HITACHI

Software Manual

Operation

NXACP For Windows®

S10VE

SEE-3-134（A）

Software Manual

Operation

NXACP For Windows®

S10VE

First Edition, October 2020, SEE-3-134(A)

<IC> (FL-MW2007)

SAFETY PRECAUTIONS

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

⚠ **DANGER**	: This safety label identifies precautions that, if not heeded, will result in death or serious injury.
⚠ **WARNING**	: Identifies precautions that, if not heeded, could result in death or serious injury.
⚠ **CAUTION**	: Identifies precautions that, if not heeded, could result in minor or moderate injury.
NOTICE	: This safety label without a safety alert symbol identifies precautions that, if not heeded, could result in property damage or loss not related to personal injury.

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

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

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

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

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

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

Revision record

Revision No.	Revision record (revision details and reason for revision)	Month, Year	Remarks
A	First edition	October 2020	

PREFACE

NXACP is a communication platform for building a cooperative autonomous distributed system on the S10VE system. This manual describes NXACP's features and linkage (interface) specifications of macro calls.
Note that the target audience of this manual is people who have a basic knowledge of CPMS (real-time OS used in this system), Windows®, and sockets used on a workstation (hereinafter abbreviated as WS).

<Organization of this manual>

PART 1 OVERVIEW
Describes the features, system configuration, basic terminology, and functional overview of NXACP.

PART 2 FUNCTION GUIDE
Describes the basic functions of NXACP necessary for system construction, program development, test, and operation, by organizing them into the following 8 chapters.

CHAPTER 1 MULTICAST COMMUNICATION FUNCTION
CHAPTER 2 DATA FIELD MANAGEMENT FUNCTION
CHAPTER 3 DUPLEXED LAN CONTROL FUNCTION
CHAPTER 4 TEST FUNCTION
CHAPTER 5 SYSTEM MANAGEMENT FUNCTION
CHAPTER 6 OPERATION MANAGEMENT FUNCTION
CHAPTER 7 NETWORK - SHARED MEMORY FUNCTION
CHAPTER 8 SYSTEM CONSTRUCTION FUNCTION

PART 3 MACRO SPECIFICATIONS
Describes the functions and linkage specifications of macro calls offered by NXACP.

<Related manuals>

Manual name	Manual number
S10VE User's Manual General Description	SEE-1-001
User's Manual REPLACEMENT for S10VE	SEE-1-146
S10VE Software Manual CPMS General Description and Macro Specifications	SEE-3-201
S10VE Software Manual Operation RPDP for Windows®	SEE-3-133
NX Dlink GUIDE	RS90-63-X050
NX Dlink REFERENCE	RS90-63-X051

* The letter X in the manual number indicates a revision number.

<Trademarks>

Microsoft® Windows® is a registered trademark of Microsoft Corporation in the United States and other countries.

<Note for storage capacity calculations>

- Memory capacities and requirements, file sizes and storage requirements, etc. must be calculated according to the formula 2^n. The following examples show the results of such calculations by 2^n (to the right of the equals signs).
  1 KB (kilobyte) = 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

FIGURES

TABLES

PART 1 OVERVIEW

CHAPTER 1 OVERVIEW

1.1 Introduction to NXACP

NXACP is a generic term for an autonomous distributed online package for programmable controllers. NXACP/S10VE is a communication platform targeted for the S10VE system. NXACP software also works together with NX Dlink (an autonomous distributed online package) running on a WS or personal computer (hereinafter abbreviated as PC), in a coordinated manner to provide an autonomous distributed network environment.
NXACP/S10VE supports the following functions listed as "Category A" in "Open Autonomous Distributed Interface Technical Manual V1.1".

Table 1-1 List of Category A Communication Functions

Primary function			Secondary function		
Class name	Function name		Class name	Function name	
A-Base-1	Multicast communication function	Y	A-Opt-1-a	Duplexed LAN control multicast communication function	Y
A-Base-2	Alive signal transmission function	Y	A-Opt-2-a	Fault information transmission function	Y
A-Opt-3	One-to-one communication function (up to 16 KB)	N			
A-Opt-4	Construction-via-network function	N			

Y: Supported, N: Not supported

[Note] Priority control defined in the cooperative autonomous distributed protocol specifications is not supported.
NXACP/S10VE always sends messages at priority level (PRI) 0.
Received messages are processed in order of reception, not according to priority level.

The following shows the main communication specifications of NXACP/S10VE.

Table 1-2 List of Communication Specifications

Item	Specification
Network	Ethernet
Communication protocol	UDP/IP
Message size	Up to 16384 bytes
Number of registered local data fields	Up to 7 DFs (including local node communication)
Number of registered remote data fields	Up to 8 DFs per local data field
Total number of registered data fields	Up to 16 DFs
Number of registered send multicast groups	Up to 128 groups
Number of registered receive multicast groups	Up to 32 groups (per DF)
Number of output transaction codes	Up to 8 TCDs per task (per DF)
Number of input transaction codes	Up to 8 TCDs per task (per DF)
Number of tasks to which the same transaction code is input	Up to 8 tasks per TCD (per DF)

[Note] For information about the valid selectable range of each item, see "PART 2 CHAPTER 8 SYSTEM CONSTRUCTION FUNCTION".

1.2 Hardware Configuration

Figures 1-1 and 1-2 describe hardware configurations necessary for using the features of this system.

- Minimum configuration
  The minimum configuration comprises of an S10VE and one of the following: a POC, a PADT, or a PC, to which the S10VE is connected through the Built-in Ethernet module by using Ethernet.

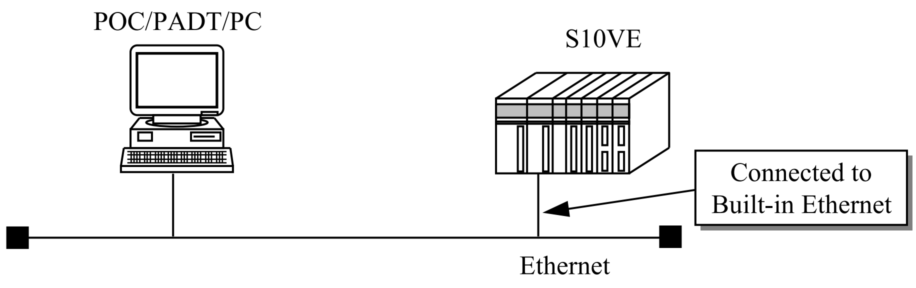


Figure 1-1 Minimum Network Configuration

- Maximum configuration
  The maximum configuration uses two network lines (Duplexed built-in Ethernet).

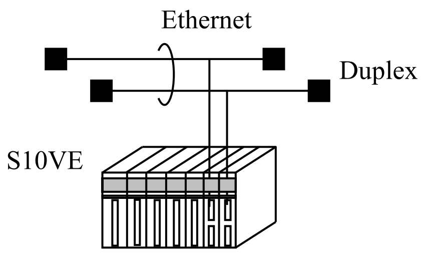


Figure 1-2 Maximum Network Configuration

You can use the routing function to communicate with (send data to and receive data from) another network over routers or gateways. However, an S10VE can communicate with up to 8 network segments by way of one directly-connected network segment. In total, an S10VE can communicate with up to 16 network segments including the directly-connected segments and segments connected by way of routers or gateways (up to 15 if you define a data field for local node communication).

1.3 Software Configuration

Figure 1-3 shows the position and the software configuration of this system. The following pieces of software are prerequisites.

- CPMS/S10VE (Realtime monitoring system)
- RCTLNET/S10VE (Ethernet or μΣNETWORK-1000 driver)
- RPDP/S10VE (Realtime system development package)
- BASE SYSTEM/S10VE (basic system)

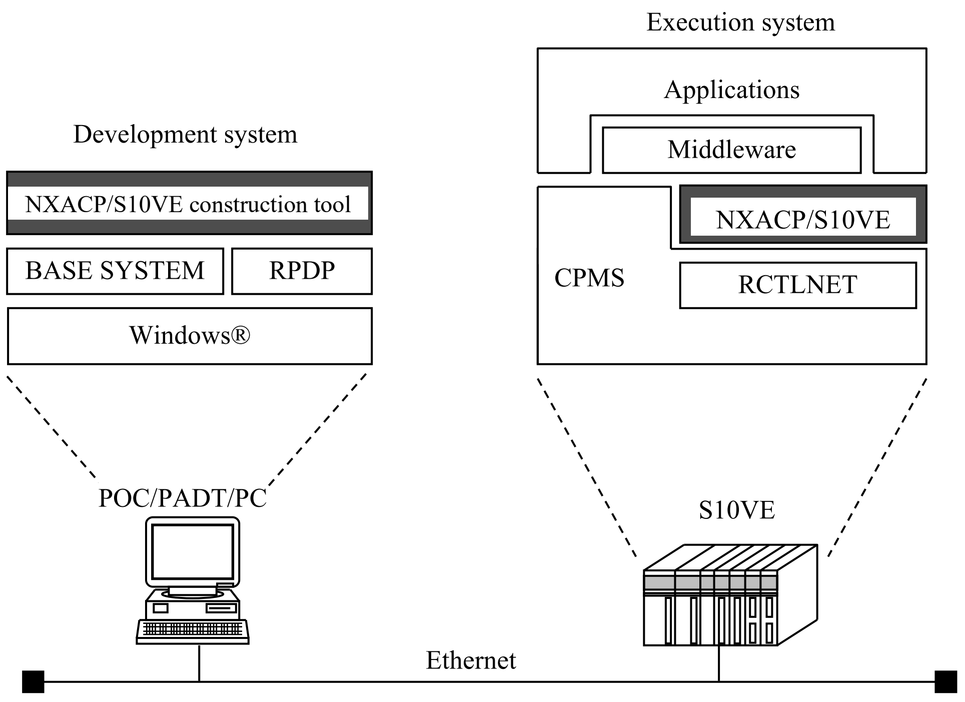


Figure 1-3 Software Configuration

CHAPTER 2 DEFINITIONS OF TERMINOLOGY

2.1 Definitions of Basic Terminology

NXACP is constructed using the elements shown in Figure 1-4.
Unlike previous network packages, NXACP uses abstract terms that do not depend on how the network is constructed.
This section describes each element shown in Figure 1-4. The numbered descriptions below the figure pertain to the numbers in parentheses () in Figure 1-4.

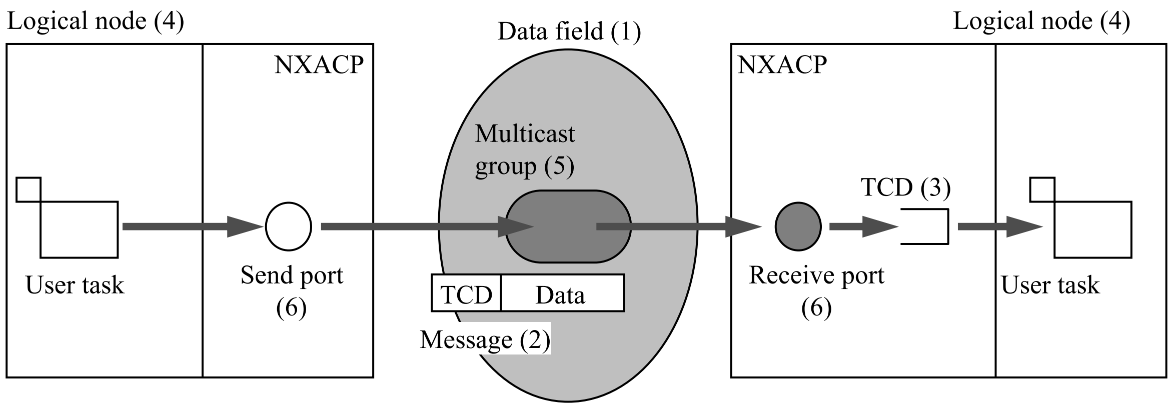


Figure 1-4 Description of NXACP Elements

(1) Data field

A data field (DF) is defined as an area that circulates messages with specific attribute information. In NXACP, a data field is defined as one LAN segment, and includes multiple devices. The purpose of using a data field is to define the scope in which messages can be transmitted.

A "data field number" (DFN) is assigned to each data field for identification purposes in a distributed system.

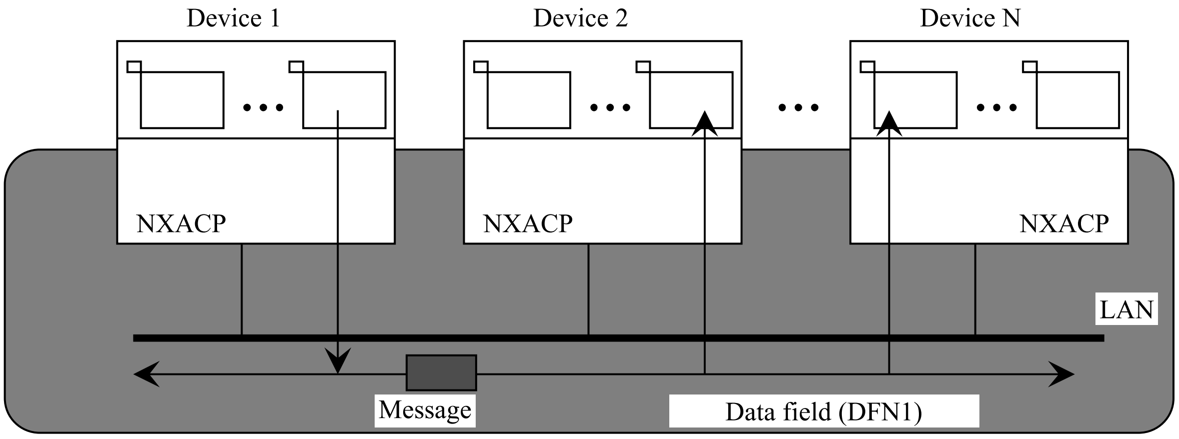


Figure 1-5 Data Field

There are two types of data fields: local data field and remote data field.

- Local data field: Data field to which the local node is directly connected
- Remote data field: Data field to which the node is indirectly connected through routers or gateways

(Example) Node 1 is directly connected to Data field 1, which is therefore a local data field of Node 1, and Data fields 2 and 3 are remote data fields of Node 1.

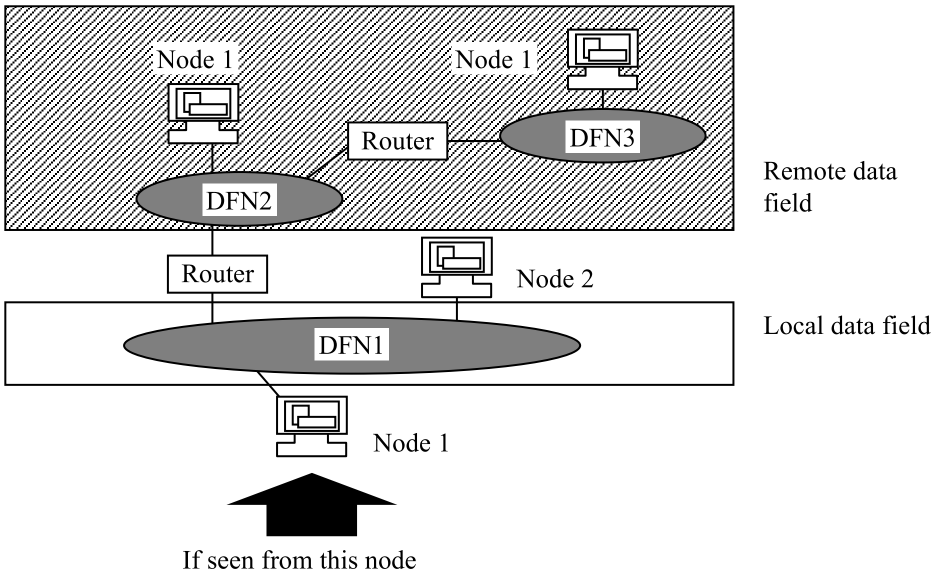


Figure 1-6 Local Data Field and Remote Data Field

(2) Message

A message is a basic unit of transaction data transmitted within NXACP, and consists of user data and NX headers.

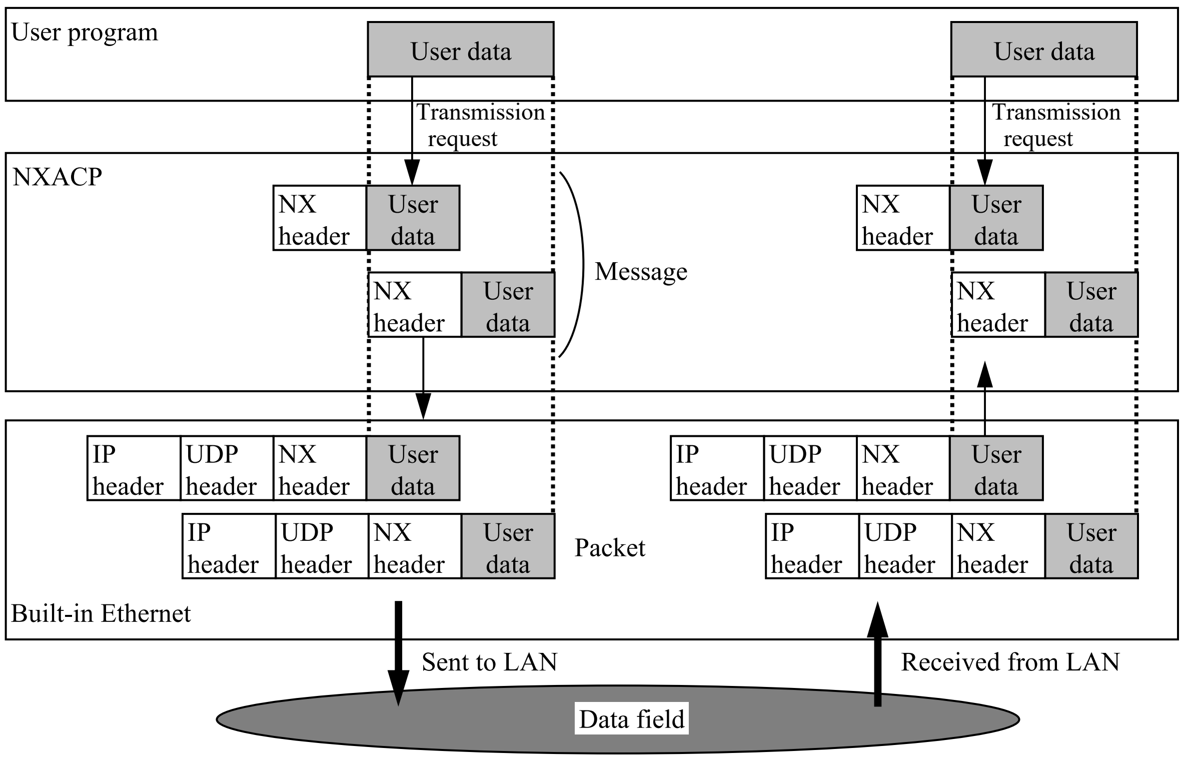


Figure 1-7　Message

(3) Transaction code

A transaction is a business deal or process of doing business, related to products, services, and real-time events. Purchasing goods in a shop and receiving tourist services at a travel agency are both examples of business deals that can be called “transactions”. Transaction processing means to execute, monitor, and record such transactions.
For this reason, everything a business application does after receiving data, such as plant data and stock exchange data (including database update processing and notification processing to other business applications), can be called transaction processing.
In fact, all data that circulates in a data field is processed as a transaction by this system.
An identifier for a transaction is called a transaction code (TCD).
Correspondence between TCDs and messages can be defined by a user as desired, in the same way as a function code (FC) supported by LNSP. A user program specifies a TCD when sending and receiving a message.
The TCD specified by a user program is stored in the NX headers of each message.

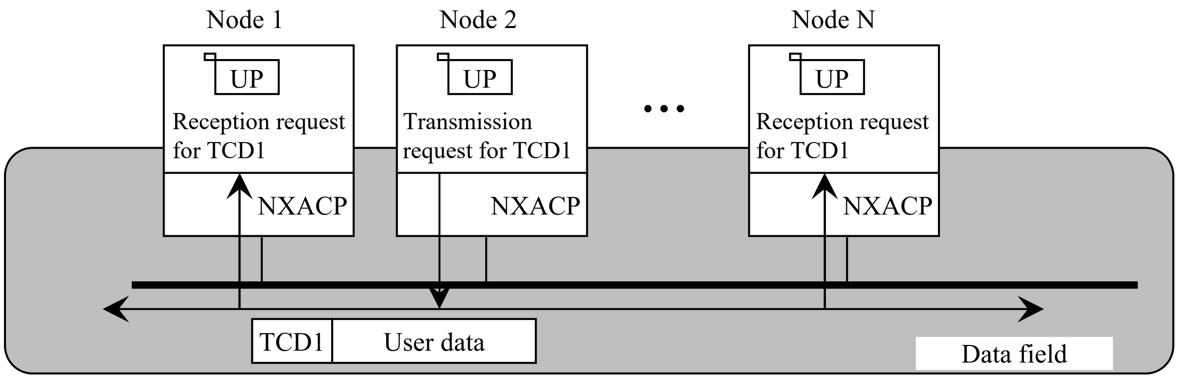

Node 1
Node 2
Node N
UP
Reception request for TCD1
UP
Transmission request for TCD1
UP
Reception request for TCD1
NXACP
NXACP
NXACP
TCD1
User data
Data field

Figure 1-8 Transaction Code

(4) Logical node
A device that belongs to a data field is called a logical node. A number called a logical node number (LNN) is assigned to each logical node so that each logical node in the data field can be uniquely identified. In NXACP, a CPU is handled as a logical node.

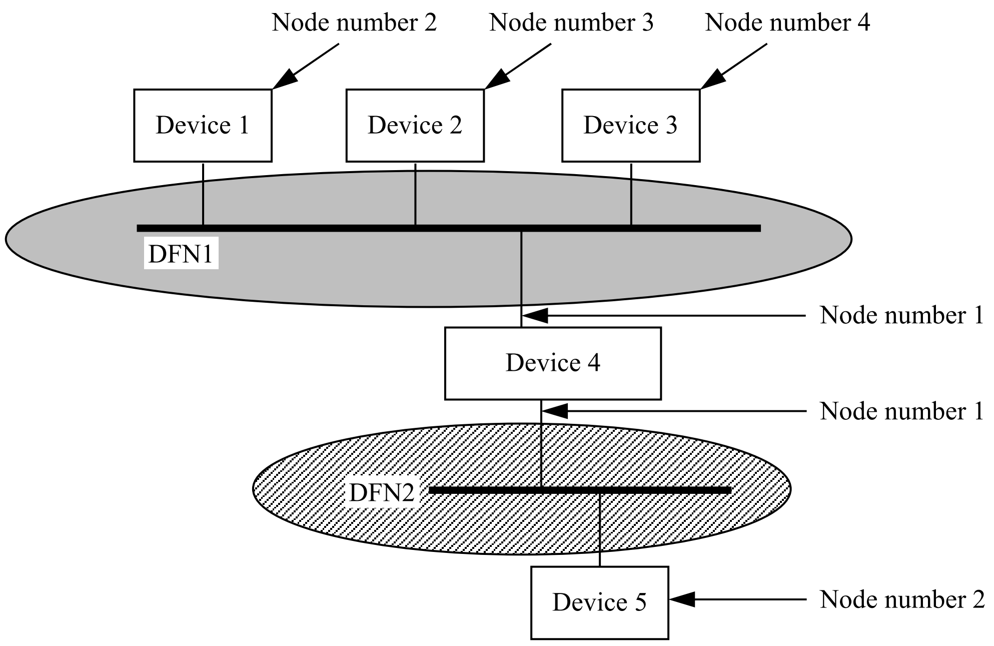


Figure 1-9 Logical Node

Each node has a mode that indicates whether it is running online, or undergoing tests. A node that is running online is called an online node, and a node that is undergoing tests is called a test node.

