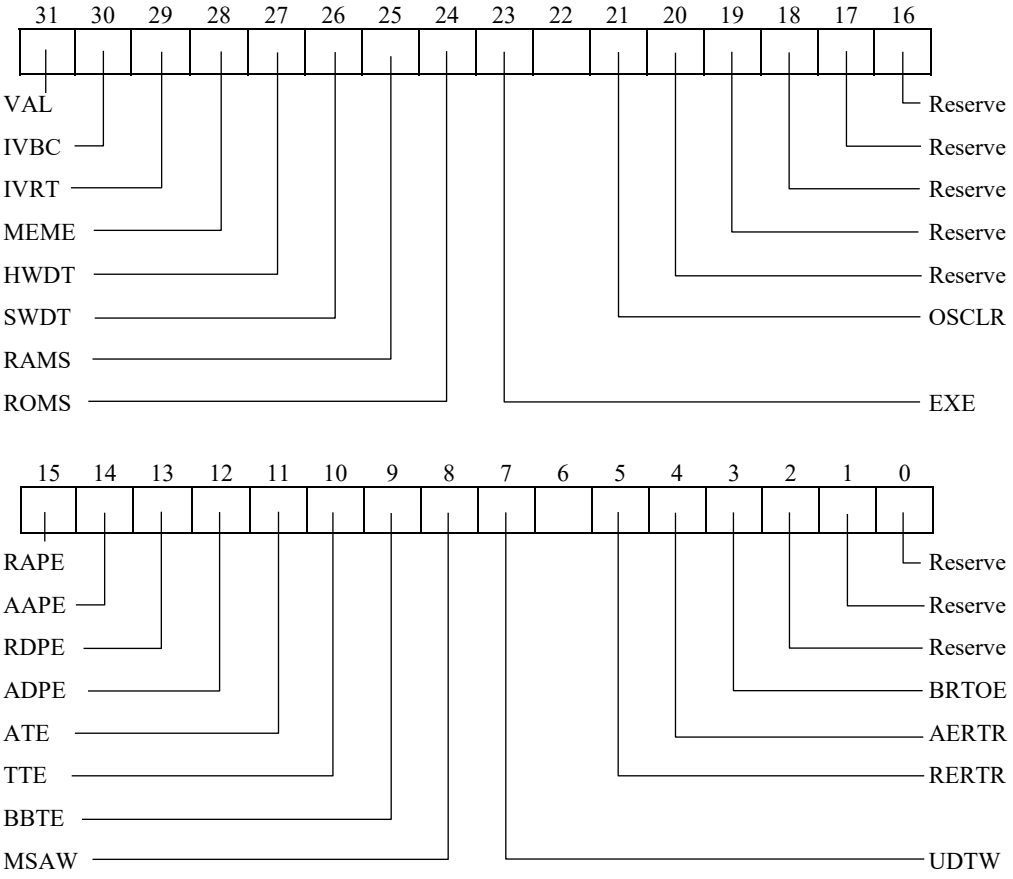
Bit	Name	Set value	Explanation
31 to 13	Reserved	–	–
12	MP_NLTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NLU access TO* error.
11	MP_NZTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NZU access TO* error.
10	MP_NOTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NOU access TO* error.
9	MP_NRTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NRU access TO* error.
8	MP_SPTOERR	1: Error 0: Normal	This indicates that an MP access caused an *SPU access TO* error.
7	MP_NGTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NGU access TO* error.
6	MP_NDTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NDU access TO* error.
5	MP_NFTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NFU access TO* error.
4	MP_NMTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NMU access TO* error.
3	MP_NETOERR	1: Error 0: Normal	This indicates that an MP access caused an *NEU access TO* error.
2	MP_NBTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NBU access TO* error.
1	MP_NSTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NSU access TO* error.
0	MP_NPTOERR	1: Error 0: Normal	This indicates that an MP access caused an *NPU access TO* error.

Table E-35 Format of the Memory Error Message (19/21)

Item	Description
NMHERST	

Bit no.	Bit name	Read-in value	Explanation
31 to 2	–	Fixed to 0	–
1	DDR2WP	0: No error 1: Write protection error	This indicates that a serious fault interrupt occurred due to a write protection error.
0	DDR2ECC	0: No error 1: ECC3 error	This indicates that a serious fault interrupt occurred due to a detected ECC3 error.

Table E-35 Format of the Memory Error Message (20/21)

Item	Description
MSW2	

Bit no.	Bit name	Read-in value	Explanation
31	VAL	0: Invalid 1: Valid	The contents of this register are valid.
30	IVBC	0: Normal 1: Battery error	Battery voltage drop, battery capacity decrease, or battery not mounted
29	IVRT	0: Succeeded 1: Failed	Whether data retention at power failure succeeded or failed
28	MEME	0: Normal 1: Error	An unrecoverable error occurred when the internal memory was being accessed.
27	HWDT	0: Normal 1: Error	A hardware watchdog timer timeout error occurred. (The CPU is included in the EXE.)
26	SWDT	0: Normal 1: Error	A software watchdog timer timeout error occurred.
25	RAMS	0: Normal 1: Error	A RAM checksum error occurred.
24	ROMS	0: Normal 1: Error	A ROM checksum error occurred.

Table E-35 Format of Memory Error Message (21/21)

Item	Description
MSW2 (continued from the preceding page)	

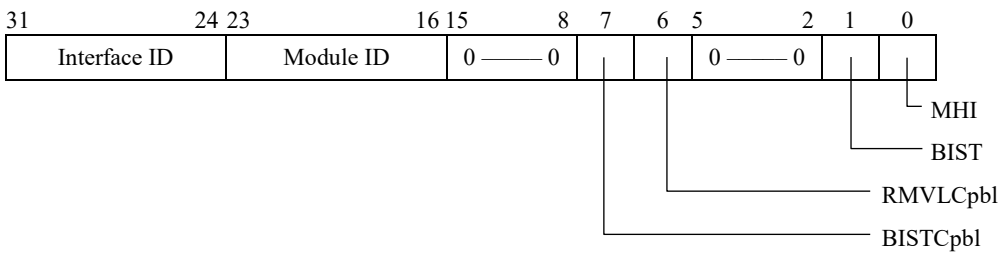
Bit no.	Bit name	Read-in value	Explanation
23	EXE	0: Normal 1: Error	A hardware error occurred.
22	Reserve	Fixed to 0	–
21	OSCLR	0: Normal 1: OS cleared	The OS was cleared.
20 to 16	Reserve	Fixed to 0	–
15	RAPE	0: No error 1: Error	A received address parity error occurred. (This occurs when the device is the master.)
14	AAPE	0: No error 1: Error	An asserted address parity error occurred. (This occurs when the device is a target.)
13	RDPE	0: No error 1: Error	A received data parity error occurred. (This occurs in during a master write or target read.)
12	ADPE	0: No error 1: Error	An asserted data parity error occurred. (This occurs during a master read or target write.)
11	ATE	0: No error 1: Error	An address cycle timeout error occurred. (Only the master is enabled.)
10	TTE	0: No error 1: Error	A transaction timeout error occurred. (Only the master is enabled.)
9	BBTE	0: No error 1: Error	A BGACK busy timeout error occurred. (Only the CPU is enabled.)
8	MSAW	0: No error 1: Error	An invalid misalignment access error occurred.
7	UDTW	0: No error 1: Error	An unsupported transaction error occurred.
6	Reserve	Fixed to 0	–
5	RERTR	0: No error 1: Error	A received error transaction occurred. (Only the master is enabled.)
4	AERTR	0: No error 1: Error	An asserted error transaction occurred.
3	BRTOE	0: No error 1: Error	A bus request timeout error occurred. (Only the master is enabled.)
2 to 0	Reserve	Fixed to 0	–

E.3.13 System bus error

This error indicates that a serious fault occurred in the system bus.
Table E-37 shows and describes the error message.

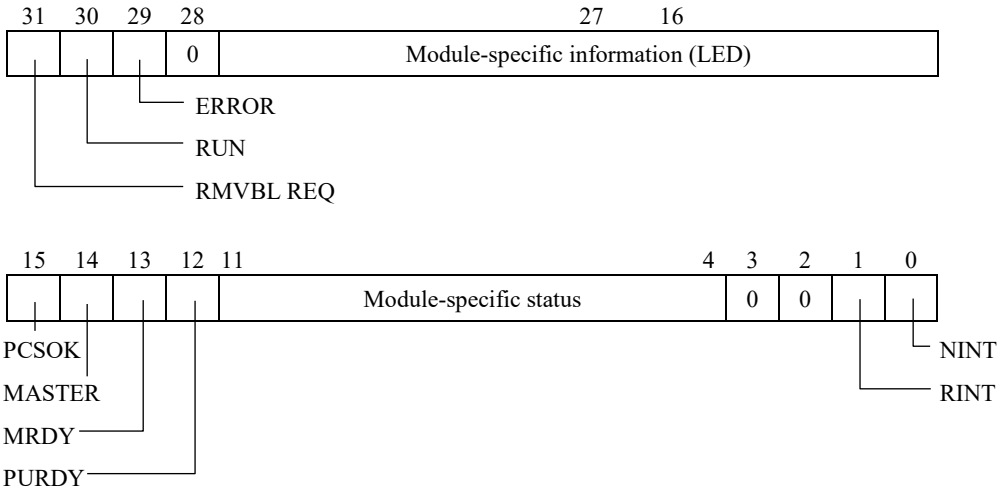
Table E-36 Format of the System Bus Error Message (1/6)

```
CPU xxxxxxxx
%CPMS-E-HARD-0013  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx System Bus Error (subtitle)
SLOT    =xxxxxxxx  MSW0    =xxxxxxxx  MSW1    =xxxxxxxx
HERST   =xxxxxxxx  STAT    =xxxxxxxx  EN      =xxxxxxxx
ADR     =xxxxxxxx  DAT     =xxxxxxxx  MST     =xxxxxxxx  CMD     =xxxxxxxx
LOGMST  =xxxxxxxx  LOGSLV  =xxxxxxxx  RBUSMNT  =xxxxxxxx  HERST   =xxxxxxxx
```

Item	Description
EC	Error code (See Table E-37.)
SLOT	Master slot number
MSW0	

Bit no.	Bit name	Read-in value	Explanation
31 to 24	Interface ID	Interface ID	Module ID code for software
23 to 16	Module ID	Module ID	Module ID code for hardware
15 to 8	–	Fixed to 0	–
7	BISTCpbl	0: Not equipped 1: Equipped	Whether the module is equipped with a self-diagnostic function (optional)
6	RMVLCpbl	0: Cannot be hot swapped 1: Can be hot swapped	Whether the module can be hot swapped (in the case of the PU, for future use)
5 to 2	–	Fixed to 0	–
1	BIST	0: Self-diagnosis ended 1: Self-diagnosis running	Operation state of the self-diagnostic function of a module (optional)
0	MHI	0: Initialization ended 1: Initialization running	Initialization state of the module hardware

Table E-36 Format of the System Bus Error Message (2/6)

Item	Description
MSW1	

Bit no.	Bit name	Read-in value	Explanation
31	RMVBL REQ	0: Not requested 1: Requested	Whether a hot swap was requested
30	RUN	0: RUN not set 1: RUN set	Module state (RUN LED turns on or flashes)
29	ERROR	0: No error 1: Error	Error status (ERR LED turns on or flashes)
28	–	Fixed to 0	–
27 to 16	Module-specific information (LED)	Module-specific information (LED)	Hardware module-specific LED state
15	PCSOK	0: Not output 1: Output	PCSOK status (output)
14	MASTER	0: Not output 1: Output	MASTER status (output)
13	MRDY	0: MRDY on 1: MRDY off	MASTER READY status (This is enabled only for the CPU.)
12	PURDY	0: PURDY on 1: PURDY off	PU READY status (This is enabled only for the PU.)
11 to 4	Module-specific status	Module-specific status	Module-specific status
3, 2	–	Fixed to 0	–
1	RINT	0: No factor detected 1: Factor detected	Whether a general interrupt other than a network-related interrupt occurred
0	NINT	0: No factor detected 1: Factor detected	Whether a network interrupt occurred

Table E-36 Format of the System Bus Error Message (3/6)