(5) Multicast group
A multicast group is a group that is defined based on whether to receive each message that is broadcast to the data field from a logical node. Multicast groups are managed on a per-data-field basis. Multiple multicast groups can be defined in one data field. A logical node in the data field can belong to multiple multicast groups.
Each multicast group is mapped to one pair of port numbers (online and test port numbers). A number called a multicast group number (MGN) is assigned to each multicast group so that each multicast group in the data field can be uniquely identified.

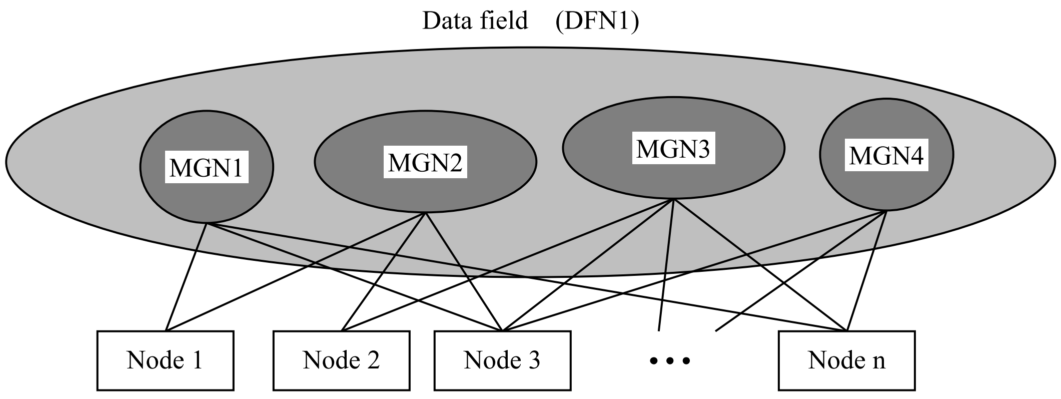


Figure 1-10 Multicast Node 1 Group

(6) Port

Relay points called “send ports” and end points called “destination ports” must be defined in order to manage the attributes of the senders and receivers in a unified way when you send and receive messages using the same physical transmission network. Note that the port numbers mapped from a multicast group are destination port numbers.

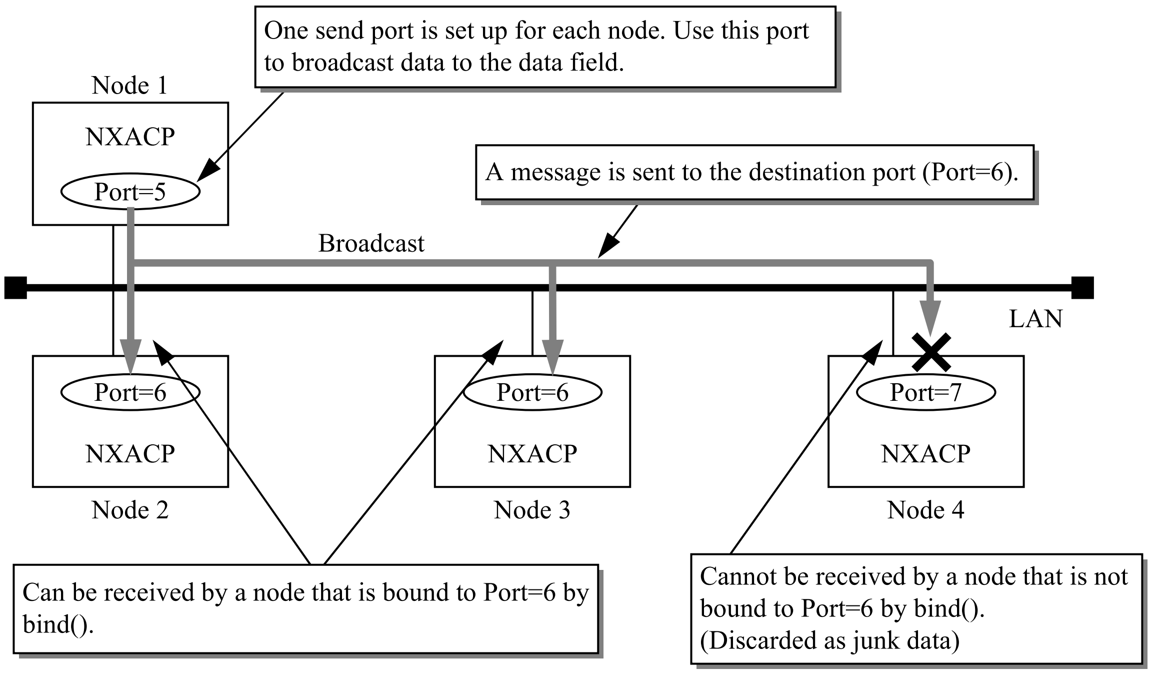

One send port is set up for each node. Use this port to broadcast data to the data field.
Node 1
NXACP
Port=5
A message is sent to the destination port (Port=6).
Broadcast
LAN
Port=6
NXACP
Node 2
Port=6
NXACP
Node 3
Port=7
NXACP
Node 4
Can be received by a node that is bound to Port=6 by bind().
Cannot be received by a node that is not bound to Port=6 by bind(). (Discarded as junk data)

Figure 1-11 Port

If port number 6 above is mapped from Multicast group 1, Nodes 2 and 3 therefore belong to Multicast group 1. Mapping between multicast group numbers and port numbers must be uniquely defined for all nodes in the same data field.
In NXACP, a destination port number is specified for a send multicast group, and a receive port number (port bound by bind()) is specified for a receive multicast group in the construction information.

CHAPTER 3 FUNCTIONAL OVERVIEW

3.1 Multicast Communication Function

Multicast communication is the basis of NX protocol, and has the following characteristics.

- Efficient data transmission. You can send the same message to multiple nodes at the same time.
- Easy to use. It is not required to handle node dependencies when sending or receiving messages. A procedure to establish a connection is not necessary.

When sending messages, the sender node specifies information such as the target data field, multicast group, and transaction code. Each receiver node autonomously receives only necessary messages.

As a result, for example, “recognition of communication peers” and “synchronization with communication peers” are no longer necessary, thus increasing independence between devices or user programs, while enhancing system expandability.

In multicast communication, data is broadcast to the network segment corresponding to a specified data field number. If a multicast group or transaction code for data is not specified for the receiver node, the data is discarded as unnecessary messages by that node.

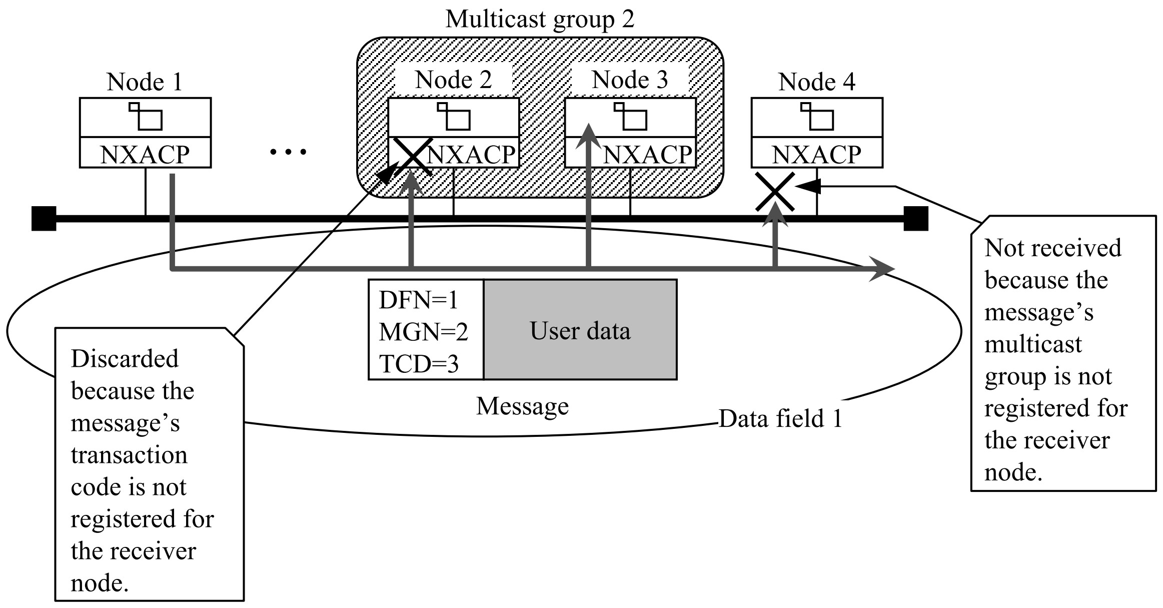

Multicast group 2
Node 1
Node 2
Node 3
Node 4
NXACP
NXACP
NXACP
NXACP
DFN=1
MGN=2
TCD=3
User data
Message
Data field 1
Discarded because the message’s transaction code is not registered for the receiver node.
Not received because the message’s multicast group is not registered for the receiver node.

Figure 1-12 Multicast Communication

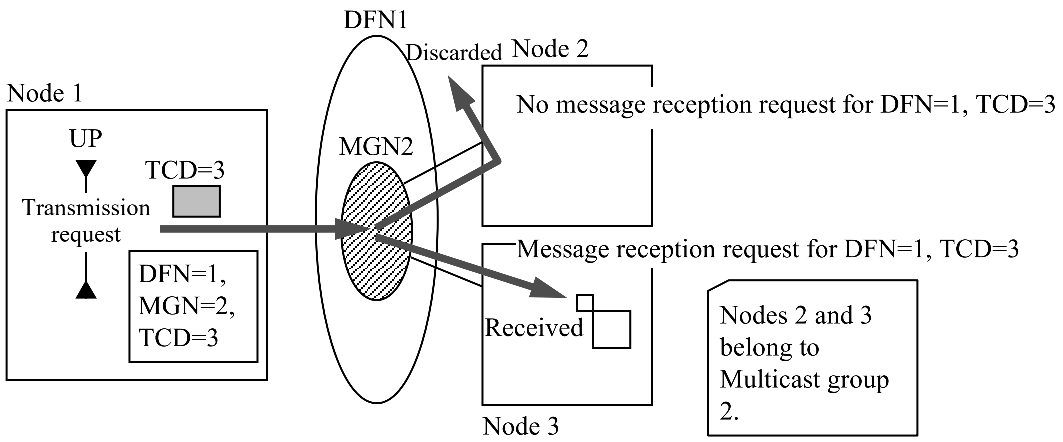

DFN1
Discarded
Node 2
No message reception request for DFN=1, TCD=3
Node 1
UP
TCD=3
Transmission request
MGN2
DFN=1, MGN=2, TCD=3
Message reception request for DFN=1, TCD=3
Received
Node 3
Nodes 2 and 3 belong to Multicast group 2.

Figure 1-13 Multicast Group

A user can send/receive a message by specifying the TCD of the sending/receiving message and issuing the nx_put()/nx_get() macro respectively. User data is then handled as follows.

(1) 16 KB data transmission/reception

A user program (UP) can send or receive up to 16 KB of data in one transmission/reception request. When data is actually transmitted over the network, the data is split into multiple network frames (MTU) and transmitted as packets. Splitting and reassembling the data is controlled by NXACP.

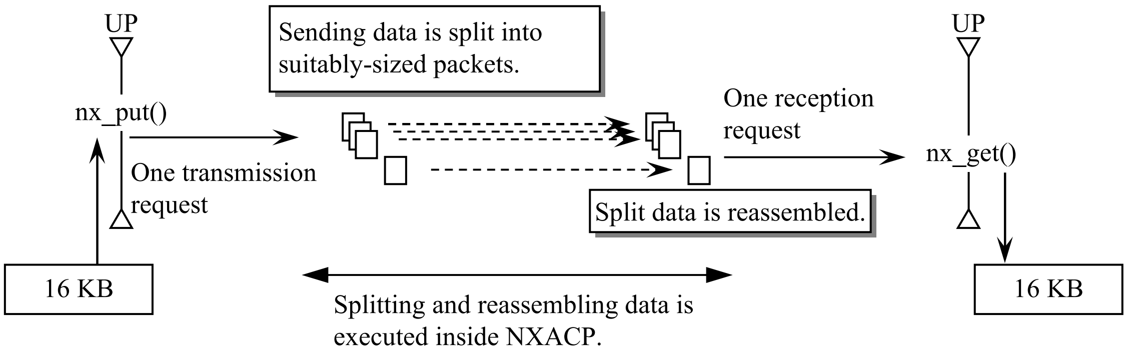

UP
Sending data is split into suitably-sized packets.
UP
nx_put()
One transmission request
One reception request
nx_get()
Split data is reassembled.
16 KB
Splitting and reassembling data is executed inside NXACP.
16 KB

Figure 1-14 Splitting and Reassembling a Message

(2) Transmission/reception area

The area where a message is sent to (transmission area of the message) differs depending on the specified data field number.

If 0 is specified for DFN in a transmission request, the message is only sent to within the local node.

If a non-zero value is specified for DFN in a transmission request, the message is only sent to the network and cannot be sent to within the local node.

In the same way, a message from within the local node can be received when 0 is specified for DFN. If you want to receive a message from another node connected over the network, specify a non-zero value for DFN.

When you send data, you must specify the data field number as a parameter of the nx_put() macro.

When you receive data, specify the data field number in the construction information.

(3) Sharing a TCD and receiving pinpoint data

One message can be received by multiple user programs. To increase the efficiency of communication, it is recommended that a sender packs multiple business data items into a single message of about the packet size (MTU size) rather than sending them in multiple messages smaller than the MTU size and that, at the receiver side, a single message with multiple business data items is received by multiple user programs.

Note that each receiver user program can receive only the necessary part of the data. By doing this, you can minimize the memory usage and time required to copy data.

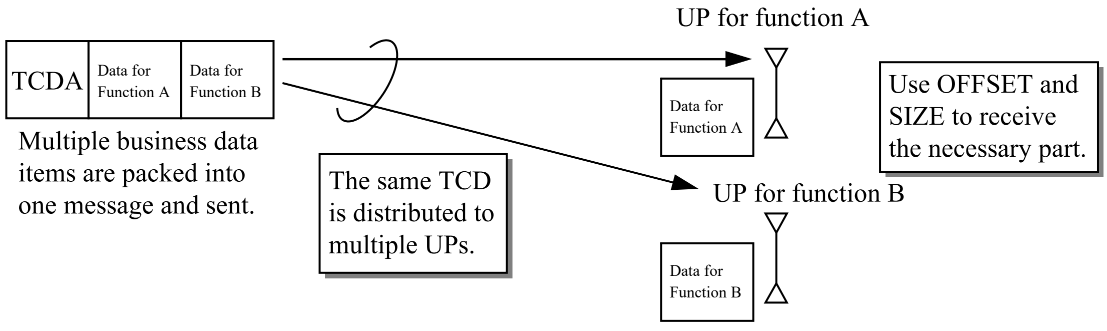

UP for function A
TCDA
Data for Function A
Data for Function B
Data for Function A
Use OFFSET and SIZE to receive the necessary part.
Multiple business data items are packed into one message and sent.
The same TCD is distributed to multiple UPs.
UP for function B
Data for Function B

Figure 1-15 Receiving Pinpoint Data in a Message

[Note: Filtering function]
In the S10VE, RCTLNET can filter out unnecessary messages. This is called the “filtering function”.
The filtering function determines which received messages are necessary, and allows unnecessary messages to be discarded in order to reduce CPU load.
In the S10VE, the filtering function filters out messages based on port number. For this reason, for a user of NXACP, the filtering function filters out messages based on multicast group number.
In NXACP, RCTLNET discards messages addressed to the multicast group numbers for which the user does not define construction information.

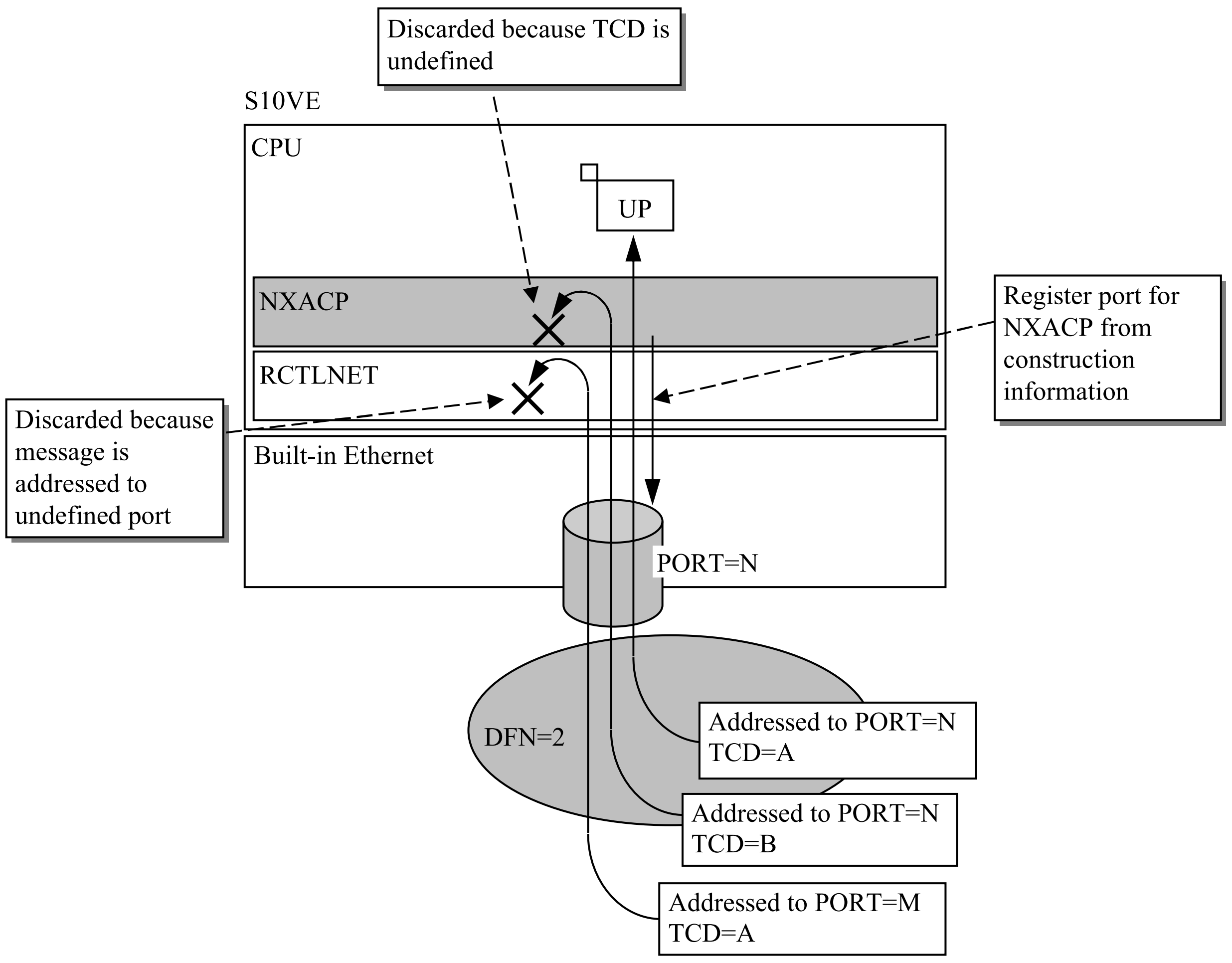

Discarded because TCD is undefined
S10VE
CPU
UP
NXACP
RCTLNET
Register port for NXACP from construction information
Discarded because message is addressed to undefined port
Built-in Ethernet
PORT=N
DFN=2
Addressed to PORT=N TCD=A
Addressed to PORT=N TCD=B
Addressed to PORT=M TCD=A

Figure 1-16 Message Filtering

3.2 Data Field Management Function

Data field management monitors the connection status of the nodes connected to a data field, and also notifies the status of the local node to other nodes.
In NXACP, inside each data field, each node in the data field periodically transmits a message called an "alive signal". This message allows the connection status between each node in the data field to be monitored by other nodes, allowing the connection status of the whole system to be managed.

- Alive signal transmission
  NXACP starts sending an alive signal message periodically when NXACP receives a start request from a user. When an alive signal message is sent, information such as the node mode, system startup time, IP address, and alive signal mode is also included in the message. The transmission cycle for alive signal messages is specified by a user in the construction information.

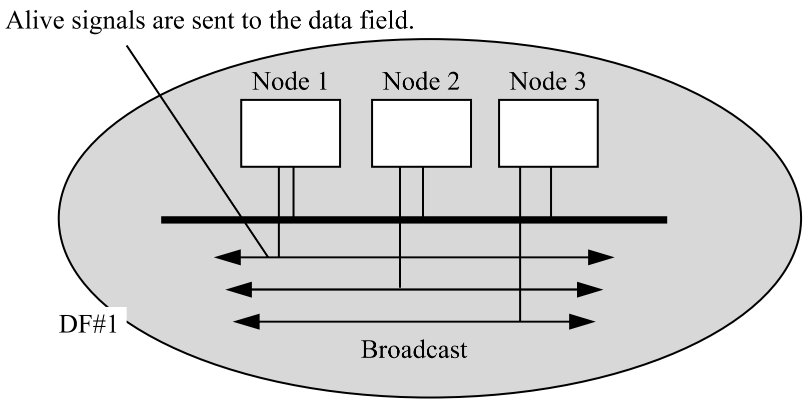
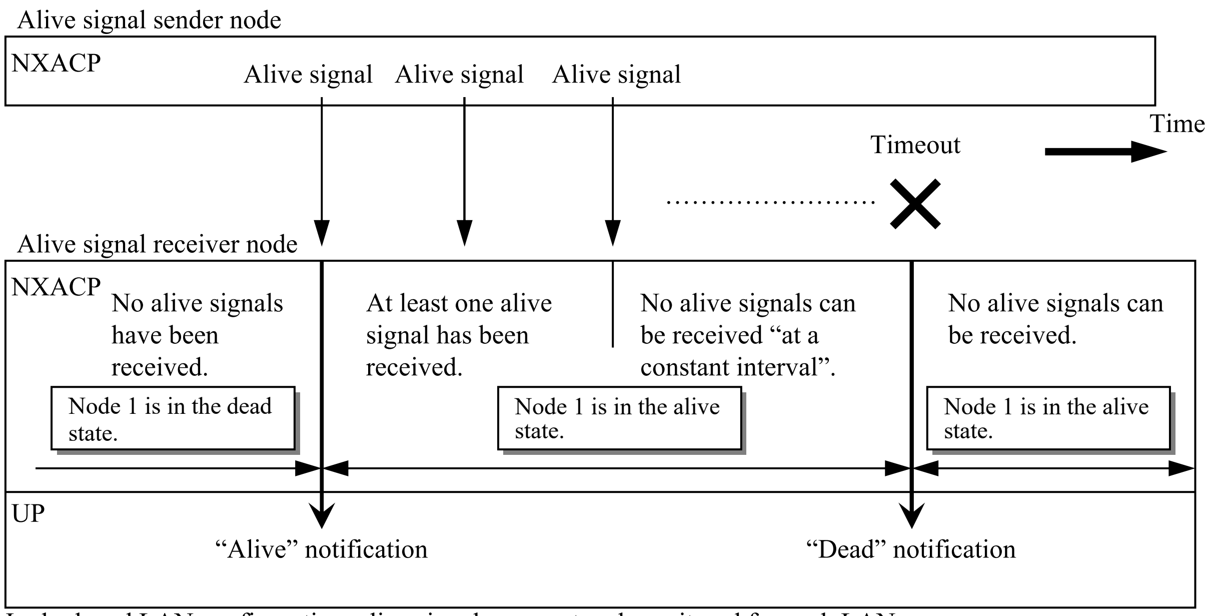


Figure 1-17 Alive Signal

- Monitoring the connection status of nodes
  This function monitors the connection status of other nodes connected to a data field. If the status changes, the change will be notified to a user. The following status changes will be reported: “Alive: Alive signals have just started to be received.” and “Dead: Timeout monitoring time has passed since alive signals stopped.” You can also check the current status using the nx_dfsts() macro.


Alive signal sender node
NXACP
Alive signal
Alive signal
Alive signal
Time
Timeout
Alive signal receiver node
NXACP
No alive signals have been received.
At least one alive signal has been received.
No alive signals can be received “at a constant interval”.
No alive signals can be received.
Node 1 is in the dead state.
Node 1 is in the alive state.
Node 1 is in the alive state.
UP
“Alive” notification
“Dead” notification

In duplexed LAN configuration, alive signals are sent and monitored for each LAN.

Figure 1-18 Node Status Management

In NXACP, sending alive signals automatically starts after a start request from a user. However, for monitoring the status of other nodes, a user can specify whether to enable the monitoring (in the construction information). The purpose of this is to allow you to reduce the load of the controller for receiving alive signals and monitoring the status change of nodes. If it is not required for a node to monitor the status change of other nodes, we recommend running that node with its connection status monitoring turned off.
If WSs or PCs are connected to the network system, we recommend, if possible, using them to monitor the status change of nodes, and not allow controllers to receive alive signals.

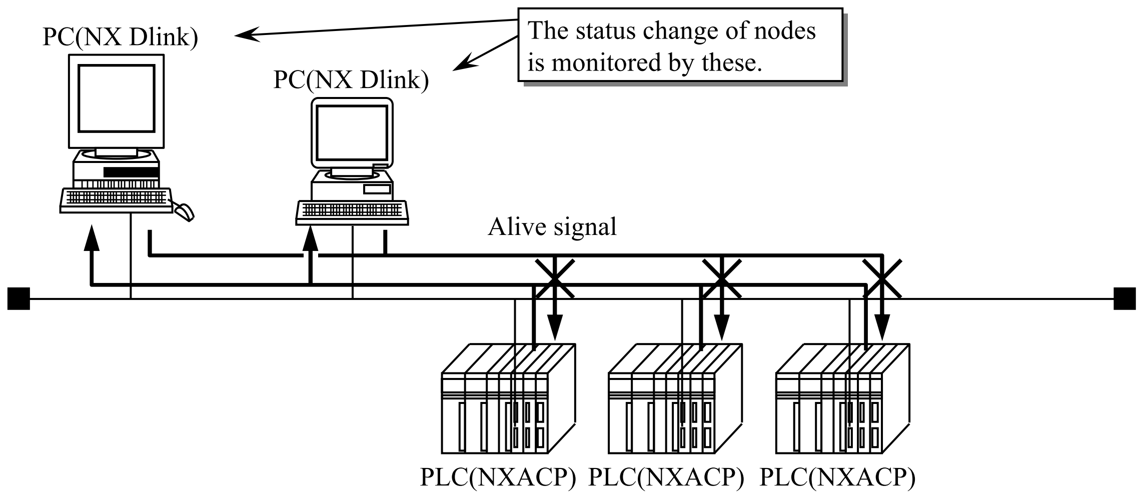


Figure 1-19　Status Monitoring Node (1)

If you must monitor the status change of nodes when WSs or PCs are not connected to the network system, we recommend assigning specific controllers for this purpose, and not allow the other controllers to receive alive signals. In this case, we also recommend that the controllers in charge of monitoring the status change of nodes are dedicated to data field monitoring tasks only.

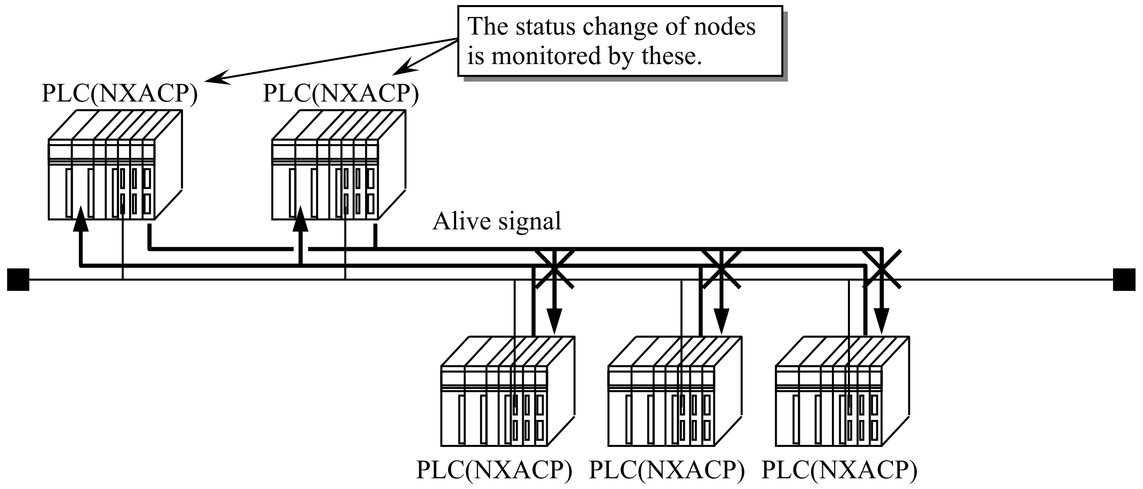


Figure 1-20　Status Monitoring Node (2)

3.3 Duplexed LAN Control Function

In NXACP, you can duplex the network that constitutes a data field. The reliability of data communication can be improved by using duplexed LANs for a data field. To use this function, you must set up duplexed-LAN-related information in the data field definition at system construction.
Even when duplexed LANs are used for a data field, a user sees the two physical LANs as one data field (as shown in the following figure), and does not have to handle the two LANs directly.

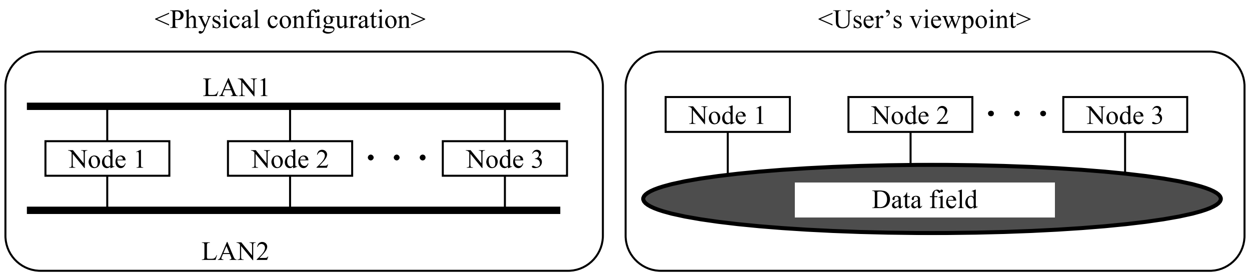


Figure 1-21 Duplexed LANs

When duplexed LANs are used, NX multicast communication uses dual paths, therefore, a user does not have to deal with complex configuration management (including master/slave arbitration of the network).
This communication method has the following characteristics.

- A message is sent to both LANs.
- A message is received from both LANs. If the same message is received twice, the message that arrives first is accepted and the later one discarded.
- A user does not have to manage master/slave arbitration of the network.

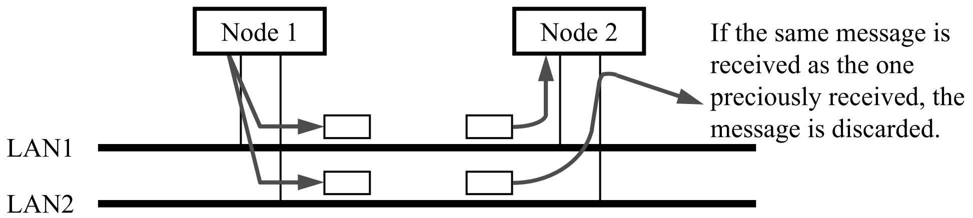

Node 1
Node 2
If the same message is received as the one preciously received, the message is discarded.
LAN1
LAN2

Figure 1-22 Duplexed LAN Communication Method

In the same way as the multicast communication described above, when sent periodically by NXACP, alive signals are sent over dual paths, and data field connection of nodes is monitored independently for each LAN.

[Caution]
You cannot specify a particular LAN to be used for communication if you send and receive data over duplexed LANs.

3.4 Test Function

The test function allows you to test each logical node independently on a system currently running online.

If a node is currently running online, the node is called an online-mode node (hereinafter abbreviated as “online node”). If a node is currently undergoing tests, the node is called a test-mode node (hereinafter abbreviated as “test node”). By setting the nodes you want to test to the test mode, they can be tested without disturbing online nodes.

A mode is also given to a message transmitted over a data field as follows: a message sent by an online node is called an online-mode message (hereinafter abbreviated as “online message”), and a message sent by a test node is called a test-mode message (hereinafter abbreviated as “test message”). By doing so, the following test configurations can be used.

- Messages are sent as test messages from test nodes. Only online messages are received by online nodes. This allows tests to be performed without disturbing online nodes in the online environment.
- By allowing online messages to be received by test nodes in the settings, you can perform tests using online data.

In NXACP, for each test node, you can select one of the following two message modes that determine how messages are received at the test node. You can construct a variety of test system configurations other than the ones described above.

- Message mode that allows only online messages to be received
- Message mode that allows only test messages to be received

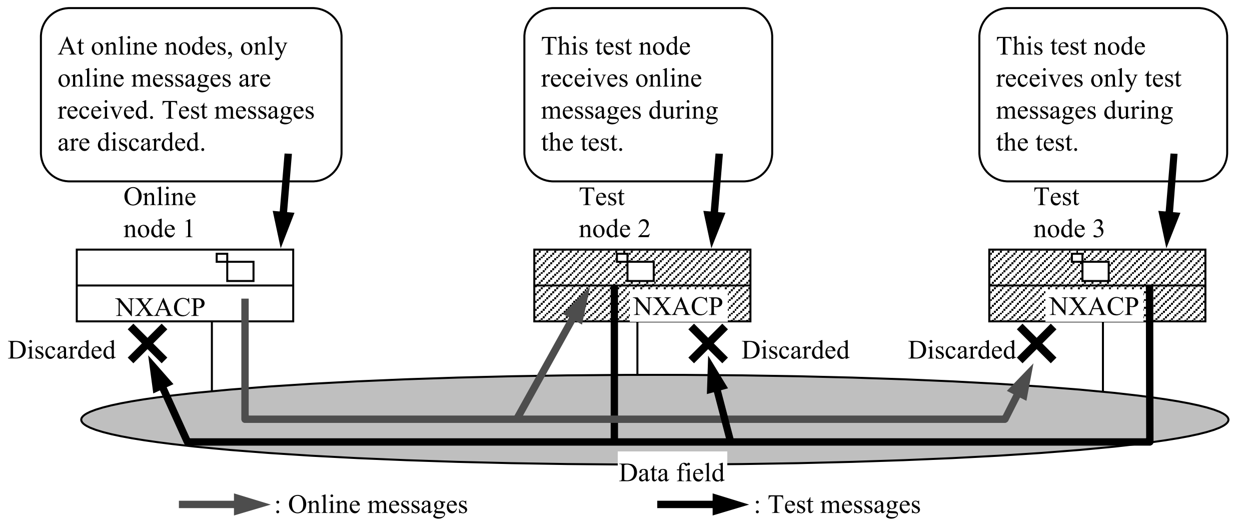

At online nodes, only online messages are received. Test messages are discarded.
This test node receives online messages during the test.
This test node receives only test messages during the test.
Online node 1
Test node 2
Test node 3
NXACP
NXACP
NXACP
Discarded
Discarded
Discarded
Data field
: Online messages
: Test messages

Figure 1-23 Test Function

[Note]

As described above, test messages are not allowed to enter online nodes, and the online nodes should not be affected. But when you use the test function, the network load must be considered because test messages are transmitted over the network.

Note that the network-shared memory function is not under the control of the message I/O control function with node modes and message modes. As for the software transfer function, this function is under the control of the message I/O control function with node modes and message modes.

3.5 System Management Function

This system has the following system management features.

- Failure notification/logging function
- DHP acquisition/control trace function
- User task state management function

● Failure notification/logging function

The following types of failures are reported to a user using the error log, EAS, or the IRSUB dedicated to failure notification.
A user can detect these types of failures in a timely manner, and take appropriate action.

- Communication failure
  NX protocol failures and socket failures are reported. Hardware-related failures are reported by RCTLNET.
- Change in resource usage (for example, buffers)
  When usage values of resources (such as buffers) in the NXACP specified by a user in the construction information exceed threshold values or overflow, the situation is reported.

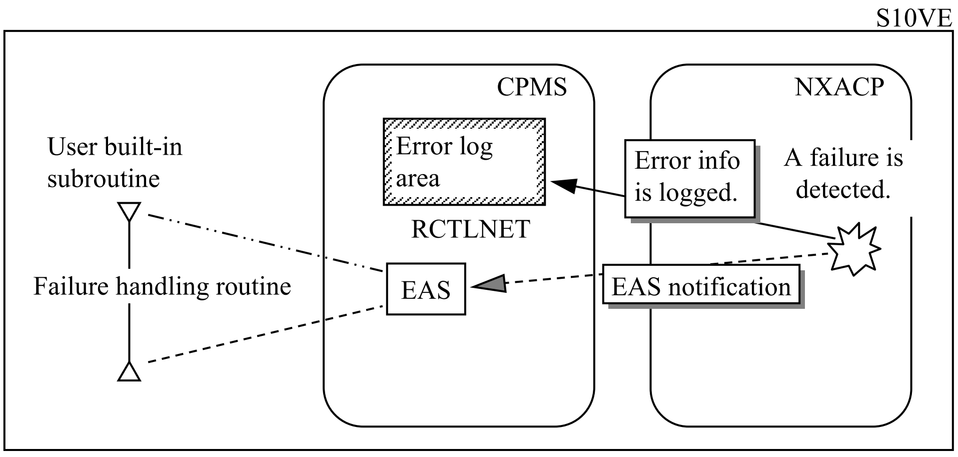


Figure 1-24 Failure Notification

In the case of node connection status change, it links to (calls) the IRSUB dedicated for change notification. When you use the alive monitoring function for other nodes, status change of the connection to the DF is reported for each LAN and for each node.

● DHP acquisition/control trace function

CPMS keeps track of when one of the predefined processing points is passed.
These records are recorded in the DHP buffer in order of occurrence. The information recorded in the DHP buffer is retrieved along with error information when an error occurs, and can be retrieved using a command. If a failure occurs, you can use this information and access information of other subsystems to analyze the behavior of a user task.
At the same time, the control information of sent and received messages is recorded in the trace buffer in order of occurrence. A user must specify whether to use the trace function and buffer area size at system construction.

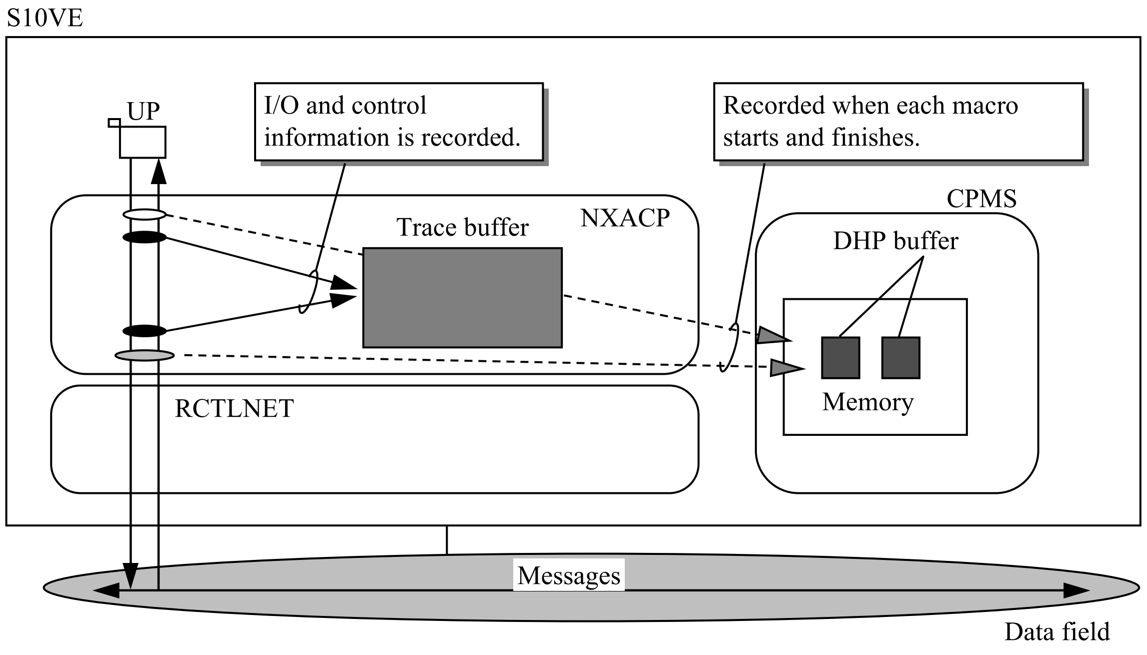

S10VE
UP
I/O and control information is recorded.
Recorded when each macro starts and finishes.
NXACP
Trace buffer
CPMS
DHP buffer
Memory
RCTLNET
Messages
Data field

Figure 1-25 Trace

- User task state management function
  This function monitors the states of user tasks that receive messages. The purpose of this function is to prevent buffer overflow when messages cannot be received.
  NXACP checks the states of user tasks when a message is received, and if the state of a user task is DORMANT, IDLE, or NON-EXIST, the message is not distributed to the task.
  In addition, when a user task is transitioning to DORMANT, the messages currently being processed to be received by the task or going to be received by the task are discarded.

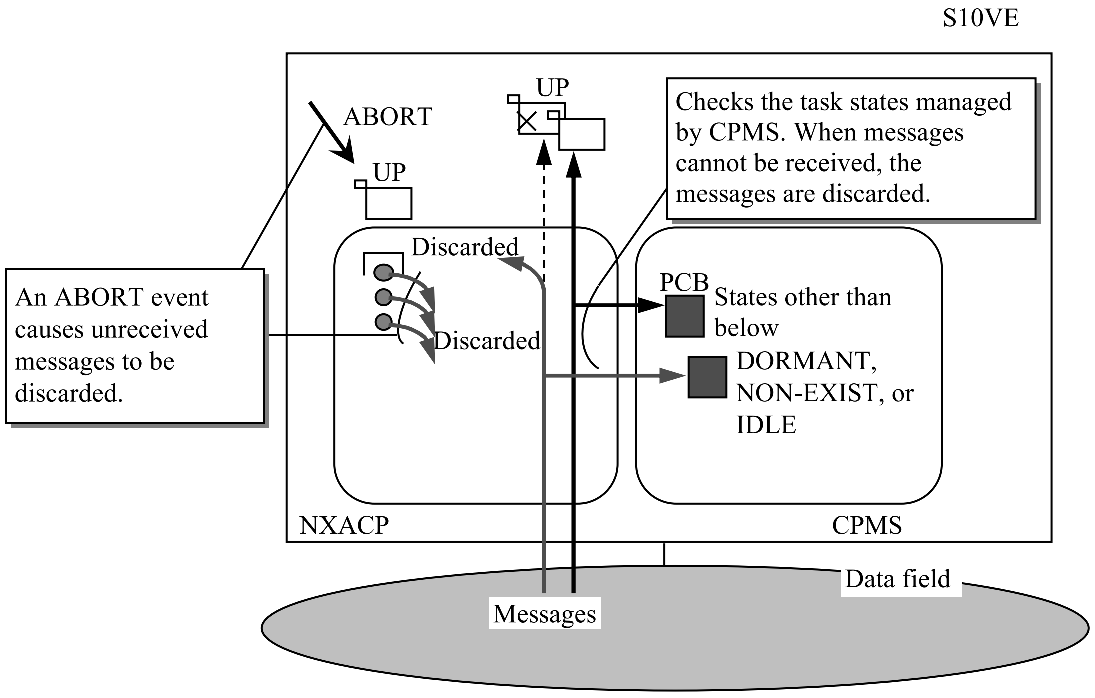

S10VE
ABORT
UP
UP
Checks the task states managed by CPMS. When messages cannot be received, the messages are discarded.
An ABORT event causes unreceived messages to be discarded.
Discarded
Discarded
PCB
States other than below
DORMANT, NON-EXIST, or IDLE
NXACP
CPMS
Data field
Messages

Figure 1-26 UP Task Management

3.6 Operation Management Function

NXACP services can be started and stopped at users' discretion.
NXACP is started and stopped by issuing nx_init() and nx_quit() respectively.
Apart from starting and stopping NXACP, you can start and stop a data field service by issuing nx_dfup() and nx_dfdwn() respectively.

(1) Starting NXACP
A user must issue nx_init() when the system starts. When nx_init() is issued, NXACP prepares to accept a service start request for each data field.

(2) Starting a data field service
To request a data field to start sending and receiving messages, issue nx_dfup() to the data field. Note that you must also issue nx_dfup() to a data field dedicated to local node communication, which does not use any networks.
If you define multiple data fields, you must issue nx_dfup() for each data field, which means you must issue nx_dfup() multiple times. Even if a data field is constructed on duplexed LANs, issue nx_dfup() once to the data field.

(3) Stopping a data field service
If you want only specific data fields to stop sending and receiving messages while the system is kept running, issue nx_dfdwn() to each of the data fields. By using this function, you can change node modes without stopping the system. If you want a data field dedicated to local node communication to stop sending and receiving messages, issue nx_dfdwn() to the data field.
When nx_dfdwn() is issued to a local data field, nx_dfdwn() notifies that this is a scheduled stop (SHUTDOWN mode alive signal) to other nodes, and then stops the send/receive service.
If you want to restart a data field service that you stopped by using nx_dfdwn(), reissue nx_dfup() to the data field.

(4) Stopping NXACP
If you want to stop NXACP or all data fields at once while the system is kept running, issue nx_quit(). Similar to nx_dfdwn(), when nx_quit() is issued, nx_quit() notifies that this is a scheduled stop to other nodes, and then stops the send/receive service.
If you want to restart NXACP that you stopped by using nx_quit(), issue nx_init() to NXACP.