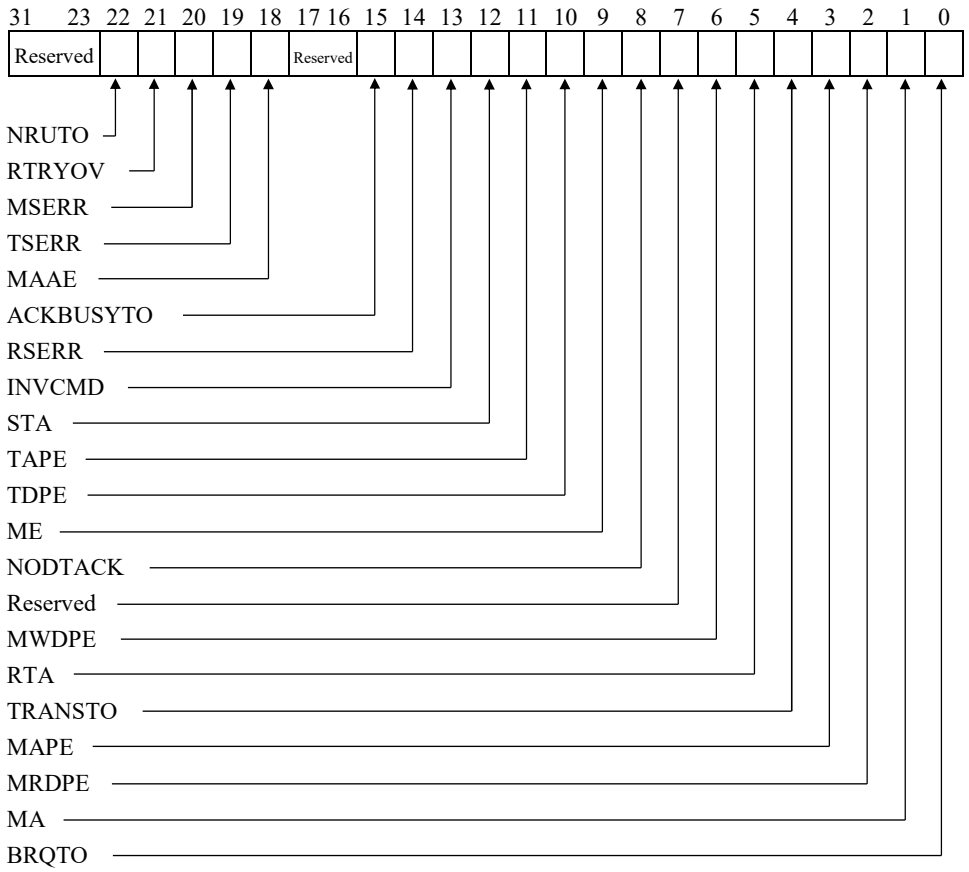
Item	Description
HERST	See Table E-35 for HERST.
STAT	See Table E-35 for STAT.
EN	

Table E-36 Format of the System Bus Error Message (4/6)

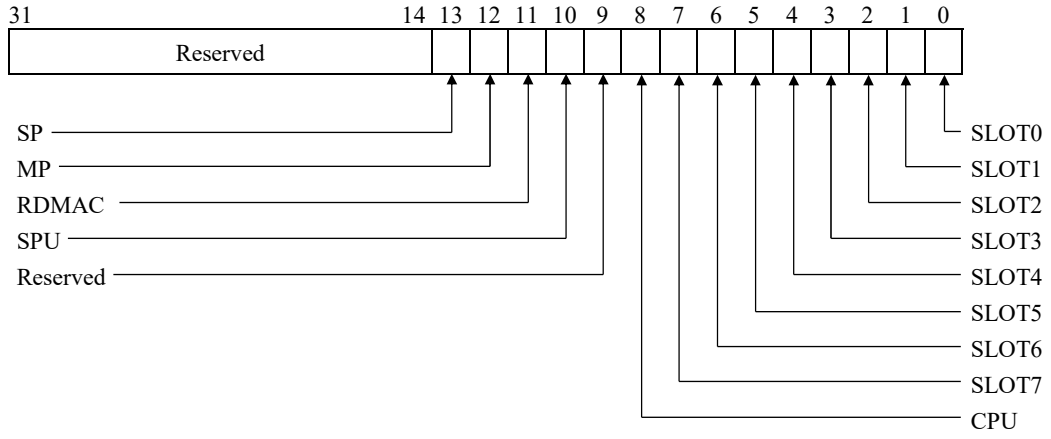
Item	Description
EN (continued from the preceding page)	

Bit	Name	Set value	Explanation
31 to 23	Reserved	–	–
22	NRUTO	1: Interrupt enabled 0: Interrupt disabled	Internal TO error
21	RTRYOV	1: Interrupt enabled 0: Interrupt disabled	The retry limit was exceeded. When the device was acting as bus master, the retry frequency limit specified in the NR_RRTRYMAX register was exceeded.
20	MSERR	1: Interrupt enabled 0: Interrupt disabled	The bus master received an SERROR. The device acting as bus master received an SERROR signal.
19	TSERR	1: Interrupt enabled 0: Interrupt disabled	The R700 acting as a target received an SERROR. The R700 acting as a target received an SERROR signal.
18	MAAE	1: Interrupt enabled 0: Interrupt disabled	A misalignment access error was detected. The R700 acting as a target detected an invalid byte-enable (BE) during a write operation.
17 to 16	Reserved	–	–
15	ACKBUSYTO	1: Interrupt enabled 0: Interrupt disabled	A BGACK busy TO was detected. This indicates that the arbiter detected a BGACK assertion timeout. (The assertion did not end.)
14	RSERR	1: Interrupt enabled 0: Interrupt disabled	An SERROR was received.
13	INVCMD	1: Interrupt enabled 0: Interrupt disabled	An invalid command was detected. This indicates that the R700 acting as a target terminated the bus operation due to an error transaction.
12	STA	1: Interrupt enabled 0: Interrupt disabled	An error transaction was transmitted. This indicates that the R700 acting as a bus target transmitted an error transaction as a response.
11	TAPE	1: Interrupt enabled 0: Interrupt disabled	An address parity error was detected. This indicates that the R700 acting as a target detected an address parity error (APE).
10	TDPE	1: Interrupt enabled 0: Interrupt disabled	A write parity error was detected. This indicates that the R700 acting as a target detected a write data parity error (WDPE).
9	ME	1: Interrupt enabled 0: Interrupt disabled	A memory error was detected. This indicates that the target performing read transaction through the R700/S10 bus detected a 4bitECC error in the memory.

Table E-36 Format of the System Bus Error Message (5/6)