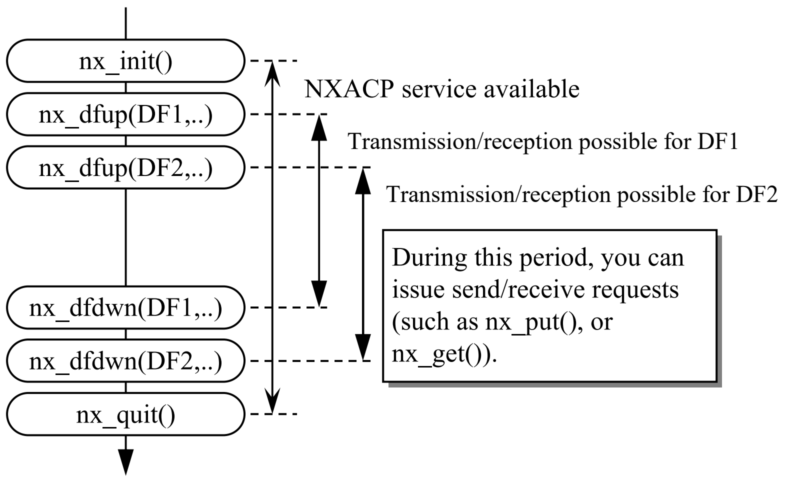

nx_init()
nx_dfup(DF1,..)
nx_dfup(DF2,..)
nx_dfdwn(DF1,..)
nx_dfdwn(DF2,..)
nx_quit()
NXACP service available
Transmission/reception possible for DF1
Transmission/reception possible for DF2
During this period, you can issue send/receive requests (such as nx_put(), or nx_get()).

Figure 1-27 NXACP Operation Management

3.7 Network-Shared Memory Function

Network-shared memory connects the nodes that belong to a data field over the network using a memory image. Data written by each node is transferred at a constant interval to reading nodes.
The same memory address can be written by only one node but read by multiple nodes (after data is transferred).

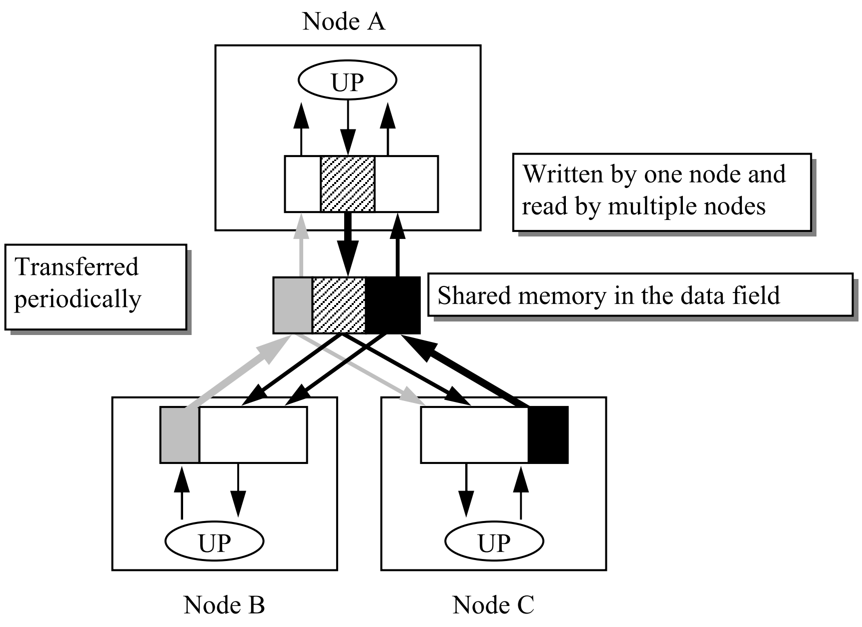

Node A
UP
Written by one node and read by multiple nodes
Transferred periodically
Shared memory in the data field
UP
UP
Node B
Node C

Figure 1-28 Network-Shared Memory

The software supports transfer intervals in milliseconds.

Network-shared memory supported by NXACP has the following characteristics.
- Encapsulation of the communication procedure
  A user can communicate using a memory interface.
- A more economical solution using software transfer
  Transfer memory over Ethernet using software is supported. This makes it easier for a user to set up memory-interface-based communication without purchasing special hardware.

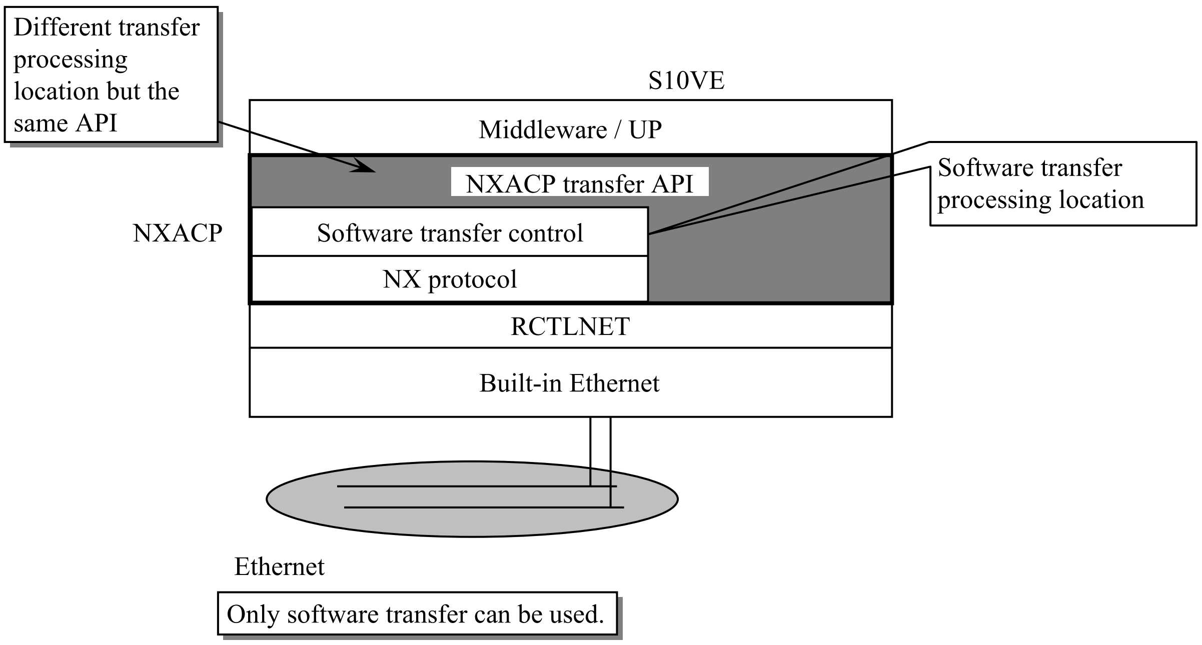

Different transfer processing location but the same API
S10VE
Middleware / UP
NXACP transfer API
NXACP
Software transfer control
NX protocol
Software transfer processing location
RCTLNET
Built-in Ethernet
Ethernet
Only software transfer can be used.

Figure 1-29 Characteristics of Shared Memory

[Note]
The network-shared memory function of NXACP does not have a ladder interface, unlike other functions of NXACP.

3.8 System Construction Function

To set up NXACP, define its construction information on POC, and then load the construction information to the target RPDP management site information by using the NXACP construction command.
To perform remote loading to S10VE, run the svrpl (remote loading) command of RPDP.

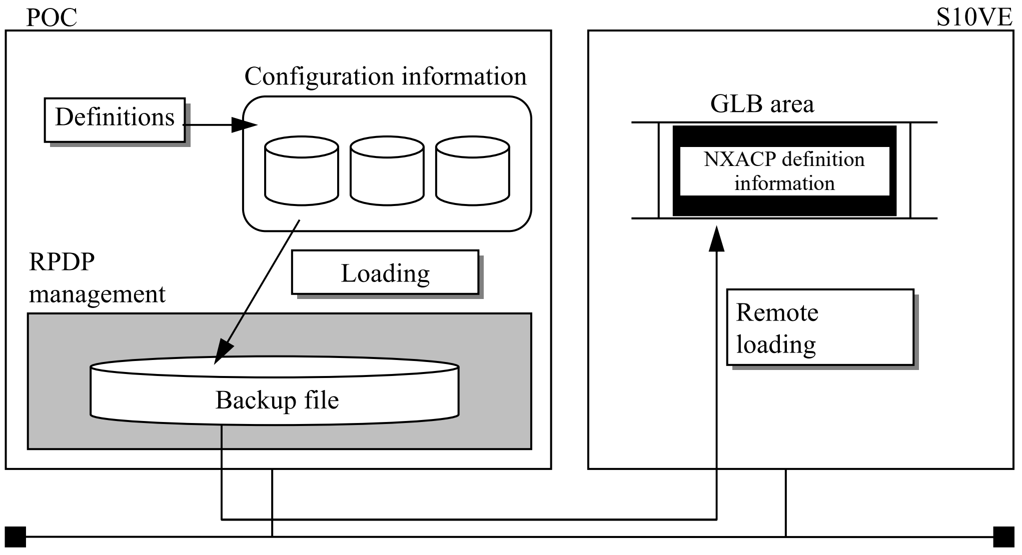


Figure 1-30 System Construction

In the S10VE, a multiprocessor configuration type (function distribution type) site can be configured by RPDP generation.

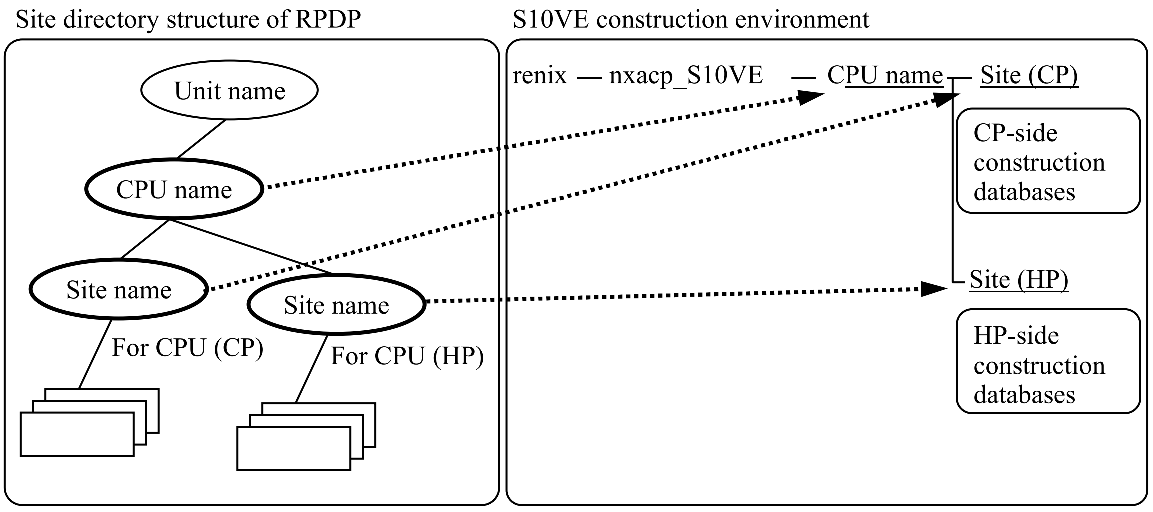

Site directory structure of RPDP
S10VE construction environment
Unit name
CPU name
Site name
Site name
For CPU (CP)
For CPU (HP)
renix — nxacp_S10VE — CPU name — Site (CP)
CP-side construction databases
Site (HP)
HP-side construction databases

Figure 1-31 NXACP Configuration Environment

The construction databases include the following construction information.

- Buffer information such as the number and size of cases in the buffer
- Alive signal information such as the alive signal transmission interval and whether to receive alive signals
- Network signal such as network address and port number
- Transaction information such as TCD, MGN, and send/receive task number

Define the information listed above in a database file, compile the database file, and then load it to a target site by using the NXACP construction information loading command. Since the NXACP construction information loading command uses an RPDP command to load the information to the GLB area, you must use the loading command after system generation is complete.
Changing construction information online is not supported. When updating construction information to a target machine, the construction loading command temporarily stops the NXACP of the target machine.
Note that, during this period, an alive signal for scheduled SHUTDOWN is not sent.

This Page Intentionally Left Blank

PART 2   FUNCTION GUIDE

CHAPTER 1 MULTICAST COMMUNICATION FUNCTION

1.1 Characteristics of Communication

The following explains the communication methods available for this system.

1.1.1 Basic unit of message communication

In this manual, a chunk of information input and output by a user is called "data".
One piece of logical information handled in NXACP and created by adding NX headers to the data specified by a user is called a "message". One information unit that is transmitted over the network is called a "packet".
NXACP adds NX headers to the data specified by a user to form a message, and then sends the message over the network as multiple packets. The NXACP on the receiver reassembles the multiple packets into a message, deletes the NX header, and then passes the data to a user.
The maximum length of data a user can specify is 16 KB.

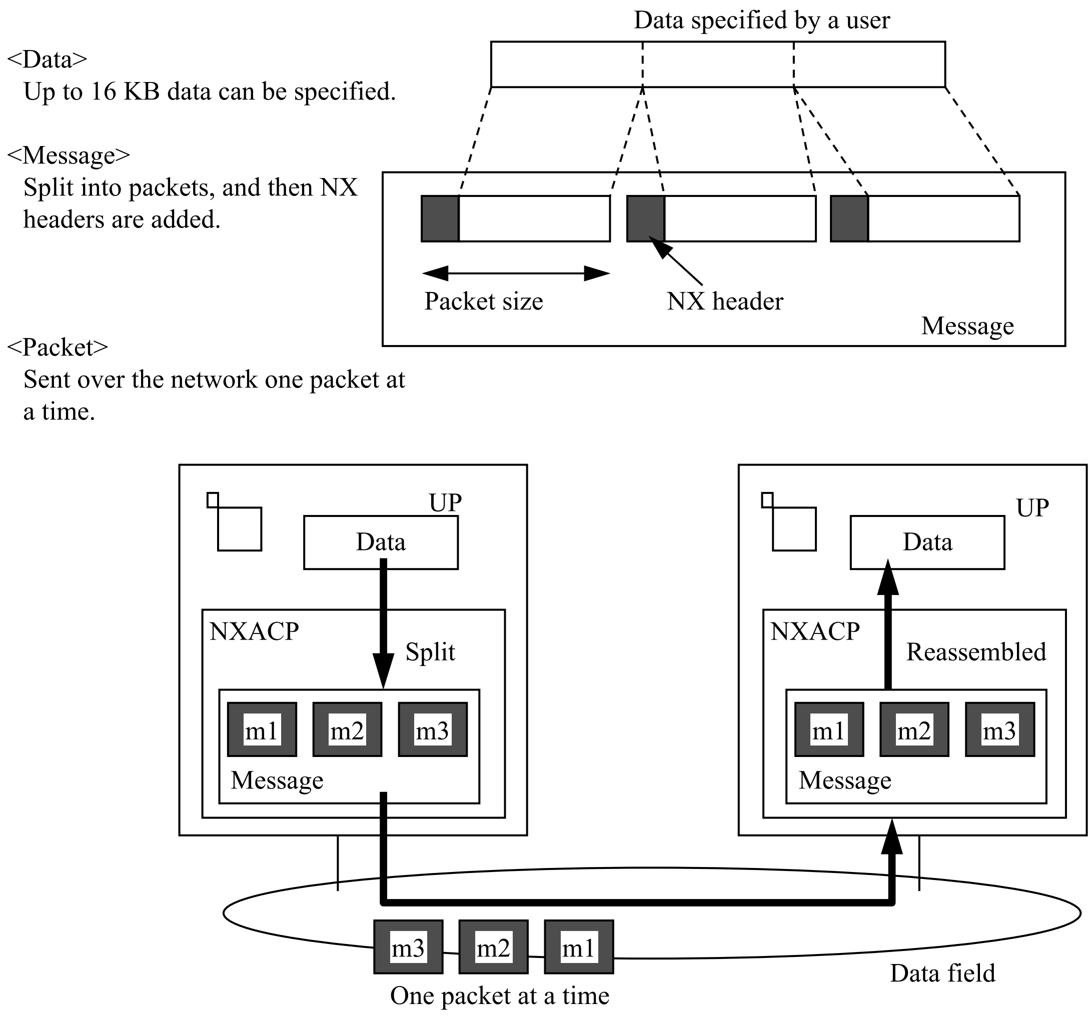


Figure 2-1 Unit of Communication

1.1.2 Splitting and reassembling a message

The maximum length of sending data a user can specify is 16 KB. NXACP splits a message into suitably-sized packets which are then reassembled at a receiver.
When a message is split, packet-size is calculated by adding the NX header size (64 bytes) to the size of one case of buffer specified for each data field by a user in the construction information.
The maximum size of one case of buffer you can specify is the same regardless of the network type, as shown below.

Table 2-1 Buffer Size

Network type	Maximum size of one case of buffer	Packet size
Ethernet (Built-in Ethernet)	1408 bytes	1472

The NX header size is fixed at 64 bytes. Therefore, the length of user data in one packet is the maximum packet size minus 64 bytes.
The following formula shows the relationship between the size of sending data specified by a user and the number of cases used by packets.

$$\text{Number of cases used by packets} = \frac{\text{(Data size specified by a user - 1)}}{\text{Size of one case of buffer}} + 1$$

For example, if the maximum size of one case of buffer is selected in the construction information and transmission of 16 KB is requested, the number of cases used by packets is calculated as follows.

$$\text{Number of cases used by packets} = \frac{(16384 - 1)}{1408} + 1 = 11 + 1 = 12 \text{ cases}$$

[Note]
For any given data field, the size of one case of buffer must be the same for all nodes.
When NXACP receives a packet larger than the size of one case of buffer, failure information is notified to a user, and the received packet is discarded.

1.1.3 Structure of a user program

When a user requests data to be sent or received, the user must issue the nx_put() macro or the nx_get() macro respectively.
The nx_put() and nx_get() macros have the following characteristics. Note that these macros can be issued by a task only when the task number of the task is between 1 and 208.

(1) Transmission request
When a user requests data to be sent, the user must issue the nx_put() macro.
In NXACP, the service for accepting a transmission request from a user and the service for sending packets to the network are independent from each other. This allows the transmission process in a user program to be executed asynchronously from hardware processing. In addition, the service for sending packets to the network is woken up at a fixed interval (defined in construction information) to reduce the number of times tasks are switched.
Therefore, even when nx_put() finishes normally, it is not guaranteed that data has been transmitted to the network normally. The information the nx_put() macro returns to a user program in response to a transmission request (nx_put()) from the user program is limited to the following information. Note that the transmission process is aborted when either one of the following failures is detected.

- Error code for detecting an error in the operation environment check
- Error code for detecting a parameter error
- Error code for NXACP send buffer full
- Error code generated after a network failure is detected

When an error is detected during execution of the service for sending packets to the network, to notify the error to a user, the network driver (not NXACP) notifies the error to EAS. For details about failures reported to EAS, refer to the *S10VE User's Manual General Description* (manual number SEE-1-001).

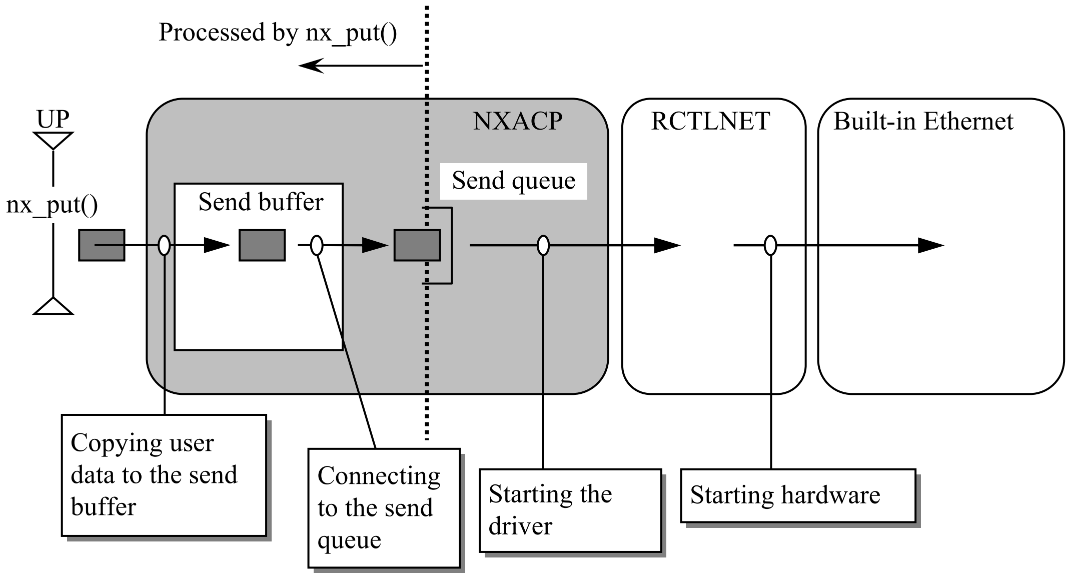


Figure 2-2 nx_put Macro

Depending on the location of failure detection, a user might need to detect a failure using EAS instead of a return code from nx_put(). In this case, the user must monitor reports of failures detected by RCTLNET and failures detected by NXACP.

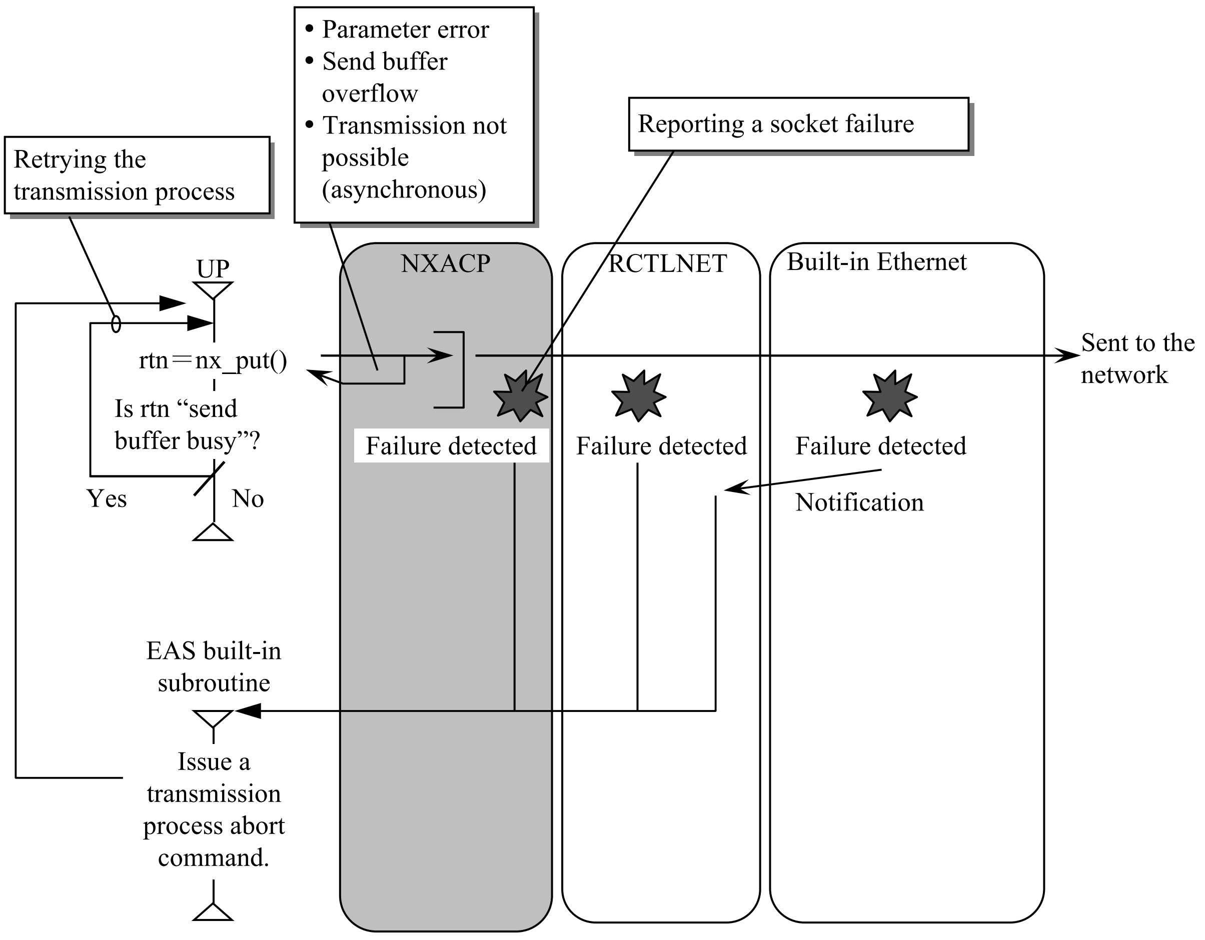


Figure 2-3 Detecting Failure Information (Transmission)

[Note]

- About a socket failure
  The details of failures generated in hardware such as Built-in Ethernet are reported by RCTLNET to EAS.
  Failures reported by NXACP to EAS result from incorrect network construction, and are called "socket failures". For details about failures reported by NXACP to EAS, see "5.1 Failure Notification Function".
- About "transmission not possible"
  If the network transmission process cannot continue due to a failure caused by hardware (such as Built-in Ethernet), nx_put() returns the error "transmission not possible" if a request is made after the failure. At the same time, all messages that were requested before failing to be sent by nx_put() are discarded and kept in the send queue, to await network transmission service.