<table>
<tr><th>Item</th><th>Description</th></tr>
<tr><td>EN
(continued from the preceding page)</td><td>
<table>
<tr><th>Bit</th><th>Name</th><th>Set value</th><th>Explanation</th></tr>
<tr><td>8</td><td>NODTACK</td><td>1: Interrupt enabled
0: Interrupt disabled</td><td>A data ready signal was not detected.
This indicates that a timeout was detected during monitoring of the period between when the S10 bus started and when the data became ready for communication (DTACK assertion). This resulted in the CPU asserting DTACK instead.</td></tr>
<tr><td>7</td><td>Reserved</td><td>–</td><td>–</td></tr>
<tr><td>6</td><td>MWDPE</td><td>1: Interrupt enabled
0: Interrupt disabled</td><td>The bus master received a WDPE.
This indicates that when the device was acting as bus master, a target detected a write data parity error (WDPE) and the master received an error notification from the target.</td></tr>
<tr><td>5</td><td>RTA</td><td>1: Interrupt enabled
0: Interrupt disabled</td><td>An error transaction was received.
This indicates that the device acting as bus master terminated its bus operation due to an error transaction.</td></tr>
<tr><td>4</td><td>TRANSTO</td><td>1: Interrupt enabled
0: Interrupt disabled</td><td>A transaction TO was detected.
This indicates that the device acting as bus master detected a transaction timeout (TO) and terminated its bus operation.</td></tr>
<tr><td>3</td><td>MAPE</td><td>1: Interrupt enabled
0: Interrupt disabled</td><td>The bus master received an APE.
This indicates that the device acting as bus master received an address parity error (APE) notification.</td></tr>
<tr><td>2</td><td>MRDPE</td><td>1: Interrupt enabled
0: Interrupt disabled</td><td>The bus master received an RDPE.
This indicates that the device acting as bus master detected a read-data parity error (RDPE).</td></tr>
<tr><td>1</td><td>MA</td><td>1: Interrupt enabled
0: Interrupt disabled</td><td>An address cycle TO was detected.
The device acting as bus master detected an address cycle timeout (TO) and terminated its bus operation.</td></tr>
<tr><td>0</td><td>BRQTO</td><td>1: Interrupt enabled
0: Interrupt disabled</td><td>A bus request TO was detected.
This indicates that the device acting as bus master detected a bus request timeout (TO) and terminated its bus operation.</td></tr>
</table>
</td></tr>
<tr><td>ADR</td><td>See Table E-35 for RERRADR.</td></tr>
<tr><td>DAT</td><td>See Table E-35 for RERRDAT.</td></tr>
<tr><td>MST</td><td>See Table E-35 for RERRMST.</td></tr>
<tr><td>CMD</td><td>See Table E-35 for RERRCMD.</td></tr>
<tr><td>LOGMST</td><td>See Table E-35 for LOGMST.</td></tr>
<tr><td>LOGSLV</td><td>See Table E-35 for LOGSLV.</td></tr>
</table>

Table E-36 Format of the System Bus Error Message (6/6)

Item	Description
RBUSMNT	

Bit	Name	Set value	Explanation
31 to 14	Reserved	–	–
13	SP	1: SP is the bus master. 0: SP is not the bus master.	SP master bus transaction This indicates that the SP is currently the bus master and is performing a bus transaction.
12	MP	1: MP is the bus master. 0: MP is not the bus master	MP master bus transaction This indicates that the MP is currently the bus master and is performing a bus transaction.
11	RDMAC	1: RDMAC is the bus master. 0: RDMAC is not the bus master.	RDMAC master bus transaction This indicates that the RDMAC is currently the bus master and is performing a bus transaction.
10	SPU	1: SPU is the bus master. 0: SPU is not the bus master.	SPU master bus transaction This indicates that the SPU is currently the bus master and is performing a bus transaction.
9	Reserved	–	–
8	CPU	1: CPU is the bus master 0: CPU is not the bus master.	CPU master bus transaction This indicates that the CPU is currently the bus master and is performing a bus transaction.
7 to 0	SLOT (7 to 0)	1: SLOT (7 to 0) is the bus master. 0: SLOT (7 to 0) is not the bus master.	SLOT (7 to 0) master bus transaction This indicates that the SLOT (7 to 0) is currently the bus master and is performing a bus transaction.

Table E-37 Error Codes, Subtitles, and their Descriptions

No.	Error code	Subtitle	Description
1	EC=03b70000	Master/Target Abort	The CPU acting as master detected a master abort or target abort in the system bus.
2	EC=03b70001	S10 Bus DTACK Timeout	The CPU acting as master detected a timeout during S10 bus access.
3	EC=03b80001	CPU Master	The CPU acting as master went down due to a serious fault occurring in the system bus.
4	EC=03b80002	CPU Target	The CPU acting as a target detected a serious fault in the system bus.

E.3.14 Ladder program error

This error indicates the detection of an abnormality preventing the ladder program from continuing processing.
Table E-39 shows and describes the error message.

Table E-38 Format of the Ladder Program Error Message (1/2)

```
CPU xxxxxxxx
%CPMS-E-SOFT-0010  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx Ladder Program Error (subtitle)
TN  =xxxxxxxx  CNO  =xxxxxxxx  PC   =xxxxxxxx  SP   =xxxxxxxx  FADR =xxxxxxxx
R0  =xxxxxxxx  R1   =xxxxxxxx  R2   =xxxxxxxx  R3   =xxxxxxxx  R4   =xxxxxxxx
R5  =xxxxxxxx  R6   =xxxxxxxx  R7   =xxxxxxxx  R8   =xxxxxxxx  R9   =xxxxxxxx
R10 =xxxxxxxx  R11  =xxxxxxxx  R12  =xxxxxxxx  R13  =xxxxxxxx  R14  =xxxxxxxx
R15 =xxxxxxxx  FPSCR=xxxxxxxx  FPUL =xxxxxxxx
FR0 =xx.xxxxxxExxx FR1  =xx.xxxxxxExxx FR2  =xx.xxxxxxExxx FR3  =xx.xxxxxxExxx
FR4 =xx.xxxxxxExxx FR5  =xx.xxxxxxExxx FR6  =xx.xxxxxxExxx FR7  =xx.xxxxxxExxx
FR8 =xx.xxxxxxExxx FR9  =xx.xxxxxxExxx FR10 =xx.xxxxxxExxx FR11 =xx.xxxxxxExxx
FR12 =xx.xxxxxxExxx FR13 =xx.xxxxxxExxx FR14 =xx.xxxxxxExxx FR15 =xx.xxxxxxExxx
DR0 =xx.xxxxxxExxx DR2  =xx.xxxxxxExxx DR4  =xx.xxxxxxExxx DR6  =xx.xxxxxxExxx
DR8 =xx.xxxxxxExxx DR10 =xx.xxxxxxExxx DR12 =xx.xxxxxxExxx DR14 =xx.xxxxxxExxx
DSEG0=xxxxxxxx  DSEG1=xxxxxxxx  DSEG2=xxxxxxxx  DSEG3=xxxxxxxx  DSEG4=xxxxxxxx
DSEG5=xxxxxxxx  DSEG6=xxxxxxxx  DSEG7=xxxxxxxx  ISEG =xxxxxxxx  SSEG =xxxxxxxx
DWE0S=xxxxxxxx  DWE0E=xxxxxxxx  DWE1S=xxxxxxxx  DWE1E=xxxxxxxx
DWE2S=xxxxxxxx  DWE2E=xxxxxxxx  DWE3S=xxxxxxxx  DWE3E=xxxxxxxx
STS  =xxxxxxxx  EXDIA=xxxxxxxx  SEQMD=xxxxxxxx  INTST=xxxxxxxx  ERRST=xxxxxxxx
ERTRI=xxxxxxxx  CERRS=xxxxxxxx  MERS =xxxxxxxx  DPEA =xxxxxxxx  BWPEA=xxxxxxxx
INST =xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
      xxxxxxxx   (PC =) xxxxxxxx xxxxxxxx
Processor type = xx
```

Item	Description
EC	Error code (See Table E-39.)
TN	Task number of the task at which an error occurred
CNO	Context number
PC	Contents of the program counter
EXPEV	Contents of the exception code register. The exception code register is a 32-bit register defining the causes of data access exceptions and alignment exceptions.
FADR	Fault address
SR	Status register. This register defines the state of the processor.
EXECD	Instruction that was executed when an abnormality occurred (error factor code)
PR	Contents of the procedure register. The procedure register is used for subroutine calls. If the executed program was the last program in the subroutine calling sequence, this register contains the return address.
SP	Contents of the stack pointer (R15 is used as a stack pointer.)

Table E-38 Format of the Ladder Program Error Message (2/2)

Item	Description
GBR	Contents of the global base register. This register contains the base addresses for GBR-indirect addressing with displacement and indexed GBR-indirect addressing.
MACH	MAC register. This register is used for storing additional values of MAC (multiply and accumulate operation) instructions, MAC instructions, and results of MUL instructions. If the calculated result is a 64-bit value, this register stores the upper 32 bits.
MACL	MAC register. If the calculated result is a 64-bit value, this register stores the lower 32 bits. If the calculated result is a 32-bit value, the register stores the 32 bits.
R*xx*	Contents of the general register represented by the value at *xx*.
FPSCR	Contents of the floating-point status and control register
FPUL	Contents of the floating-point communication register. Data is transferred between a general register and floating-point register through this register.
FR*xx*	Contents of the 32-bit floating-point register represented by the value at *xx*. When FPSCR.FR (bit 21 of a 31-0 bit value) is 0, this is the value of FPR*xx*_BANK0. When FPSCR.FR is 1, this is the value of FPR*xx*_BANK1.
DR*xx*	Contents of the 62-bit floating-point register represented by the value at *xx*. When FPSCR.FR (bit 21 of a 31-0-bit value) is 0, this is the value of FPR*xx*_BANK0. When FPSCR.FR is 1, this is the value of FPR*xx*_BANK1.
DSEG*x*	Data segment register value
ISEG	Instruction segment register value
SSEG	Stack segment register value
DWE*x*S	SP_DWExS register value
STS	SP_STS register value
EXDIA	SP_EXDIA register value
SEQMD	SP_MODE register value
INTST	SP_INTSTS register value
ERRST	SP_ERRSTS register value
ERTRI	SP_ERTRI register value
CERRS	SP_CxERRS register value
MERS	SP_MERS register value
DPEA	SP_DPEA register value
BWPEA	SP_BWPEA register value
INST	Eight instructions preceding and following the PC address
PC	Instruction of the PC
STACK	Eight long-word data items preceding and following the SP address
SP	Contents of the SP address (contents of the stack)
PC	Information about the address contained in the program counter is displayed in parentheses. When the address is for a program, the following information is displayed: `name` = *program name*, `type` = *program type (program location)*, `raddr` = *relative address from the program*

Table E-39 Error Codes, Subtitles, and their Descriptions (Ladder Program Errors)

No.	Error code	Subtitle	Description	Explanation
1	EC=03D00001	Data Access Protection	Data access protection error	A data access to a protected area was detected.
2	EC=03D00002	Stack Overflow	Stack overflow error	A stack pointer overflow was detected.
3	EC=03D00003	Illegal Instruction	Illegal instruction error	An illegal instruction error occurred.
4	EC=03D00004	FP Program Error	Floating-point calculation error	A floating-point calculation error occurred.
5	EC=03D00005	Segment Address Overflow	Segment address overflow error	A segment address overflow was detected.
6	EC=03D00006	Illegal SH Instruction	Illegal SH instruction detected (SH instruction processing error)	An SH processing instruction that could not be processed was detected.
7	EC=03D01101	P-Coil CP DOWN Detect	P-coil CP detected to be down	When the P-coil was operated, it was detected that the CP was down.
8	EC=03D01208	N-Coil Nesting Over	N-coil overflow (SH instruction processing error)	The maximum nesting level was exceeded in the N-coil.
9	EC=03D0120A	Illegal User Function	Address registration error of user-defined function (SH instruction processing error)	The registered address of the user-defined calculation function was abnormal (outside the area that could be registered, odd-number address).
10	EC=03D0120C	Illegal Function Parameter	System calculation function parameter error	A parameter error was detected in the system calculation function.
11	EC=03D01210	Ladder Area Sum Mismatch	Sum value check mismatch	The following did not match: The calculated value and attached checksum value calculated from, for example, the table or program data in the LADDER area.
12	EC=03D01212	Ladder Table Empty	Unregistered table	The table shared by the ladder execution control task and kernel could not be obtained.
13	EC=03D01214	Illegal Factor	Initiation factor error	The ladder program was started by an illegal factor from the scheduler task.

E.3.15 Other error

This error indicates an error that is output by a user.
The error code and error details are defined by the user who performs output.

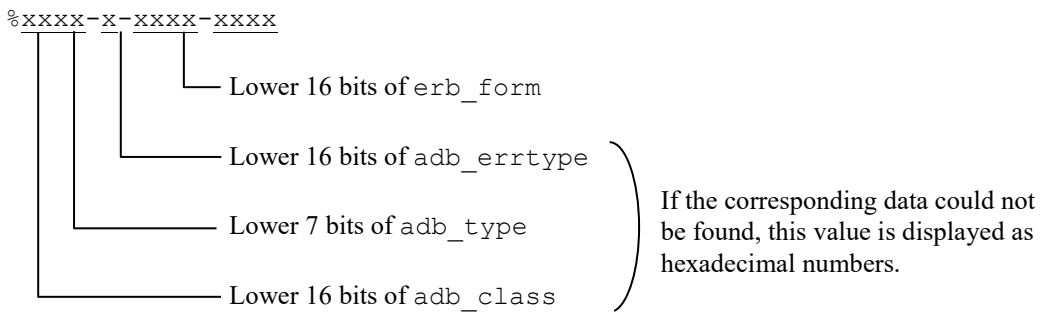
Table E-40 Format of the Other Error Message

```
CPU xxxxxxxx
%xxxx-x-xxxx-xxxx  SITE=xxxxxxxxxxxxxx RC=xxxxxxxx yyyy/mm/dd hh:mm:ss LOG=xxx
EC=xxxxxxxx
0x00000000 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  `xxxxxxxxxxxxxxx`
0x00000010 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  `xxxxxxxxxxxxxxx`
0x00000020 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  `xxxxxxxxxxxxxxx`
0x00000030 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  `xxxxxxxxxxxxxxx`
0x00000040 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  `xxxxxxxxxxxxxxx`
0x00000050 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  `xxxxxxxxxxxxxxx`
0x00000060 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  `xxxxxxxxxxxxxxx`
0x00000070 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx  `xxxxxxxxxxxxxxx`
(Other data might be displayed depending on the error.)
```

RC: Return code

EC: Error code

Title: The title is output in the following format. If the CPMS could not recognize the data, the data value is displayed as hexadecimal numbers.