(2) Reception request

A user that requests data to be received must issue the nx_get() macro.
The reception process is similar to the transmission process in that the service for accepting a reception request from a user and the service for interrupts from the network are independent from each other. This allows the reception process in a user program to be executed asynchronously from the handing of interrupts from hardware.
The information the nx_get() macro returns to a user program (in response to a reception request) is limited to the cases below. Note also that, in these cases, the reception process is always aborted.

- Error code for detecting an error in the operation environment check
- Error code for detecting a parameter error
- Error code for no message arrived within the predefined time (specified as a parameter by UP)
- Error code for detecting NXACP has stopped

Failures generated during the message reception process in RCTLNET or hardware such as Built-in Ethernet are reported by RCTLNET to EAS. Note that, even if failures are generated in RCTLNET or hardware such as Built-in Ethernet, nx_get() does not return an error.

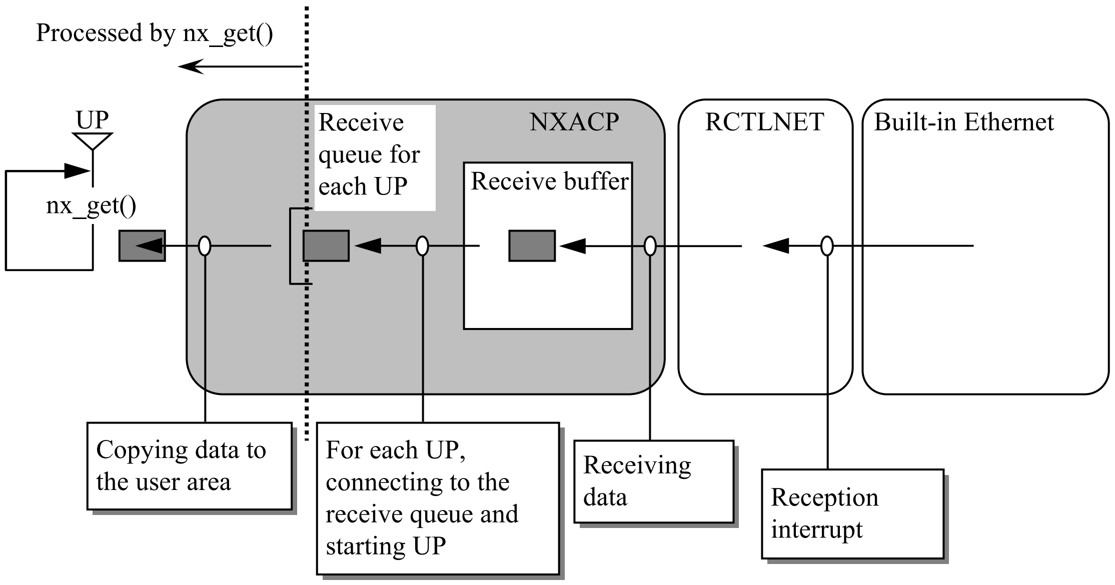

Processed by nx_get()
UP
nx_get()
Receive queue for each UP
NXACP
Receive buffer
RCTLNET
Built-in Ethernet
Copying data to the user area
For each UP, connecting to the receive queue and starting UP
Receiving data
Reception interrupt

Figure 2-4 nx_get Macro

It is recommended that a user task for issuing nx_get() is constructed as a loop and issues a transaction reception request at the beginning of the loop. Compared to using exit() and initial start for a user task, there is less overhead if a user task waits in nx_get() for a receiving message to arrive, and has NXACP wake up the task when a message arrives.

Depending on the location of failure detection, a user may need to detect a failure in a timely manner using EAS instead of detecting a failure based on a return code from nx_get(). In this case, the user must monitor the report of failures detected by RCTLNET, in addition to failures detected by NXACP.

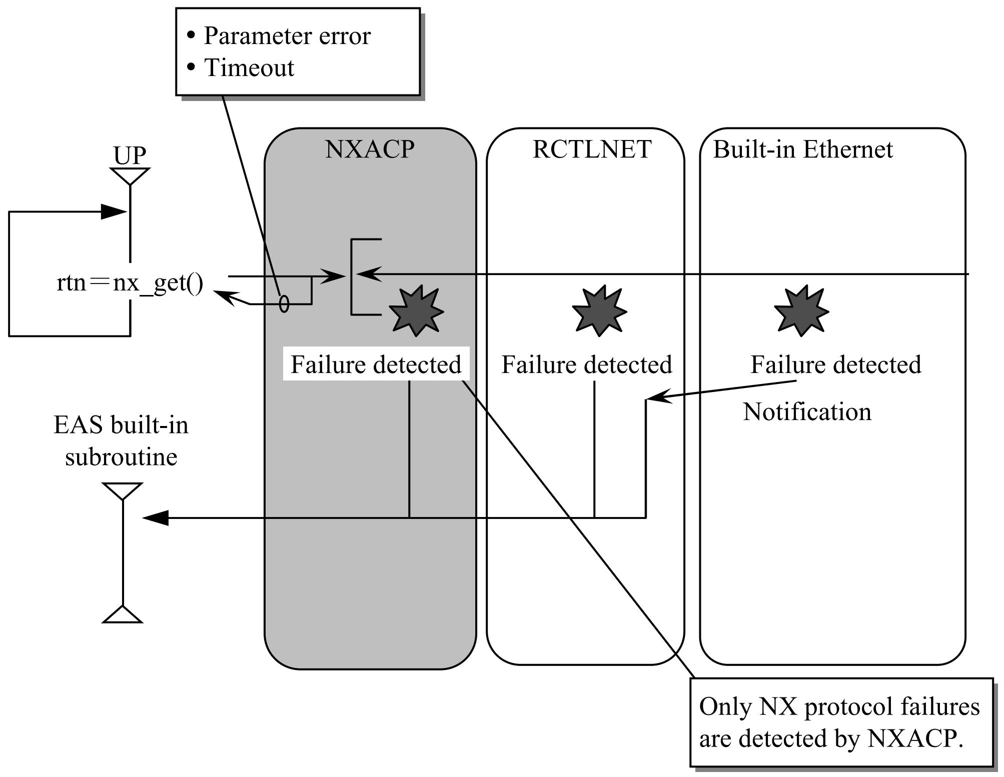


Figure 2-5 Detecting Failure Information (Reception)

1.1.4 Message Processing Order

In the message transmission and reception process of NXACP, messages are serviced in order of message generation (FCFS) for both transmission and reception. Therefore, sending messages are serviced in order of transmission requests from a user (if task levels are the same), and receiving messages are serviced in order of arrival for each multicast group.

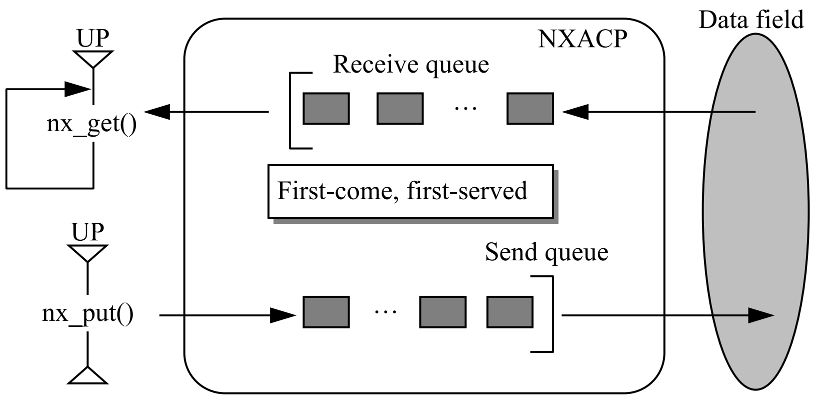


Figure 2-6 Processing Order

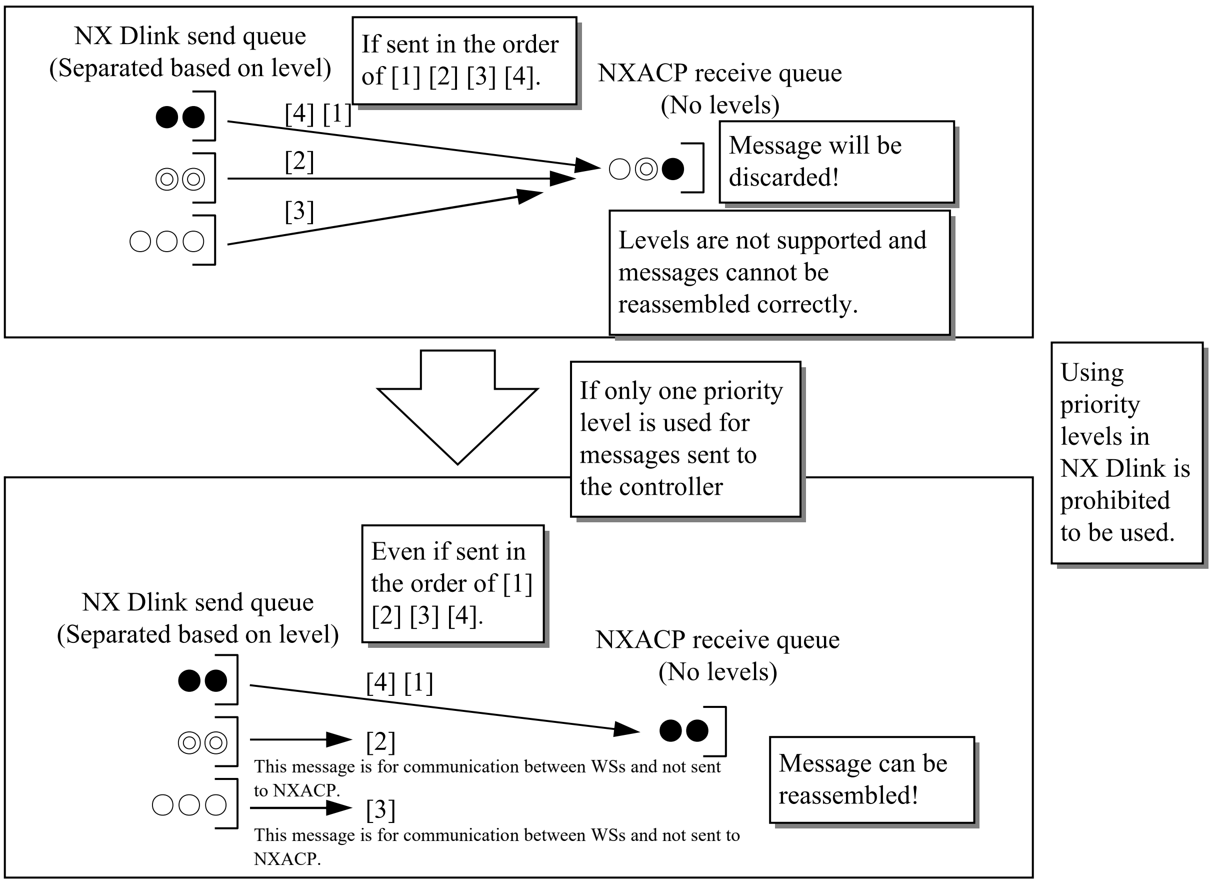
[Note]

NXACP does not support message priority control.
NX protocol supports “message priority levels”. This feature allows you to prioritize messages processed during transmission and reception. Messages with a higher priority will have precedence when serviced.
If you use this feature, you can, for example, execute file transfer when the processing of online messages is not busy, by assigning higher priorities to online messages and lower priorities to file transfer data. This feature is not supported because controllers do not handle data of the file transfer type.
NXACP always specifies 0 (zero) for the transmission level in the header when a message is sent to the network.
If the transmission level of a message is 0 (zero), you can assign any priority level (for reception) to the message in an NX family that supports assigning priority levels (for example, NX Dlink). For details, refer to the manual of the NX support package for each device.
Note that when messages are received from the network, the priority level in a header is not evaluated, and messages are processed in order of arrival.

[Restriction]

When NX Dlink is connected, note that the following restrictions apply to message priority levels set by NX Dlink.

When NX Dlink sends long messages (a long message is a message that must be split into and reassembled from two or more packets), the same priority level must always be used.


NX Dlink send queue
(Separated based on level)
If sent in the order of [1] [2] [3] [4].
NXACP receive queue
(No levels)
[4] [1]
[2]
[3]
Message will be discarded!
Levels are not supported and messages cannot be reassembled correctly.
If only one priority level is used for messages sent to the controller
Using priority levels in NX Dlink is prohibited to be used.
Even if sent in the order of [1] [2] [3] [4].
NX Dlink send queue
(Separated based on level)
NXACP receive queue
(No levels)
[4] [1]
[2]
This message is for communication between WSs and not sent to NXACP.
[3]
This message is for communication between WSs and not sent to NXACP.
Message can be reassembled!

●: One message consists of two packets.
◎: One message consists of two packets.
○: One message consists of three packets.

Figure 2-7 Restrictions on Using Priority Levels

1.2 Message Transmission Function

This section describes the transmission function of the nx_put() macro.
When a user task issues nx_put(), you must specify “data field number (DFN)” as the destination of the transmission.
If you specify a non-zero value for the data field number, the message is sent to the network.
Specifying 0 for the data field number has a special meaning: communication within the local controller.

- Network transmission (if a non-zero value is specified for DF)
  If the sender and the receiver are on different nodes, the transmission is called a network transmission.
- Local node transmission (if 0 is specified for DF)
  If the sender and the receiver are on the same node, the transmission is called a local node transmission.

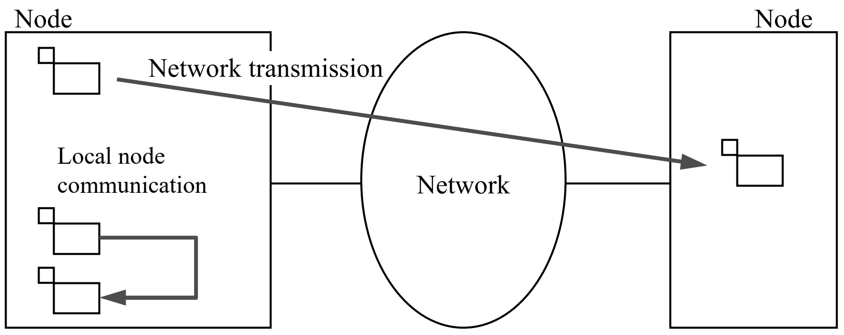
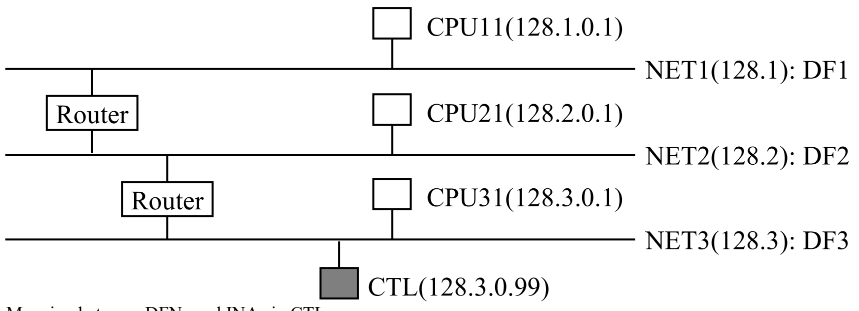


Figure 2-8 Data Field Number and Transmission Area

1.2.1 Network Transmission

(1) Specifying a Transmission Area

To send a message to another node connected over the network, you must specify a non-zero value for the data field number when you issue a transmission request.
Before transmission, you must define the mapping between data field numbers and network addresses (INA) in the construction information. Data field numbers are logically mapped to the addresses of the network segments you want to send messages to; therefore, specify a data field number instead of a network address when you send a message. In the following example, if you want to send a message from CTL to the network where CPU11 is connected, specify 1 for the data field number when you issue nx_put().



Mapping between DFNs and INAs in CTL

DFN	INA	Name
1	128.1	NET1
2	128.2	NET2
3	128.3	NET3

In NXACP, network segments and data fields are mapped 1-to-1.

Figure 2-9 Data Field Numbers and Network Segments

In addition to a data field number, you must specify a multicast group number (MGN) and a transaction code (TCD).
In addition, before transmission, multicast group numbers also must be mapped to port numbers in the construction information for both online and test modes.

Table 2-2 Multicast Group Numbers and Ports

MGN	Online mode destination port number	Test mode destination port number
1	55001	57001
2	55002	57002
3	55003	57003
4	55004	57004

One multicast group is mapped to two ports, one for each mode.

NXACP broadcasts a message to the network segment specified by the data field number. The destination port number of the message is the port number mapped from the specified multicast group number.
Therefore, the only nodes that can receive the sent message are those for which the port mapped from the specified multicast group number is defined (as a receive multicast group). In nodes where ports are not defined, messages are discarded at the OS level (in the case of Built-in Ethernet).

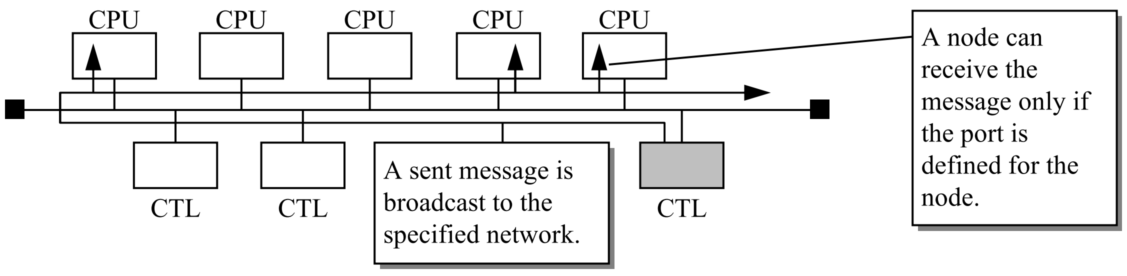

CPU
CPU
CPU
CPU
CPU
A node can receive the message only if the port is defined for the node.
CTL
CTL
A sent message is broadcast to the specified network.
CTL

Figure 2-10 Multicast and Selective Reception

In short, by specifying a multicast group number, the message is sent to only those nodes connected to the data field, and to which the message is targeted. Mapping between multicast group numbers and port numbers must be uniquely defined for all nodes connected to the data field.
A transaction code is an ID used for determining the destinations of a message (user tasks, in the case of NXACP) in the receiver node. For information about the relationship between transaction codes and user tasks in NXACP, see “1.3 Message Reception Function”.

(2) Transmission to a remote data field

In a case where you send a message to a remote data field connected by way of a router, you must specify a data field number when you issue a transmission request for the message. Only one path can be specified for the transmission path to the remote data field. If you want to duplex the path, use the duplexed LAN control function (described later).

In Figure 2-11, there are two paths you can use for sending a message from Node A to Node B: a path by way of Data field 3, and a path by way of Data field 2. NXACP selects one of the paths statically as specified by a user (in the construction information). For this reason, you cannot specify multiple paths, and if an error occurs, you cannot switch paths by detecting the error.

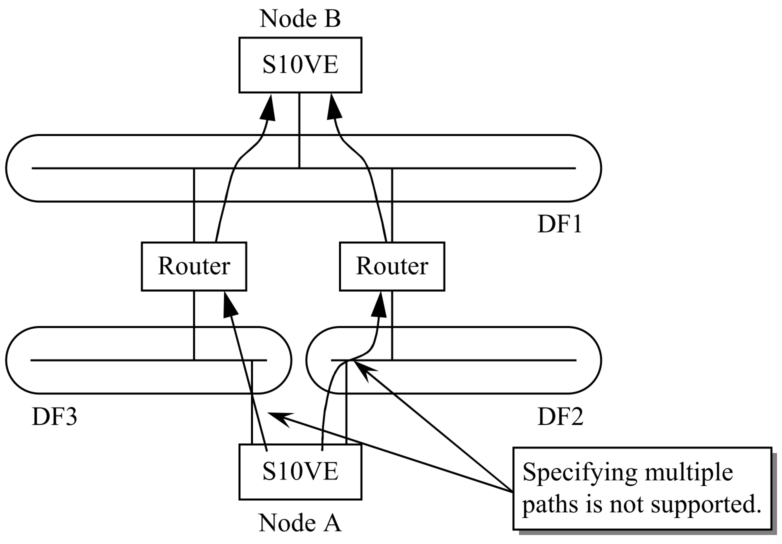


Figure 2-11 Remote Data Field (1)

In addition, the following configuration and transmission/reception method are not allowed. That is, a remote data field must be connected to only one local data field.

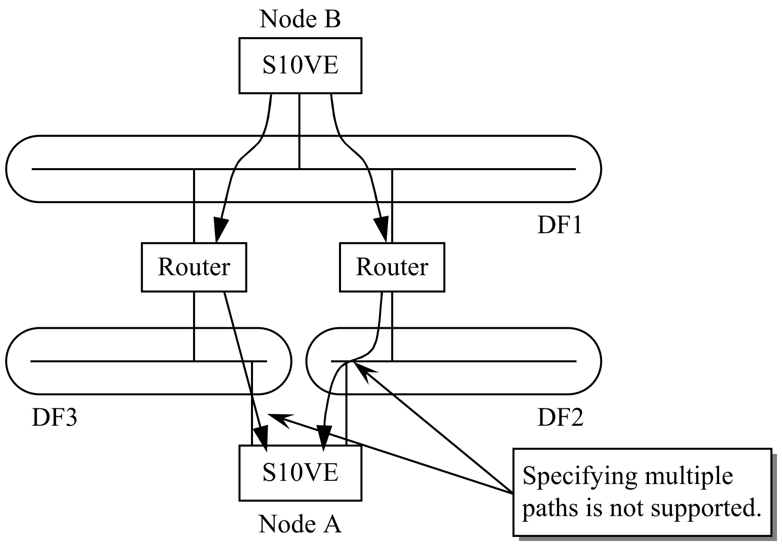


Figure 2-12 Remote Data Field (2)

A remote data field must correspond to only one local data field and be assigned only one path.
If you want to improve the reliability of communication, we recommend duplexing the router and the network.