E.4 Reading Displayed DHP Data

DHP data is displayed in reverse chronological order (from newest to oldest).
DHP data is divided into *task*, *idle*, and *OS* areas by using the DISPATCH_E event as the separator.
The DISPATCH_E event uses task numbers ranging from 00000001 to 000000FF to represent tasks.
The time is displayed in hours, minutes, and seconds to six decimal places (microseconds).
For the correspondence between DHP events and DHP data, see APPENDIX F LIST OF DHP CODES.

Example of displayed DHP data

The Description of Behavior column next to the following DHP data indicates the task behavior and OS behavior when switching the task to be run.

New

			Description of Behavior
15	00.665805 TASK_PRI 112 10	00000071 00000032	Task 112 is running.
16	00.665799 RLEAS 112 10	00000071	
17	00.665791 DISPATCH_E 112 10	00000070 00000032 85877000 00000002	The OS aborts task 111 and then switches to run task 112.
18	00.665786 DISPATCH 111 10	0000006F 00000032 85871000	
19	00.665771 DISPATCH_E 111 10	0000006F 00000032 85871000 00000002	
20	00.665764 RUNQ 112 10	00000070	
21	00.665761 DISPATCH 112 10	00000070 00000032 85877000	
22	00.665756 RUNQ 112 10	0000006F	
23	00.665752 WAKEUP 112 10	F0000000	
24	00.665745 ABORT 112 10	0000006F	Task 112 is running.
25	00.665739 GFACT 112 10	00000003	
26	00.665729 DISPATCH_E 112 10	00000070 00000032 85877000 00000002	The OS delays task 111 and then switches to run task 112.
27	00.665721 DISPATCH 111 10	0000006F 0000001C 85871000	
28	00.665716 TASK_PRI 111 10	0000006F 0000001C 00000000	
29	00.665698 DELAY 111 10	00000BB8	Task 111 is running.
30	00.665693 RUNQ 111 10	00000070	
31	00.665689 QUEUE 111 10	00000070 00000003	
32	00.665683 TASK_PRI 111 10	00000070 00000032	
33	00.665677 RLEAS 111 10	00000070	
34	00.665672 GFACT 111 10	00000002	
35	00.665664 DISPATCH_E 111 10	0000006F 00000032 85871000 00000002	The OS keeps task 110 waiting and then switches to run task 111.
36	00.665657 DISPATCH 110 10	0000006E 00000032 8586B000	
37	00.665653 SLEEP 110 10	841E27CC 00000032	
38	00.665647 WAIT 110 10	5004502C	Task 110 is running.
39	00.665643 RUNQ 110 10	0000006F	
40	00.665639 QUEUE 110 10	0000006F 00000002	
41	00.665632 TASK_PRI 110 10	0000006F 00000032	
42	00.665626 RLEAS 110 10	0000006F	
43	00.665618 DISPATCH_E 110 10	0000006E 00000032 8586B000 00000001	The OS stops task 119 and switches to run task 110.
44	00.665613 RUNQ 110 10	0000006E	
45	00.665610 DISPATCH 110 10	0000006E 00000032 8586B000	
46	00.665601 DISPATCH_E 110 10	0000006E 00000032 8586B000 00000002	
47	00.665593 DISPATCH 119 10	00000077 00000032 85883000	
48	00.665583 EXIT 119 10		Task 119 is running.
49	00.665576 RUNQ 119 10	0000006E	
50	00.665571 WAKEUP 119 10	841E27CC	
51	00.665563 POST 119 10	5004502C 00001234	

Old

Table F-1 List of DHP Codes (1/4)

Code value	DHP display name	Trace point	para1	para2	para3	para4	para5
0x010000		CPMS processing (trace processing)					
0x010001	TRACE_ON	Trace start					
0x010002	TRACE_OFF	Trace stop					
0x010003	TRACE_TBU	Time record	old tbu (time base upper)	new tbu (time base upper)			
0x020000		CPMS processing (scheduling processing)					
0x020001	WAKEUP	WAKEUP processing	wchan				
0x020002	SLEEP	SLEEP event	wchan	pri (priority level)			
0x020003	DISPATCH	Before task switch	tn (task number)	pri (priority level)	cont (CPMS stack information)		
0x020083	DISPATCH_E	After task switch	tn (task number)	pri (priority level)	cont (CPMS stack information)	Record point identifier	
0x020004	RUNQ	RUNQ connection	tn (task number)				
0x020005	IDLE	IDLE processing					
0x020006	TASK_PRI	Priority level control	tn (task number)	pri (priority level)			
0x030000		CPMS processing (error log, built-in subroutine processing)					
0x030001	ULSUBLN	Before built-in subroutine link	nest	point	entry		
0x030081	ULSUBLN_E	After built-in subroutine link	nest	point	entry	ret (output information)	
0x030002	ELSET	elset processing	type	class	form	erd (error code)	
0x030003	IOERR	I/O error processing	uno	dev	dva	ioec	
0x030004	PRGERR	Program error processing	tn (task number)	fadr (fault address)	PC (program counter)	EXPEVT (exception code)	
0x030005	WDTERR	WDT error processing	time				
0x030007	MODERR	Module error processing	errcode	slot	HERST (serious fault factor)	INTST (interrupt code)	
0x030008	KERN_PANIC	Panic processing	tn (task number)	fadr (fault address)	PC (program counter)	excode	
0x030009	ULSUB_ERR	Built-in subroutine error processing	point	fadr (fault address)	PC (program counter)	nest	
0x03000A	ASSERT	Assertion panic processing	file	line	str		
0x03000B	CPUSTOP	CPU stop processing	nest	point	ret		
0x040000		CPMS processing (startup/stop processing)					
0x040001	SETUP_MAIN	Startup processing	id (initiation factor)				
0x040002	HDUTL_STOP	Stop processing					
0x040003	HDUTL_RSUM	Restart processing					
0x040004	HDUTL_ERR	ERROR processing					
0x050000		CPMS processing (exception processing)					
0x050001	EXCEPTION	Exception processing	errcode	EXPEVT (exception code)			
0x050002	SLIH_SRES	System reset exception	NMIST (NMI factor)	PC (program counter)			
0x050005	SLIH_SM	System management interrupt exception	MSW (machine status word)				
0x050007	SLIH_HERR	Serious fault interrupt processing	HERST (serious fault factor)				
0x060000		Kernel processing (patrol)					
0x060001	RMVBLOFF	When hot swapped at RMVBL_SW	slot_no	MSW0	MSW1		
0x060002	RMVBLON	When hot swapped at RMVBL_SW	slot_no	MSW0	PCSR register value		
0x060003	BATTCHRG	Battery charge HKP utility call	Record point identifier				
0x100000		CPMS processing (macro processing)					
0x100000	NOSYS	Undefined macro issuance					
0x100001	QUEUE	`queue` issuance	tn (task number)	fact (initiation factor)			
0x100002	RLEAS	`rleas` issuance	tn (task number)				
0x100003	SFACT	`sfact` issuance	tn (task number)	fact (initiation factor)			
0x100004	ABORT	`abort` issuance	tn (task number)				