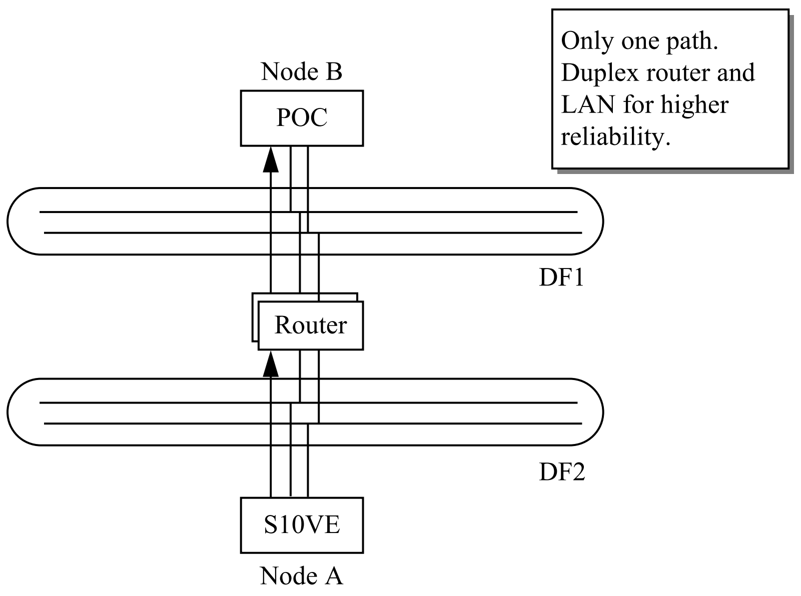


Figure 2-13 Remote Data Field (3)

[Caution] Cautions to take when using a remote data field
When designing a system that uses a remote data field, carefully consider network traffic over remote/local data fields, and the characteristics of the router to be used.

(3) Sending back to the local node
If you issue a transmission request of a message to a non-zero data field number, the message will not be sent to the local node. If the multicast group number of the sent message is set to the received multicast number for the local node, the message is discarded by NXACP. In the reception process of NXACP, the sender node number is checked. If the sender node is the local node, the message is discarded.
In short, a message transmission request for a non-zero data field number is dedicated only to sending a message to other nodes.

(4) Send buffer management
The send buffer is managed on a per-data-field basis, meaning that when the send buffer for one data field is busy, other data fields are not affected.
In constructing a system, specify the number of cases in the buffer for each data field.

(5) Transmission permission check
If you intend to allow user tasks to send messages by using NXACP, we recommend that, at system construction, you define which transaction codes are to be sent by each task. Based on this definition, nx_put() checks whether the sender user task matches the transmission transaction code, and if the sender user task tries to send an undefined transaction code, the process is aborted as an error.
Using this function is optional, and the above definition is not mandatory. But it is recommended that correspondence between transaction codes and sender user tasks is defined and managed in the construction information.
The maximum number of transaction codes one user task can send per data field is 8 TCDs.

Table 2-3 Correspondence between Transaction Codes and User Tasks

TCD	Sender TN1	Sender TN2	Sender TN3	···	Sender TN8
33	1	41	25		111
48	41	–	–		–
76	37	112	55		–
99	25	39	40		41

The definitions for user tasks and transmission transaction codes must be defined on a per-data-field basis. If transmission permission check is not necessary, you can skip these definitions.
You can select whether to enable transmission permission check on a per-user-task basis. If transmission permission check is enabled for a user task, define transmission transaction codes for the user task in all the construction files in the destination data field.

(6) Sending to a specific node

NXACP supports only multicast transmission: you cannot specify a destination node for a transmission. However, by assigning a multicast group number or transaction code to a specific logical node, you can send a message to the node.

This can be done in either of the following ways.

(a) Assigning a multicast number to a destination node number

By assigning a multicast group to a specific destination node, you can facilitate transmission to that node.

- Dedicate the receive port number of one multicast group to each node. In the case of duplexed CPUs, assign one multicast group to two nodes.
- All nodes in a data field must have their own dedicated multicast group number.

Table 2-4 Correspondence between Multicast Groups and Nodes

MGN	Destination port number	Corresponding node number
1	55001	A
2	55002	B
3	55003	C
4	55004	D, E

Unique across all nodes

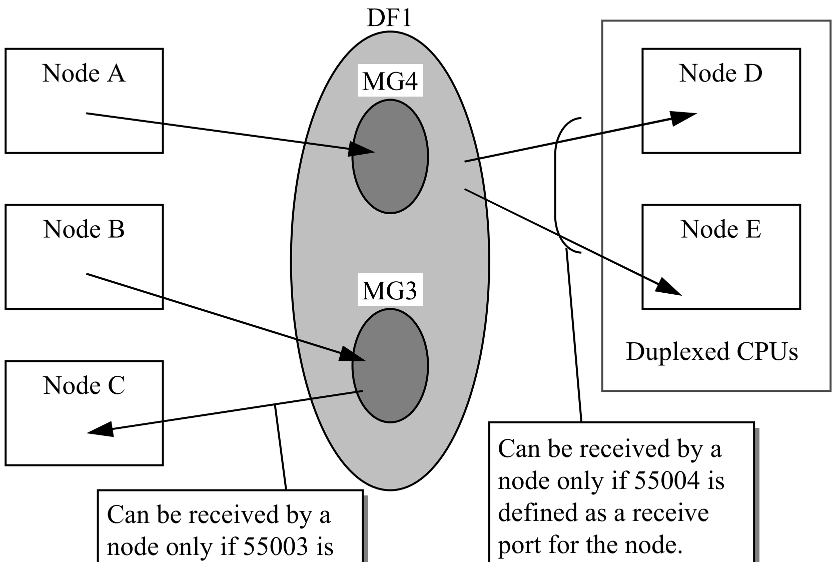

DF1
Node A
Node B
Node C
MG4
MG3
Node D
Node E
Duplexed CPUs
Can be received by a node only if 55003 is defined as a receive port for the node.
Can be received by a node only if 55004 is defined as a receive port for the node.

Figure 2-14 Transmission to a Specific Node (1)

In NXACP, you can define, per data field, up to 128 send multicast groups and up to 32 receive multicast groups.

(b) Assigning a transaction code to a destination node number

By assigning a transaction code to a destination node, you can facilitate transmission to a specific node.

- Dedicate one specific transaction code to each node respectively. In the case of duplexed CPUs, assign one transaction code to two nodes.
- All nodes in a data field must have their own dedicated transaction code.

Table 2-5 Correspondence between Transaction Codes and Nodes

MGN	TCD	Corresponding node number
1	1 to 10	A
1	11 to 20	B
1	21 to 30	C
2	31 to 40	D, E

Unique across all nodes

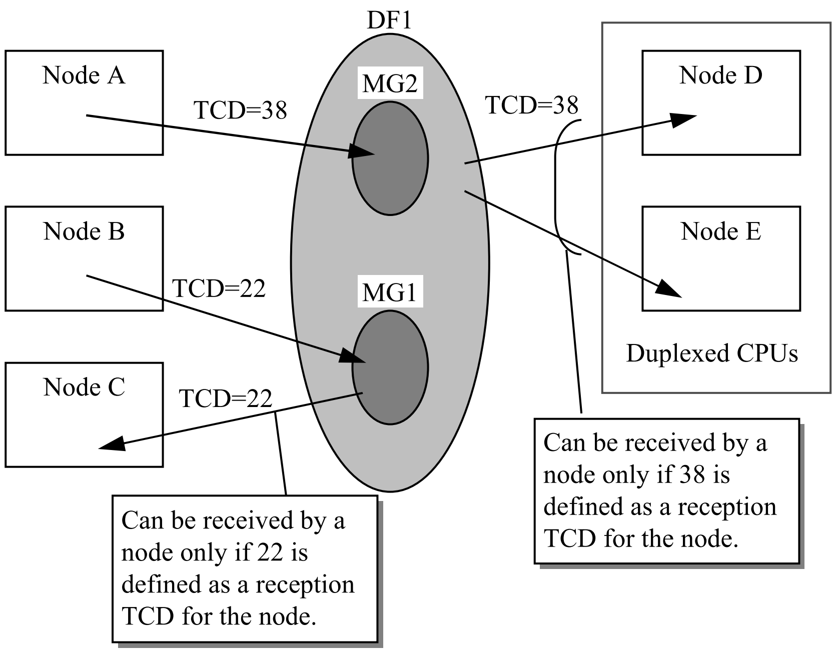

DF1
Node A
TCD=38
MG2
TCD=38
Node D
Node E
Duplexed CPUs
Node B
TCD=22
MG1
Node C
TCD=22
Can be received by a node only if 22 is defined as a reception TCD for the node.
Can be received by a node only if 38 is defined as a reception TCD for the node.

Figure 2-15 Transmission to a Specific Node (2)

When methods (a) and (b) are compared, unnecessary messages are discarded by hardware or at the OS level in the case of (a), or by NXACP on a CPU in the case of (b). We recommend using method (a) to minimize CPU load.

1.2.2 Local node transmission

Data field # 0 is dedicated to local node transmission.
If you specify 0 for the data field number and send a message, the message is passed using the buffer dedicated to communication inside the local node. This data field cannot send or receive messages to or from external nodes.
When you specify 0 for the data field number and send a message, specify 0 for the multicast group number because the multicast group number is meaningless. Even if a user specifies a non-zero value for the multicast group number, NXACP interprets the multicast group number as zero.
Transaction codes used here are the ones for the data field dedicated to the local node transmission. This is because transaction codes are uniquely defined for each data field. Nevertheless, we recommend that assignment of functions to transaction codes is uniquely defined for the whole network because messages sent within the local node and messages sent from other nodes can be received using the same procedure by a receiver task.

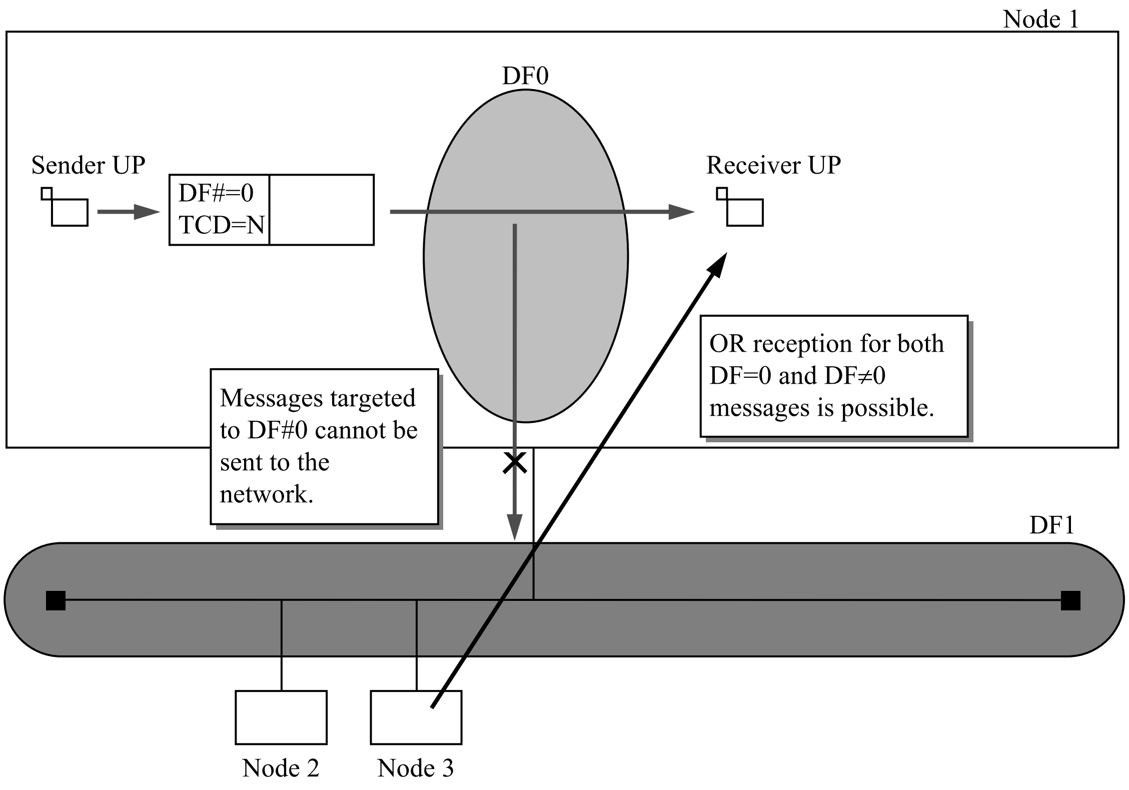

Node 1
DF0
Sender UP
DF#=0
TCD=N
Receiver UP
OR reception for both DF=0 and DF≠0 messages is possible.
Messages targeted to DF#0 cannot be sent to the network.
DF1
Node 2
Node 3

Figure 2-16 Local Node Communication

Send buffers are managed on a per-data-field basis, and consequently, local node communication has a send buffer dedicated to data field #0. In constructing a system, specify the number of cases in the buffer in the same way as other data fields. Even though the size of one case of buffer is also managed on a per-data-field basis (and you can specify a different value for the local node communication than for other data fields), we recommend that you use the same size as other data fields.
Local node communication uses one buffer as both send and receive buffers (managed as a receive buffer).
You do not have to set up both send and receive buffers independently. When you define construction information, define only a receive buffer.

1.3 Message Reception Function

This section describes the reception function of the nx_get() macro.
When a user task issues nx_get(), you must define transaction codes for each data field that receives messages. This definition information is necessary for each data field to receive messages from NXACP. In this way, a user task that issues nx_get() can receive messages in the same way for local node communication or inter-node communication.

1.3.1 Message reception

The data reception function of nx_get() has the following characteristics.

- Multiple data fields and transaction codes can be defined to be received by one user task. Note that only one message can be received by one reception request. Also note that you can check the control information of a received message, such as the sender address.
- One message can be shared and received by multiple user tasks. By using this feature, you can pack multiple business data items into one message to improve the efficiency of communication.
- When you receive a message, you can specify the offset and size of the data to be received.
  By using this feature, each user task can retrieve only necessary information when multiple business data items are packed in one message.

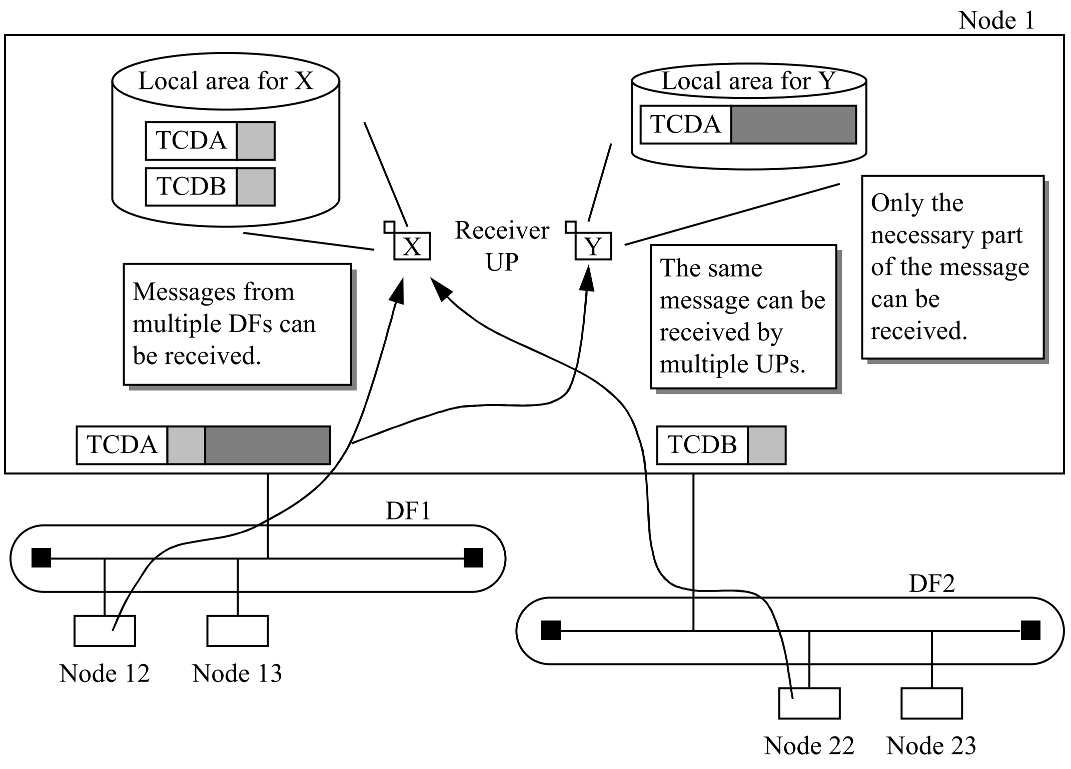

Node 1
Local area for X
TCDA
TCDB
Local area for Y
TCDA
X
Receiver UP
Y
Messages from multiple DFs can be received.
The same message can be received by multiple UPs.
Only the necessary part of the message can be received.
TCDA
TCDB
DF1
DF2
Node 12
Node 13
Node 22
Node 23

Figure 2-17 Characteristics of nx_get()

(1) Definition information and return information

To allow user tasks to receive messages by using NXACP, you must define the data field numbers and transaction codes received by each task at system construction.
The reason why this information is not included in the parameters of nx_get() is to make user tasks and the reception management information independent, and to improve user task portability.
Per data field, the maximum number of transaction codes one user task can receive is 8. The maximum number of tasks that can receive the same TCD is 8. In addition, multiple data field numbers can be defined to be received by one user task. Note that only one message can be received each time nx_get() is issued.
Received messages are processed on a first-come-first-served basis, and then passed to user tasks. For all data fields and transaction codes defined at system construction, which data field and transaction code apply to a received message is notified to a user in return information.

Table 2-6 Correspondence between Transaction Codes and User Tasks

TCD	Receiver TN1	Receiver TN2	Receiver TN3	⋯	Receiver TN8
33	1	41	25		111
48	41	–	–		–
76	37	112	55		–
99	25	39	40		41

Return information of the nx_get()

```
typedef  struct    {
        unsigned  char   dfn;      /* Sender data field number          */
        char   fu1;                /* For future use                    */
        unsigned  short   mgn;     /* Multicast group number            */
        unsigned  short   tcd;     /* Transaction code                  */
        short   fu2;               /* For future use                    */
        unsignid  long   sa;       /* Sender address                    */
        unsignid  long   rcvlen;   /* Receive message size              */
}nx_ginfo;
```

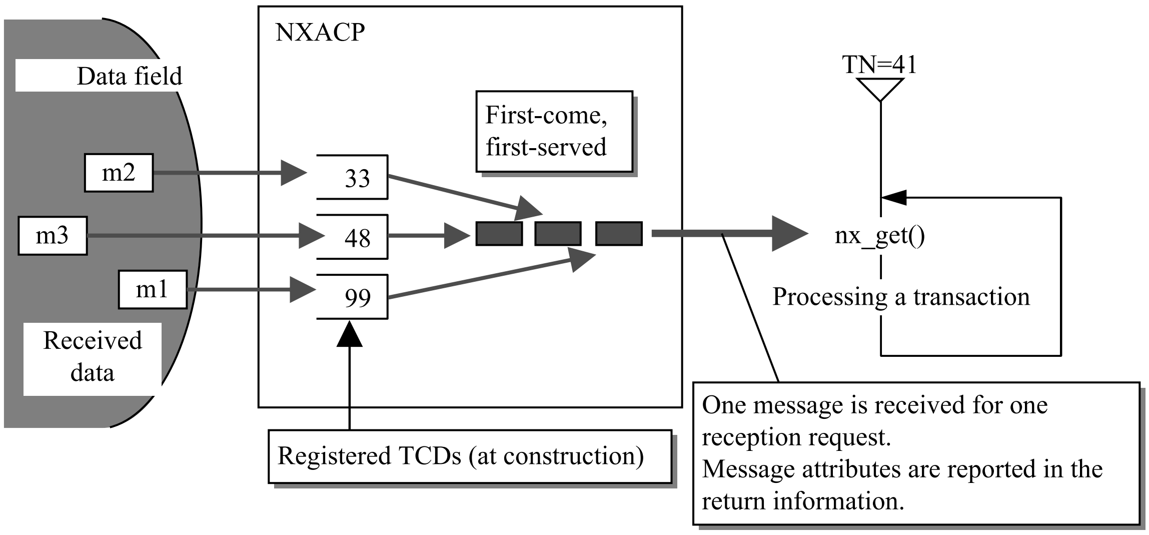

NXACP
Data field
m2
m3
m1
Received data
33
48
99
First-come, first-served
TN=41
nx_get()
Processing a transaction
Registered TCDs (at construction)
One message is received for one reception request.
Message attributes are reported in the return information.

Figure 2-18 Reception Process Flow

(2) Shared transaction code reception
One transaction code can be shared by multiple user tasks, and messages with this transaction code can be received by these user tasks. By using this feature, you can pack the data of multiple business messages into one message to improve the efficiency of network communication. You can also reduce the number of interrupts from the network to reduce the CPU load.
At the user task side, multiple business messages can be processed by each business task in parallel.

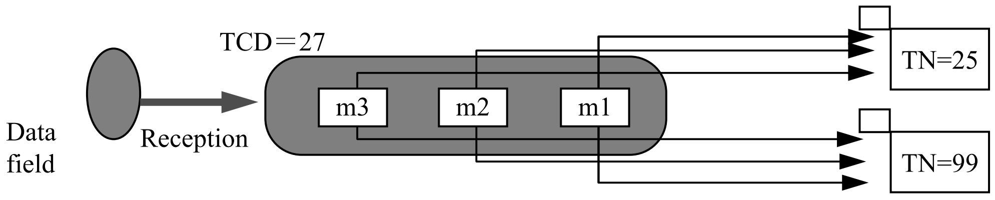


Figure 2-19 Shared Reception

Each user task that shares the same transaction code receives only the necessary part of the data. By using this feature, you can minimize memory usage and the time required for copying data.
Which part of the data is received must be specified in parameters of the nx_get() macro.