APPENDIX F LIST OF DHP CODES

Table F-1 List of DHP Codes (2/4)

Code value	DHP display name	Trace point	para1	para2	para3	para4	para5
0x100005	SUSP	susp issuance	tn (task number)				
0x100006	RSUM	rsum issuance	tn (task number)				
0x100007	CTIME	ctime issuance	tn (task number)	fact (initiation factor)			
0x100008	WAIT	wait issuance	ecb (ECB address)				
0x100009	POST	post issuance	ecb (ECB address)	pcode (post code)			
0x10000A	TIMER	timer issuance	id (event type)	tn (task number)	fact (initiation factor)	t (duration/time)	cyt (cycle duration)
0x10000B	DELAY	delay issuance	t (milliseconds)				
0x10000C	STIME	stime issuance	year (year)	month (month)	day (day)	msec (milliseconds)	
0x10000D	CHAP	chap issuance	tn (task number)	chgp (priority level)			
0x10000E	RSERV	rserv issuance	n (number of shared resources)	para1	para2	para3	para4
0x10000F	FREE	free issuance	n (number of shared resources)	para1	para2	para3	para4
0x100010	PRSERV	prsrv issuance	n (number of shared resources)	para1	para2	para3	para4
0x100011	PFREE	pfree issuance	n (number of shared resources)	para1	para2	para3	para4
0x100012	GFACT	After gfact issuance	fact (initiation factor)				
0x100013	GTIME	gtime issuance	time (time_t address)				
0x100014	EXIT	exit issuance					
0x100015	ASUSP	asusp issuance					
0x100016	ARSUM	arsum issuance					
0x100017	OPEN	open issuance	uno	flag			
0x100018	CLOSE	close issuance	uno				
0x100019	READ	read issuance	uno	vaddr	cnt		
0x10001A	WRITE	write issuance	uno	vaddr	cnt		
0x10001B	IOCTL	ioctl issuance	uno	req	arg		
0x10001E	DHPCTL	dhpctl issuance	cmd (command)	id (major ID)	info		
0x10001F	DHPREAD	dhpread issuance	vaddr (logical address)	size			
0x100023	CHML	chml issuance	vaddr (logical address)	para1	para2	para3	para4
0x10002E	DCMSTAT	dcmstat issuance	dustsr	duisr			
0x100056	CFREAD	cfread issuance	sector	size	saddr		
0x100057	CFWRITE	cfwrite issuance	sector	size	saddr		
0x10005A	PIO10CTL	pio10ctl issuance	radr	request	arg		
0x10005F	PUDHPCTL	pudhpctl issuance	Control code				
0x100060	XPUREQ	Setting of inter-PU communication data	Source processor number	Destination processor number	Communication data information	Communication data information	
0x100061	XPUSINT	Inter-PU interrupt issuance	Interrupting processor number	Interrupted processor number	Interrupt information		
0x100062	XPURINT	Inter-PU interrupt reception	Interrupting processor number				
0x100063	XPUSYNC	Inter-PU time synchronization	Time (seconds)	Time (nanoseconds)			
0x100064	XPUERR	XPU error	Error code	Processor number of the processor at which an error occurred			
0x100065	XPUNOTFY	XPU error report reception	Error code	Slot number of the slot at which an error occurred	data1		
0x100066	PUWAIT	puwait issuance	ecb (ECB address)				
0x100067	PUPOST	pupost issuance	ecb(ECB address)	pcode (post code)			
0x100068	PURSERV	purserv issuance	n (number of shared resources)	para1	para2	para3	para4
0x100069	PUFREE	purfree issuance	n (number of shared resources)	para1	para2	para3	para4
0x10006A	PUSEND	pusend issuance	channel	s_data	size		
0x10006B	PURECV	purecv issuance	channel	r_data	size		
0x10006C	CPUGTIME	cpugtime issuance	time	t			
0x10007F	ELCTL	elctl issuance	cmd				
0x1000A1	TIMER_QUEUE	queue issuance from timer processing	tn (task number)	fact (initiation factor)	id (timer ID)		
0x1000B7	PIOSIMCTL	piosimct macro issuance		Table address			

Table F-1 List of DHP Codes (3/4)