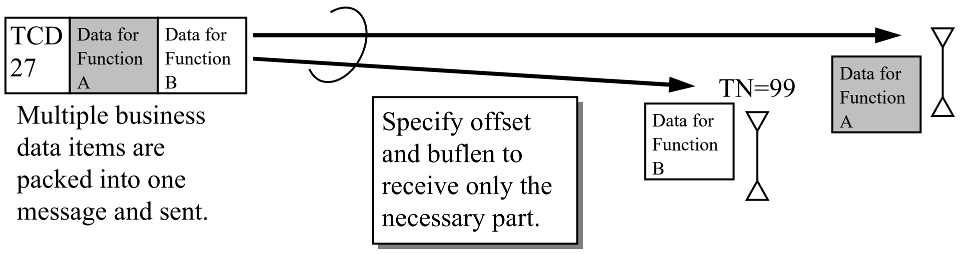
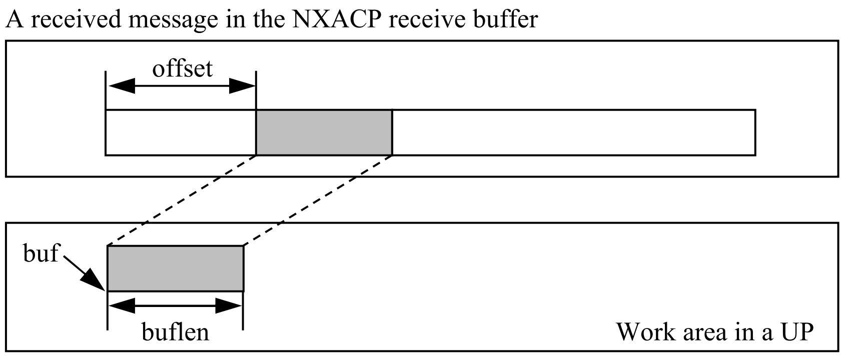

TCD 27
Data for Function A
Data for Function B
Multiple business data items are packed into one message and sent.
Specify offset and buflen to receive only the necessary part.
TN=99
Data for Function B
Data for Function A

Figure 2-20 Receiving Pinpoint Data



buf: The start address in the work area in a UP
buflen: Data size to be received. Also, the size of the receive work area.
offset: Relative address of the pinpoint message to be received
(In relative address, the start address of the whole message is 0.)

Figure 2-21 Receiving Pinpoint Data and Memory Usage

[Caution]
A message received by multiple user tasks is cleared from the buffer when the message is received by all user tasks that must receive the message. If user tasks with different processing durations and priorities receive the same transaction code, buffer efficiency may be reduced, and in worst cases, messages may be discarded due to the receive buffer being full.

(3) Receive buffer usage check
To avoid overflow of the receive buffer caused by malfunction of a specific task, you can define the maximum number of buffer cases used by each transaction code. If this maximum number is exceeded, messages with that transaction code will be discarded when received. By using this feature, you can prevent messages with a specific transaction code from using up the receive buffer.
A user must specify the maximum number of buffer cases used by each transaction code at system construction. You must allocate a large number of buffer cases to transaction codes that must not be discarded.

1.3.2 Receive timeout monitoring

In NXACP, it is recommended that a user task is constructed as an event-driven structure (loop structure).
This is because, if a user task for issuing nx_get() is constructed as a loop and issues a transaction reception request at the beginning of the loop, overhead can be reduced.
However, if a message cannot be received within a certain amount of time, the user task may need to exit from the loop. To allow this, nx_get() has a timeout monitoring feature.

(1) Specifying the timeout monitoring time

Specify the timeout monitoring time in a parameter “time” for nx_get().
You can specify between 0 and 3600 (in seconds: maximum error is ±1).
If you specify 0, nx_get() waits indefinitely for a message to arrive. If you do not want to wait for a message to arrive, specify “-1” for “time”.

(2) Detecting a timeout

The nx_get() macro returns control to the user task with return code 0 when a message is received normally. At the same time, the received data size is set in “rcvlen” in “ginfo”.
The nx_get() macro returns control to the user task with a non-zero return code when terminated with an error.
If a message cannot be received, nx_get() returns control to the user task with return code 0 because nx_get() exits normally as a routine in NXACP, but “-1” is set in “rcvlen” in “ginfo”. If you enable timeout monitoring when you issue nx_get() (time > 0), check the value of “rcvlen” in “ginfo” after you check the return code. Then, determine whether nx_get() exits from the wait due to timeout or because a message has been received, and then handle the result of nx_get() accordingly. Similarly, if you specify non-blocking reception (time = -1), check the value of “rcvlen” in “ginfo”. Then, determine whether nx_get() exits from the wait because a message is not available or because a message has been received, and then handle the result of nx_get() accordingly.

1.3.3 Defining multicast groups

For a data field for a network segment (non-zero DF), you must define the mapping between the multicast group numbers to be received and the receive port numbers. If this definition is not set up, messages targeted to these multicast groups cannot be received.
This mapping must be uniquely defined in all nodes that are connected to the data field.

Table 2-7 Multicast Group Numbers and Receive Port Numbers

MGN	Receive port number
1	5501
11	5511

This mapping must be unique.

You must also define the mapping between the multicast group numbers to be sent and the send destination port numbers. The send destination port number and receive port number corresponding to the same multicast group number must be uniquely defined in all nodes in a data field.

Table 2-8 Multicast Group Numbers and Send/Receive Port Numbers

MGN	Send destination port number
1	5501
11	5511

This mapping must be unique.

MGN	Receive port number	Send destination port number
1	5501	5501
11	5511	5511

Define the receive port in the nodes where you want to receive MG1.
Define the send destination port in the nodes where you want to send MG1.

This mapping must be unique.

A user task specifies a multicast group number when sending a message. This message is broadcast by NXACP to the send destination port number corresponding to the multicast group number specified by the user.
On the other nodes, if the above send destination port number is not defined as a receive port, the message is discarded by hardware or OS. The message can be received by a node only if the same port number as the send destination port number is defined as a receive port on that node.

[Caution]
Note that, if you define both send and receive ports for a multicast group on one node, the node can receive message from other nodes but cannot receive messages from the local node.

Figure 2-22 shows the relationship between ports and multicast group numbers for transmission and reception.

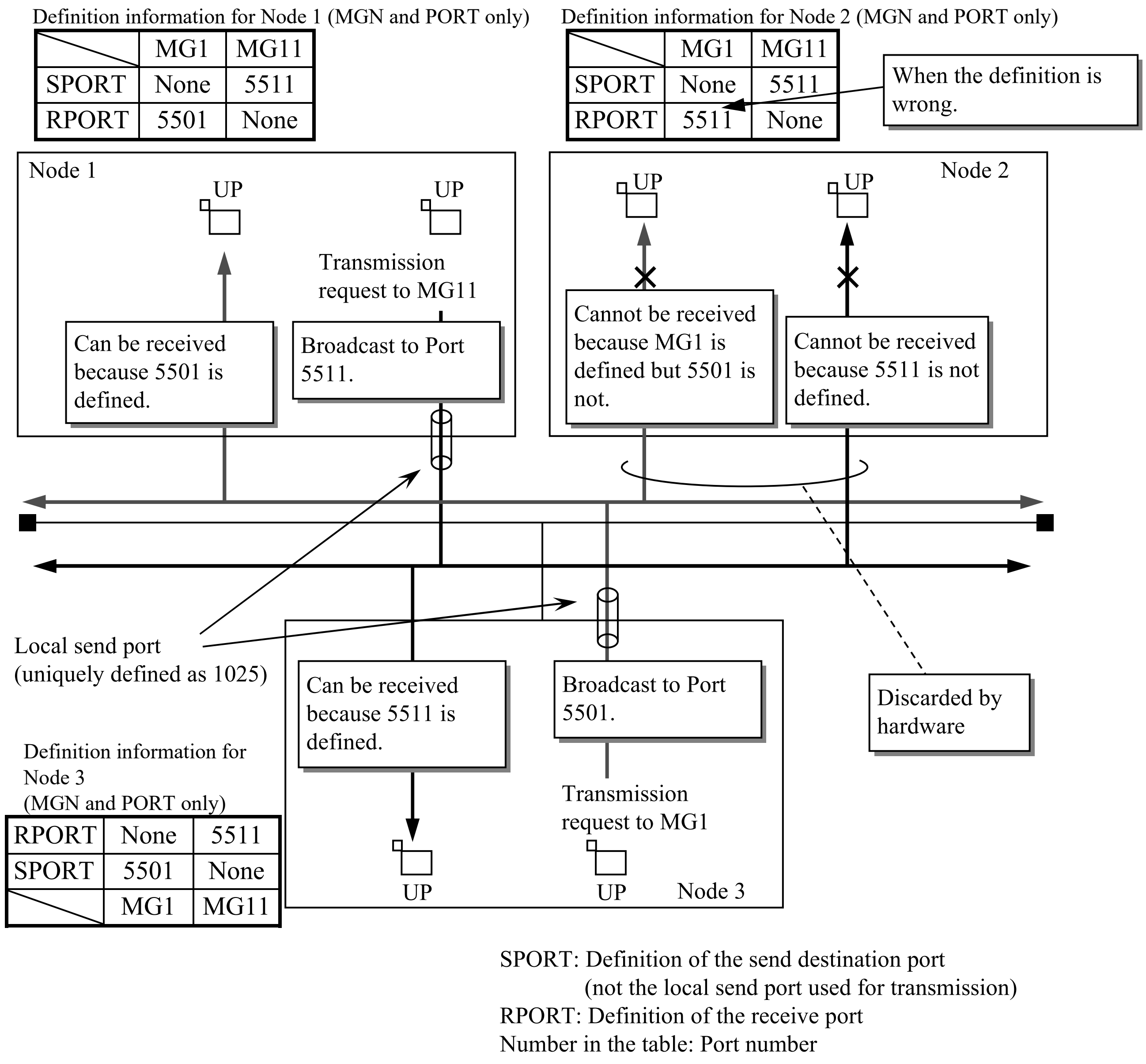

Definition information for Node 1 (MGN and PORT only)
	MG1	MG11
SPORT	None	5511
RPORT	5501	None
Definition information for Node 2 (MGN and PORT only)
	MG1	MG11
SPORT	None	5511
RPORT	5511	None
When the definition is wrong.
Node 1
UP
UP
Transmission request to MG11
Can be received because 5501 is defined.
Broadcast to Port 5511.
Node 2
UP
UP
Cannot be received because MG1 is defined but 5501 is not.
Cannot be received because 5511 is not defined.
Local send port (uniquely defined as 1025)
Can be received because 5511 is defined.
Broadcast to Port 5501.
Discarded by hardware
Definition information for Node 3 (MGN and PORT only)
RPORT	None	5511
SPORT	5501	None
	MG1	MG11
Transmission request to MG1
UP
UP
Node 3
SPORT: Definition of the send destination port (not the local send port used for transmission)
RPORT: Definition of the receive port
Number in the table: Port number

Figure 2-22 Send Destination Multicast Groups and Receive Multicast Groups

1.4 Remote Data Field Control Function

By connecting multiple LANs with routers and nodes and using the routing function of Internet Protocol (IP), the S10VE can access to a large-scale network.

1.4.1 System connection topology

In a large-scale network configuration, seen from one node, LANs fall into two categories: a LAN where the local node is connected to directly (local data field), and a LAN where the local node is connected indirectly through a router, etc. (remote data field).
A remote data field must be connected to only one local data field.

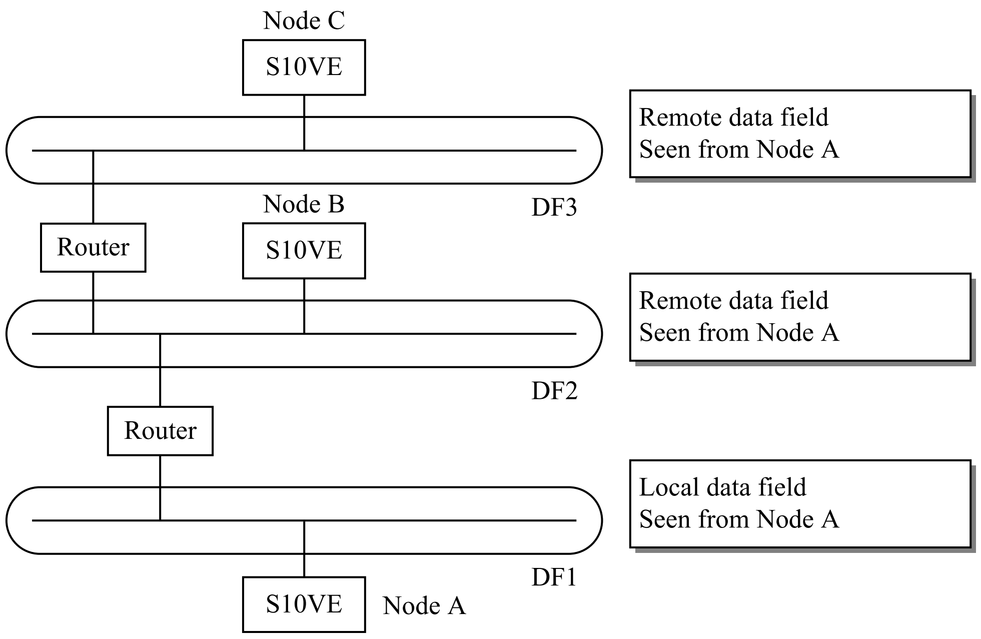


Figure 2-23 Data Field Types

[Caution]
Transmission to a remote data field uses the routing function of Internet Protocol (IP).
The routing function is implemented in RCTLNET, not NXACP.

A remote data field must correspond to only one local data field as a transmission path. You cannot let one remote data field correspond to N local data fields (1-to-N relationship).
If you want to duplex a communication path to improve reliability, see “CHAPTER 3 DUPLEXED LAN CONTROL FUNCTION”.

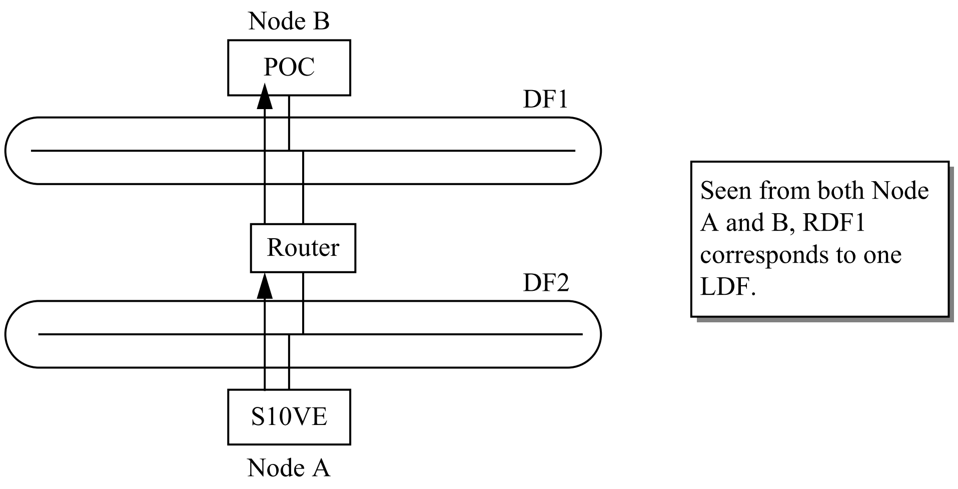


Figure 2-24 Remote Data Field (1)

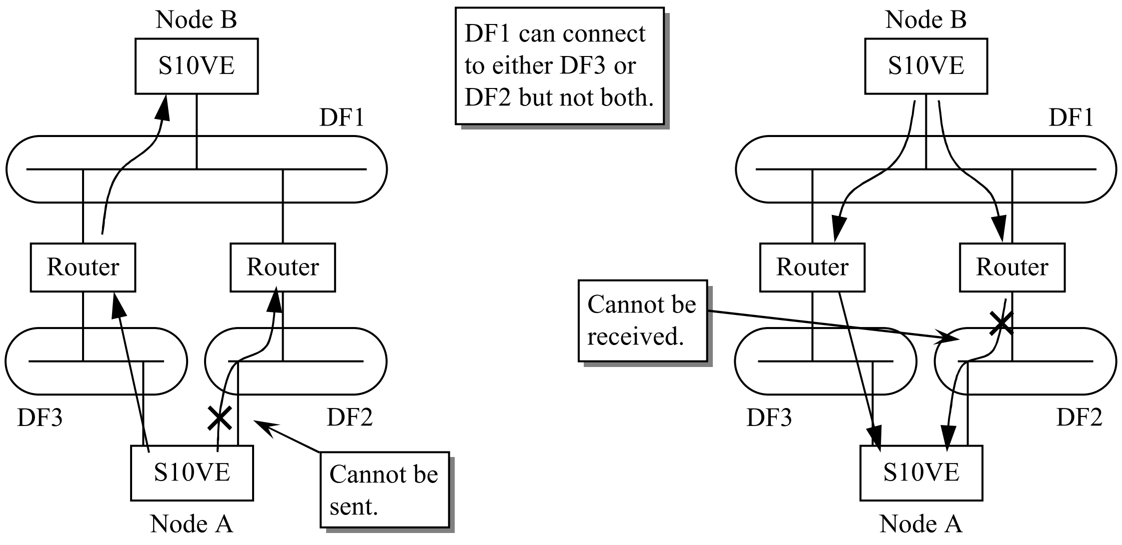


Figure 2-25 Remote Data Field (2)

1.4.2 Message transmission and reception

(1) Destination of communication

You can send a message to a multicast group defined in a data field, regardless of whether the data field is remote or local. However, note that, a message cannot be sent to multiple data fields or multiple multicast groups at the same time in one transmission request.

As for reception, each node can join a multicast group defined in a local data field but cannot join a remote multicast group. Therefore, a node can receive a message sent to the local data field but cannot receive a message sent to a remote data field.

(2) Sending and receiving a message

When you send a message, specify the destination data field number, multicast group number, and transaction code, and issue a transmission request (nx_put()), regardless of whether the destination data field is remote or local. Then, the message can be received only in the specified destination data field.

Even when a message is sent to a remote data field through one or two routers, the message cannot be received in any data fields (data fields in the path, or local data fields) other than the destination data field. However, a received message is always connected to the receive queue of the local data field where the message is sent to, regardless of whether the sender node is remote or local. Therefore, to receive a message, you must define the local data field number and the transaction code and issue a reception request (nx_get()). Note that you can receive only messages sent to the local data fields.

The transmission/reception service to a remote data field is processed by the program for the local data field the remote data field corresponds to. If this local data field is not working, you cannot send a message to the remote data field.

Figure 2-26 summarizes how a message is sent to and received from a remote data field.

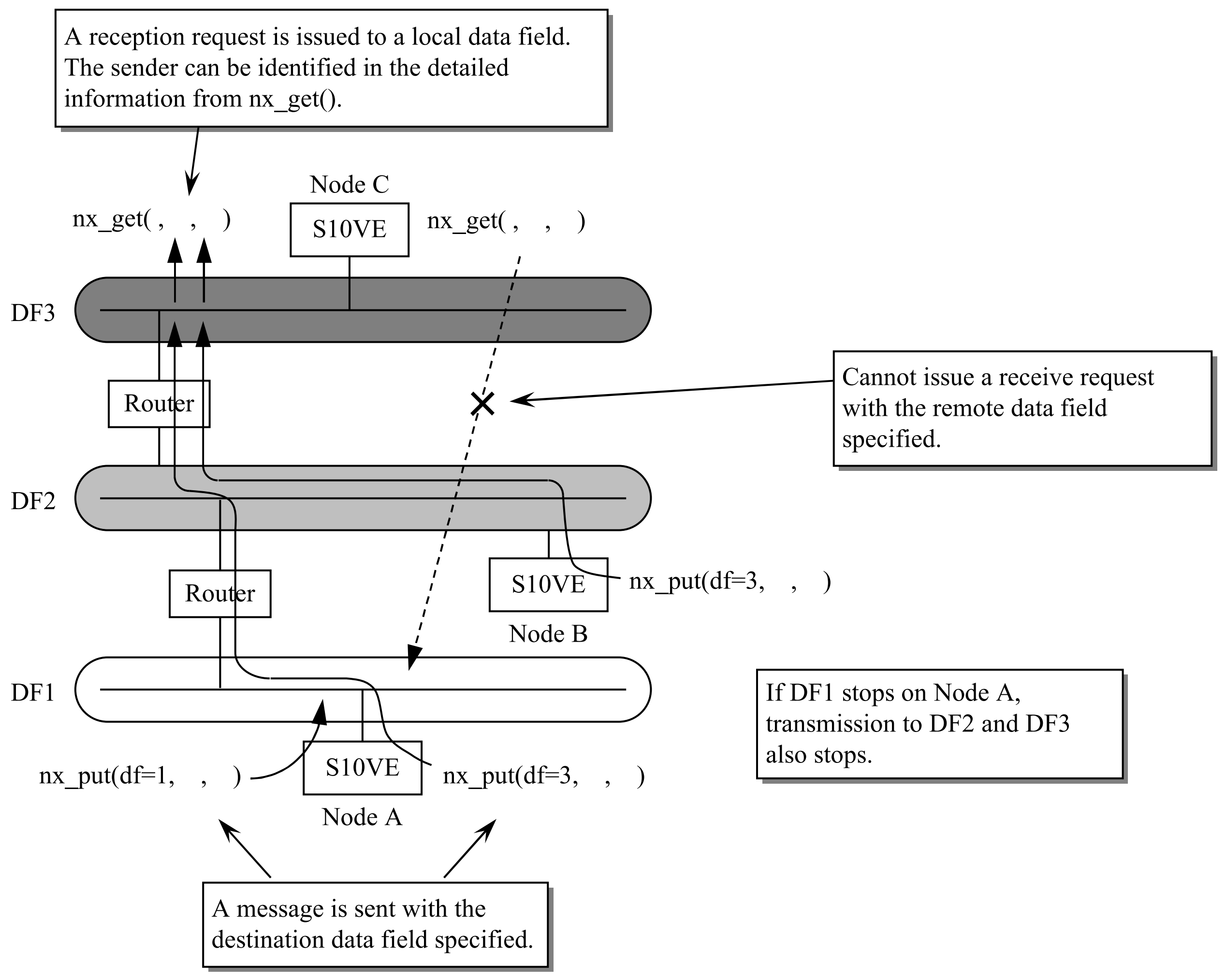

A reception request is issued to a local data field. The sender can be identified in the detailed information from nx_get().
Node C
nx_get(, ,)
S10VE
nx_get(, ,)
DF3
Router
Cannot issue a receive request with the remote data field specified.
DF2
Router
S10VE
nx_put(df=3, ,)
Node B
DF1
If DF1 stops on Node A, transmission to DF2 and DF3 also stops.
nx_put(df=1, ,)
S10VE
nx_put(df=3, ,)
Node A
A message is sent with the destination data field specified.

Figure 2-26 Sending and Receiving a Message to and from a Remote Data Field

Operation of a remote data field is affected by the operating status of the local data field. For details about operation, see "CHAPTER 6 OPERATION MANAGEMENT FUNCTION".

1.5 Buffer Management

Send/receive buffers are managed on a per-data-field basis, and also independently for transmission and reception. You must define a buffer independently for transmission and reception for all local and remote data fields. As an exception, define only a receive buffer for the data field for local node communication (data field number = 0).

Table 2-9 Data Field Types and Buffer Definition

	Local data field (for local node communication)	Local data field (not for local node communication)	Remote data field
Send buffer	Definition is not required.	Definition is required.	Definition is required.
Receive buffer	Definition is required.	Definition is required.	Definition is required.

As buffer management functions, the detection function for the buffer overflow/HIGH-WATER alarm/LOW-WATER alarm and the EAS notification function are supported. For details about buffer management, see “5.1 Failure Notification Function”.