Code value	DHP display name	Trace point	para1	para2	para3	para4	para5
0x1000BD	DCMCHECK	dcmcheck issuance	info				
0x1000BE	DCMDUISR	DCM interrupt detection	dustsr	duisr			
0x1000CA	LDRSTART	ldrstart issuance	addr				
0x1000CB	APLLEDCTL	aplledctl issuance	index	led			
0x1000CC	LDRRESTART	LDS post processing	Number of LDS entries	LDS1 return value	LDS2 return value	LDS3 return value	LDS4 return value
0x1000CF	PPINFOCTL	ppinfoctl issuance	cmd	arg			
0x1000D1	USRDISPCTL	usrdispctl issuance	cmd	arg	size		
0x200000		CPMS processing (RPDP processing)					
0x200004	SETTCB	settcb issuance	toptn	cnt	tcaddr		
0x200005	CLRTCB	clrtcb issuance	tn (task number)				
0x200006	ADTSET	adtset issuance	ADT mode	Break channel	Break address		
0x200007	ADTREAD	adtread issuance	Register storage area address	ADTB storage area address		Address mask pattern	Break mode
0x200008	SETBRK	setbrk issuance	Mode	Break point address	Instruction code storage address		
0x200009	GETBRK	getbrk issuance	UBCB storage area address				
0x20000A	GOTASK	gotask issuance					
0x20000C	REGSET	regset issuance	reg (register type)	Setting value storage area address			
0x20000D	LDRSETBRK	ldrsetbrk issuance	kind	tn	addr		
0x20000E	LDRGETBRK	ldrgetbrk issuance	param				
0x20000F	LDRSTEP	ldrstep issuance					
0x200010	LDRGO	ldrgo issuance					
0x200011	LDRSETREG	ldrsetreg issuance	reg	data			
0x200012	LDRGETREG	ldrgetreg issuance	param				
0x300000		RCTLNET processing					
0x300001	SOCKET	SOCKET issuance	uno (unit number)	Type	Protocol	Work	Work
0x300002	BIND	BIND issuance	Socket ID	Port number	IP address	Work	Work
0x300003	LISTEN	LISTEN issuance	Socket ID	Maximum number of sockets waiting for connection	Work	Work	Work
0x300004	ACCEPT	ACCEPT issuance	Socket ID	Address information pointer	Address information length	Work	Work
0x300005	CONNECT	CONNECT issuance	Socket ID	Port number	IP address	Work	Work
0x300006	SEND	SEND issuance	Socket ID	Buffer address	Data length + transmission flag	Work	Work
0x300007	SENDTO	SENDTO issuance	Socket ID	Data length + transmission flag	Port number	IP address	Task information
0x300008	RECV	RECV issuance	Socket ID	Buffer address	Data length + reception flag	Work	Work
0x300009	RECVFROM	RECVFROM issuance	Socket ID	Buffer address	Data length + reception flag	Address information pointer	Address information length
0x30000A	SETSOCKOPT	SETSOCKOPT issuance	Socket ID	Level	Option	Option information address	Option information length
0x30000B	GETSOCKOPT	GETSOCKOPT issuance	Socket ID	Level	Option	Option information address	Option information length address
0x30000C	SHUTDOWN	SHUTDOWN issuance	Socket ID	Socket shutdown method	Work	Work	Work
0x30000D	NET_END	Abnormal termination of macro	Socket ID	errno (error number)	Work	Work	Work
0x300010	NET_CTLR	IOCTL issuance	Unit number + slot number	Control information	Control information	Control information	Control information
0x300010	NET_CTLR	Remote CPU control reception	Station number + command	Frame length + transmission no.	Target type + data length	Data address	Work
0x300011	NET_START	NCP I/O startup	Socket ID	Task information	Command code + socket status	Startup information 1	Startup information 2
0x300011	NET_START	Built-in Ethernet transmission	Socket ID + ETHER_TYPE	Packet header information			
0x300012	NET_TERM	NCP termination interrupt	Socket ID	Task information	Response information	Status code	Interrupt information
0x300012	NET_TERM	Built-in Ethernet termination interrupt	Socket ID + FFFF	LANCE descriptor information (TMD0, TMD1, TMD2, TMD3)			
0x300013	NET_ATEN	NCP attention interrupt	Socket ID	Task information	Response information	Status code	Interrupt information
0x300013	NET_ATEN	Built-in Ethernet reception	Socket ID + ETHER_TYPE	Packet header information			
0x300014	NET_STO	Software timeout	Socket ID	Task information	Startup information	Startup information	Startup information
0x300015	NET_SUB	Error detection	Error type	Error information	Error information	Error information	Error information
0x300030	CYCM_CHK_S	μΣNETWORK-1000 transfer alive-monitoring start	Alive-monitoring CM address				
0x300031	CYCM_CHK_E	μΣNETWORK-1000 transfer alive-monitoring end	Number of detected nodes that are alive	Number of detected reissuances of stcycm	Number of detected nodes that are dead	Cache purge flag	Number of alive-monitoring blocks
0x300032	NET_UDP_RCV	UDP reception information for NCP-E	Source IP address	Source port no. + destination port no.	Received data length	Address information size	Reception information

Table F-1 List of DHP Codes (4/4)

Code value	DHP display name	Trace point	para1	para2	para3	para4	para5
0x400000		NXACP processing					
0x400001	NX_INITS	nx_init() start					
0x400002	NX_DFUPS	nx_dfup() start	dfn (data field number)	nmode (node mode)	mmode (message mode)		
0x400003	NX_DFDOWNS	nx_dfdwn() start	dfn (data field number)	bfsz (buffer size)			
0x400004	NX_DF_QUITS	nx_quit() start					
0x400005	NX_PUTS	nx_put() start	dfn (data field number)	mgn (multi-cast group number)	tcd (transaction code)	len (message size)	
0x400006	NX_GETS	nx_get() start	tmout (timeout monitoring period)		offset (offset)		
0x400007	NX_GET_INQS	nx_get_inq() start					
0x400008	NX_PUT_REPLYS	nx_put_reply() start					
0x400021	NX_INITE	nx_init() end	rtncd (process result)				
0x400022	NX_DFUPE	nx_dfup() end	dfn (data field number)	rtncd (process result)			
0x400023	NX_DFDOWNE	nx_dfdwn() end	rtncd (process result)	sa (source address)			
0x400024	NX_DF_QUITE	nx_quit() end	rtncd (process result)				
0x400025	NX_PUTE	nx_put() end	rtncd (process result)				
0x400026	NX_GETE	nx_get() end	rtncd (process result)		da (source address)	tcd/len	
0x400027	NX_GET_INQE	nx_get_inq() end					
0x400028	NX_PUT_REPLYE	nx_put_reply() end					
0x500000		CPMS library processing					
0x500026	SYSDO	`sysdo` issuance	arg				
0x500027	LEDCTL	`ledctl` issuance	ctl				
0x500028	WDTSET	`wdtset` issuance	msec				
0x500029	CARDSTAT	`cardstat` issuance	slot	point	stat		
0x50002A	CARDOFF	`cardoff` issuance	slot				
0x50002B	CARDON	`cardon` issuance	slot				
0x50002C	DSUSTAT	`dsustat` issuance	stat				
0x50002D	DSUCTL	`dsuctl` issuance	ctl				
0x50002F	DCMCTL	`dcmctl` issuance	ctl	arg			
0x500032	WRTMEM	`wrtmem` issuance	from	to	cnt		
0x50004C	ROMREAD	`romread` macro issuance	slot	bank	size	saddr	--
0x50004D	ROMWRITE	`romwrite` macro issuance	slot	bank	size	saddr	--
0x5000AB	CPUSTPCTL	`cpustpctl` macro issuance	Life-sharing mode				
0x5000CD	LDRSTPSTAT	ldrstpstat issuance					
0x5000CE	LDRSTPCTL	`ldrstpctl` issuance	cmd	data			
0x5000D0	LDRSIMUSTAT	ldrsimustat issuance					
0x600000		For user					
0x600000	USR0	User definition					
0x600001	USR1	User definition					
0x600002	USR2	User definition					
0x600003	USR3	User definition					
0x600004	USR4	User definition					
0x600005	USR5	User definition					
0x600006	USR6	User definition					
0x600007	USR7	User definition					