CHAPTER 2 DATA FIELD MANAGEMENT FUNCTION

NXACP provides the following status management functions for data fields.
• Transmission of alive signals from the local node
• Alive status monitor function and status change notification function for other nodes
Each node where NX is installed sends an "alive signal" by broadcasting at a constant interval to all local data fields the node is directly connected to. Other nodes can receive this alive signal. Consequently, the alive/dead state of each node in the data field can be monitored.
In the NX protocol, alive signals have the following three modes.
• Alive report (normal mode)
• Scheduled SHUTDOWN report (scheduled device shutdown)
• Scheduled maintenance report (temporary device shutdown for maintenance)
Out of these three NX protocol alive signal modes, NXACP supports an alive report and a scheduled SHUTDOWN report. A scheduled maintenance report is handled as the equivalent of a scheduled SHUTDOWN report.

2.1 Alive Signal Transmission Function

When NXACP accepts an initial request (nx_dfup() is issued), NXACP starts transmitting an alive signal for an "alive" report at a constant interval (specified at system construction). When NXACP accepts a shutdown request, (nx_dfdwn() or nx_quit() is issued), and then NXACP transmits an alive signal for a "scheduled SHUTDOWN" report. This "scheduled SHUTDOWN" report allows other nodes to determine whether the shutdown is scheduled or caused by an error.

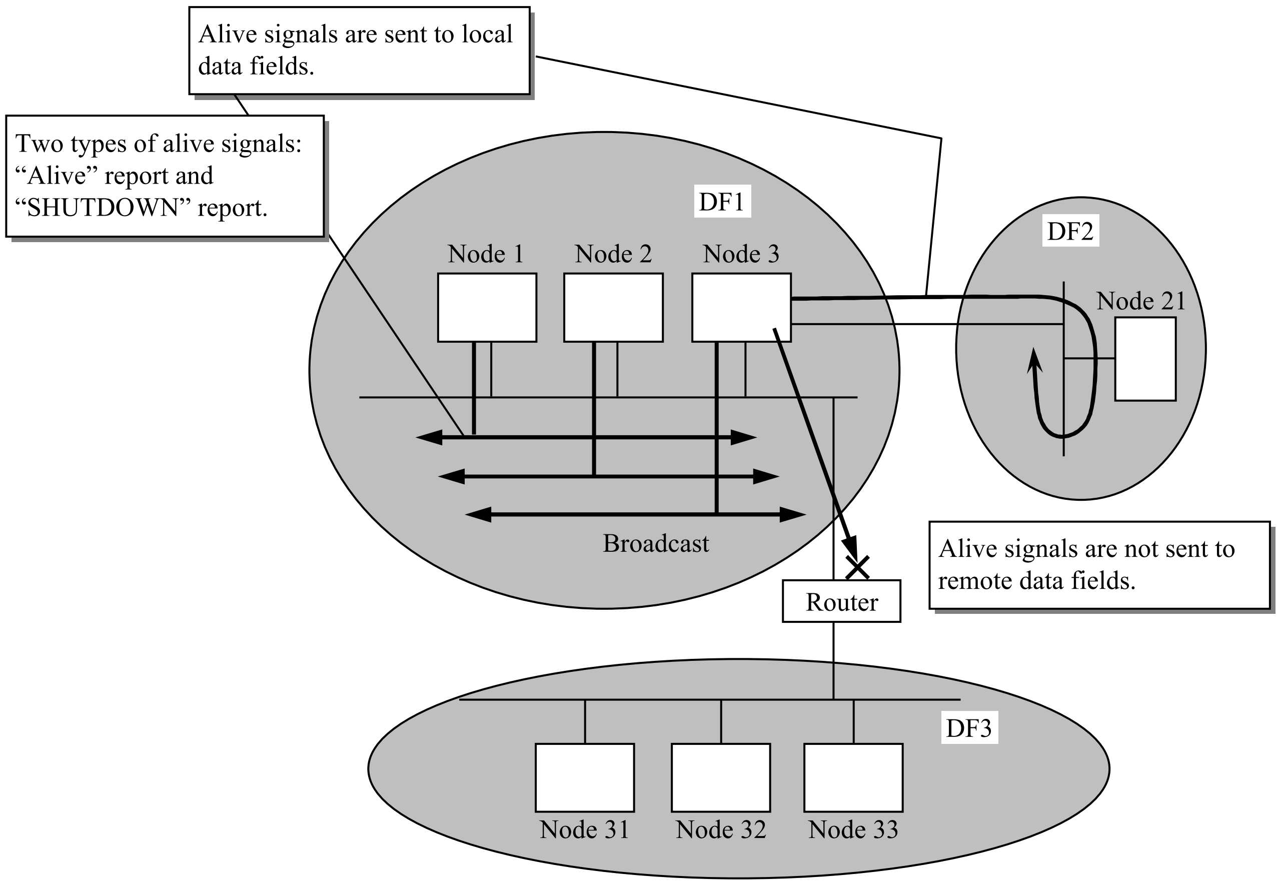

Alive signals are sent to local data fields.
Two types of alive signals: "Alive" report and "SHUTDOWN" report.
DF1
Node 1
Node 2
Node 3
Broadcast
DF2
Node 21
Router
Alive signals are not sent to remote data fields.
DF3
Node 31
Node 32
Node 33

Figure 2-27 Alive Signal

By using alive signals, you can monitor the connection status of other nodes on a per-data-field basis. “Monitor” here does not mean to check whether a specific node is working properly in the system, but rather to monitor whether, for each data field used as the basis of business applications, each node is properly connected to the data field and can work with other nodes cooperatively.
In the example in Figure 2-27, Node 3 is connected to Data fields 1 and 2. If a communication device (cable, Ethernet module, etc.) at the interface between Data field 1 and Node 3 in this configuration fails, other nodes connected to Data field 1 recognize that the status of Node 3 is changed from “alive” to “dead” and notify a user of the change. But in Data field 2, Node 3 is still recognized by other nodes as “alive”.
Table 2-10 summarizes the types of alive signals sent from NXACP.

Table 2-10 Alive Signal Types

Report type	Alive report	Scheduled SHUTDOWN
Transmission start timing	nx_dfup() request from a UP	nx_dfdwn()/nx_quit() request from a UP
Number of transmissions	Indefinitely until a stop request	Three times
Transmission interval	Specified by a user (in seconds)	Immediately, one second, and two seconds after nx_dfdwn()/nx_quit() is issued.

The nxdfup()/nx_dfdwn()/nx_quit() macros are used for starting or stopping NXACP on a per-data-field basis. (For details, see “CHAPTER 6 OPERATION MANAGEMENT FUNCTION”.)
To summarize, when nx_dfup() is called to issue a start request to a data field, an alive signal for “alive report”starts being sent at a constant interval.
When nx_dfdwn() or nx_quit() is called to issue a shutdown request, an alive signal for “scheduled SHUTDOWN” is sent three times, namely immediately after the macro is called, one second later, and two second later.
The value of the transmission interval of “alive report” can be specified in seconds by a user at system construction. At the same time, a user can define the following information shown in Table 2-11. Each piece of information is attached to an alive signal and broadcast. The contents can be checked by the dfstat command of NX Dlink.

Table 2-11 Additional Information in an Alive Signal

User-defined information	Description
Node number	Number uniquely assigned in each data field
Node name	Name uniquely assigned in each data field
Alive signal timeout monitoring time	The time it takes to recognize the state is changed to “death” after alive signals are cut off. (For details see “2.2 Alive Status Monitor and Status Change Notification Function”.)
IP address	INA assigned to each logical node

The node mode when an alive signal is transmitted is set to the mode specified in a parameter of nx_dfup().

2.2 Alive Status Monitor and Status Change Notification Function

The alive status monitor function monitors the arrival of alive signals sent at a constant interval from other nodes on a per-node basis. This function checks whether alive signals are cut off or whether the received alive signal is changed from an “alive” report to a “scheduled SHUTDOWN (scheduled maintenance)” report.
The status change notification function notifies the change and the node number to a user when the status of one of the other nodes has changed.
Alive status monitor must be enabled or disabled by a user at system construction. Status change notification can be used only when alive status monitor is enabled.

[Caution]
Note that, when the alive status monitor function is enabled, the load of a controller that receives alive signals may significantly rise depending on the number of nodes connected to the monitored data field and the alive signal transmission interval of other nodes. Under normal circumstances, it is recommended that alive signals are monitored by a WS or PC, or by a controller dedicated to data field monitoring (if no WSs or PCs are installed in the data field).
The alive signal transmission interval is especially critical. If all nodes connected to the data field send an alive signal simultaneously, the maximum number of nodes that can be monitored in Built-in Ethernet is 64. If the S10VE monitors 64 or more nodes, an alive node can be detected as “dead”, regardless of whether the node is alive, because alive signals cannot be received.
If you want to monitor the alive status of more than 64 nodes, use a WS or PC for this purpose.

(1) Status management
The following lists the node statuses recognized by NXACP.

Alive (connected):	An alive signal (alive report) arrives periodically. The node is considered to be connected to the data field and healthy.
Dead (disconnected):	No alive signals have arrived for a predefined amount of time. Or a scheduled SHUTDOWN (scheduled maintenance) report has arrived. The node is considered to be disconnected from the data field and dead.

The predefined amount of time equals the alive signal timeout monitoring time specified by a user in construction information. The alive signal timeout monitoring time defines the time it takes to recognize a node is dead after alive signals supposed to arrive periodically from the node are cut off. Therefore, the following relationship must be observed between the alive signal timeout monitoring time and the alive signal transmission interval.

Alive signal timeout monitoring time > Alive signal transmission interval

Due to the instability and excessive load of the network, alive signal packets can be lost. It is recommended that the alive signal timeout monitoring time is long enough to assure that missing only one alive signal does not cause the node to be detected as dead.

(2) Change notification
NXACP notifies to a user when the status change of a node is detected.
Note that both an alive signal for scheduled SHUTDOWN and an alive signal for scheduled maintenance are handled in the same way, and reported as "dead due to scheduled SHUTDOWN".

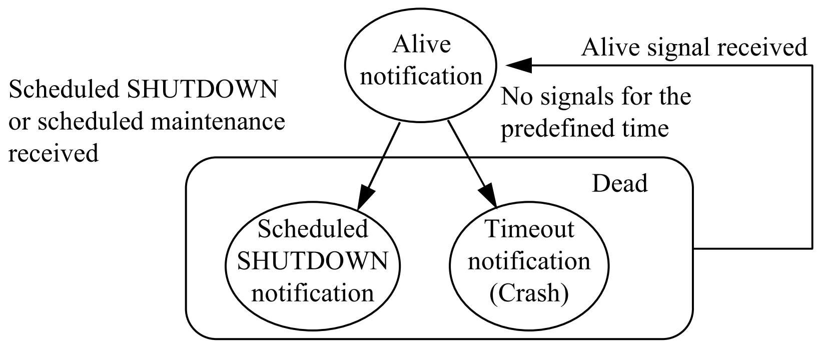

Alive notification
Alive signal received
Scheduled SHUTDOWN or scheduled maintenance received
No signals for the predefined time
Dead
Scheduled SHUTDOWN notification
Timeout notification (Crash)

Figure 2-28 Change Notification

The following shows the time sequence of change notifications reported to a user when status change occurs.

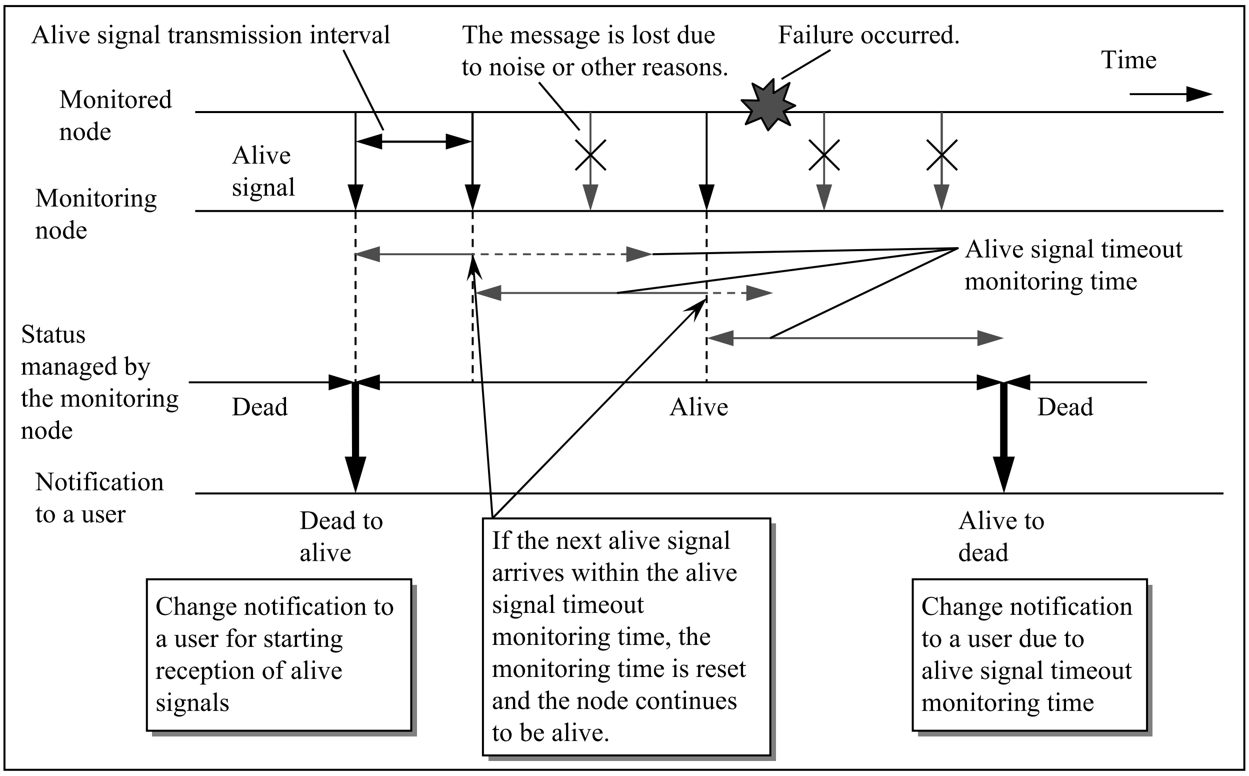

Alive signal transmission interval
The message is lost due to noise or other reasons.
Failure occurred.
Time
Monitored node
Alive signal
Monitoring node
Alive signal timeout monitoring time
Status managed by the monitoring node
Dead
Alive
Dead
Notification to a user
Dead to alive
Alive to dead
Change notification to a user for starting reception of alive signals
If the next alive signal arrives within the alive signal timeout monitoring time, the monitoring time is reset and the node continues to be alive.
Change notification to a user due to alive signal timeout monitoring time

Figure 2-29 Time Sequence for Change Notifications (Timeout)

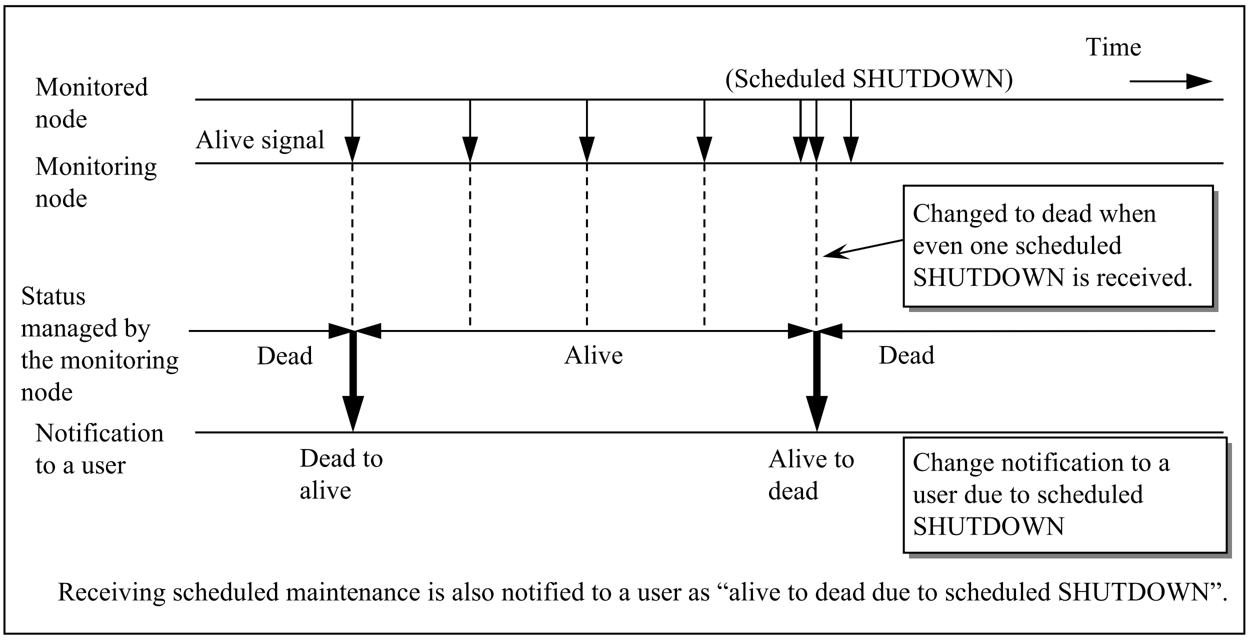

Time
(Scheduled SHUTDOWN)
Monitored node
Alive signal
Monitoring node
Changed to dead when even one scheduled SHUTDOWN is received.
Status managed by the monitoring node
Dead
Alive
Dead
Notification to a user
Dead to alive
Alive to dead
Change notification to a user due to scheduled SHUTDOWN
Receiving scheduled maintenance is also notified to a user as “alive to dead due to scheduled SHUTDOWN”.

Figure 2-30 Time Sequence for Change Notifications (Scheduled SHUTDOWN)

(3) Conditions on notification to a user
Status change notification to a user always links to IRSUB (#332). For details, see "APPENDIX C NODE STATUS CHANGE NOTIFICATION FORMAT". The following describes the timing of a notification and its nature.
Node status change notifications can be detected only by a user (UP) on a node that uses the status monitoring function and that is in the local data field of the local node.
"Change to alive" is notified to a user when an alive signal from one of the other nodes arrives after nx_dfup() is issued. However, "change to dead" is not notified to a user when scheduled SHUTDOWN or scheduled maintenance is received from one of the other nodes if no alive signals for an alive report have ever been received from the node.
"Change to dead" is reported for a node only if at least one alive signal for alive report has been received from the node after nx_dfup() is issued. "Change to dead" is detected at the following time: when alive signal timeout occurs and when an alive signal for scheduled SHUTDOWN or maintenance is received.

[Caution]

- Handling of the local node
  As for status change of the local node, change to "alive" is notified to a user only once when the processing of nx_dfup() is complete after nx_dfup() is issued. Note that change to "dead" is not notified to a user when nx_dfdwn() or nx_quit() is issued.
- Handling of a stop request
  After nx_dfdwn() or nx_quit() is issued to a data field, the processing of the data field stops. Consequently, change to "dead" is not reported for all nodes in the data field, including the local node. You must recognize all nodes as "dead" when processing of nx_dfdwn() is complete.
- To enable change notification, the node must be in the mode that allows receiving of alive signals (you must specify this in construction information), and IRSUB (#332) must be registered before NXACP starts (nx_init()).

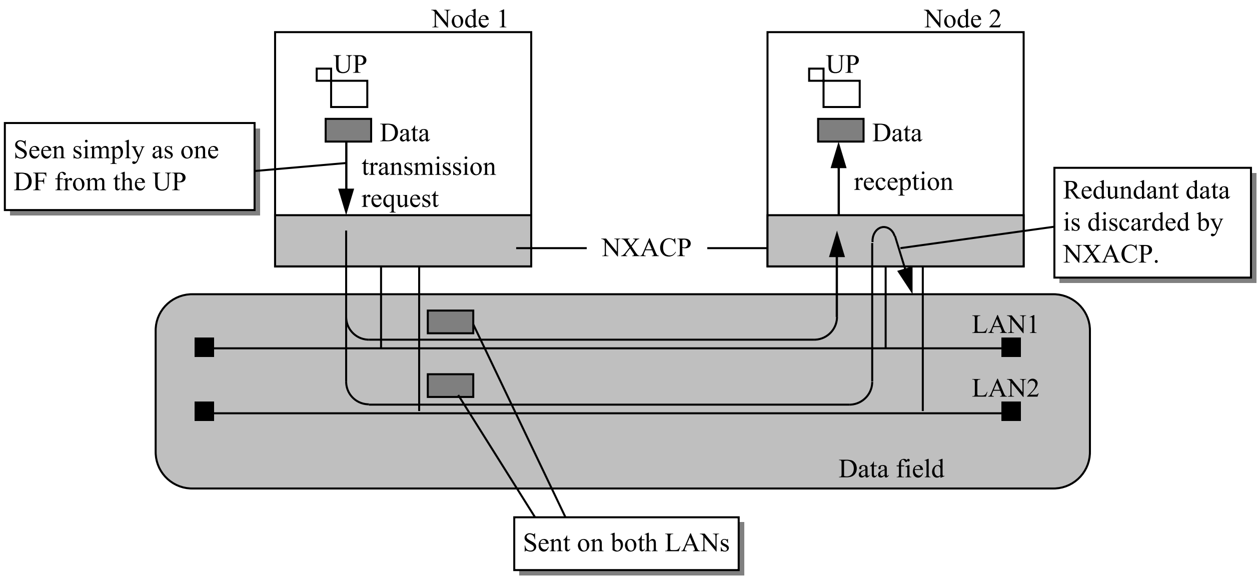
CHAPTER 3 DUPLEXED LAN CONTROL FUNCTION

Depending on the characteristics of the Ethernet network, packets might drop out when power to the controller is turned on or off, or when network load increases. If this situation occurs, transmission must be retried. However, these retries may increase the network/CPU load and may result in excessive overall system load.
The purpose of duplexed LAN control in the NX multicast communication is to reduce loss of data during transmission, and to i

3.1 Message Transmission and Reception on Duplexed LANs

The following shows some characteristics of the duplexed LAN control function in terms of how the function is used.

- From a user task that uses NXACP, physically two LANs are seen as one data field. The user task does not have to take care of LAN1 and LAN2 directly.
- In the duplexed communication method supported by NXACP, a message is sent to both LANs, and a message is received from both LANs. If the same messages are received, the redundant message is discarded by NXACP. As a destination send port number or receive port number corresponding to a multicast group number, the same port number is used for both LANs. Therefore, you do not have to switch LANs (configuration control) when one of the LANs fails.



Alive signal messages are also sent to both LANs. Therefore, messages are not lost even when one of the LANs fails. Configuration control is not necessary when one of the LANs fails.

Figure 2-31 Message Transmission and Reception on Duplexed LANs

[Caution]
Only the same types of networks can constitute duplexed LANs. The rationale behind this is as follows. If a message is sent to NXACP by way of duplexed LANs, and the arrival time of the message is considerably different between both LANs due to a delay on network communication paths, NXACP may mistake a message received for the first time as redundant data, and may discard the message.

If duplexed LANs are used to connect to a remote data field, both local and remote data fields must be constructed as duplexed LANs.

● If both local and remote data fields are constructed as duplexed LANs, a message is transmitted to corresponding LANs for each field. Make sure that routing addresses are consistent throughout LANs and routers when you construct a system.

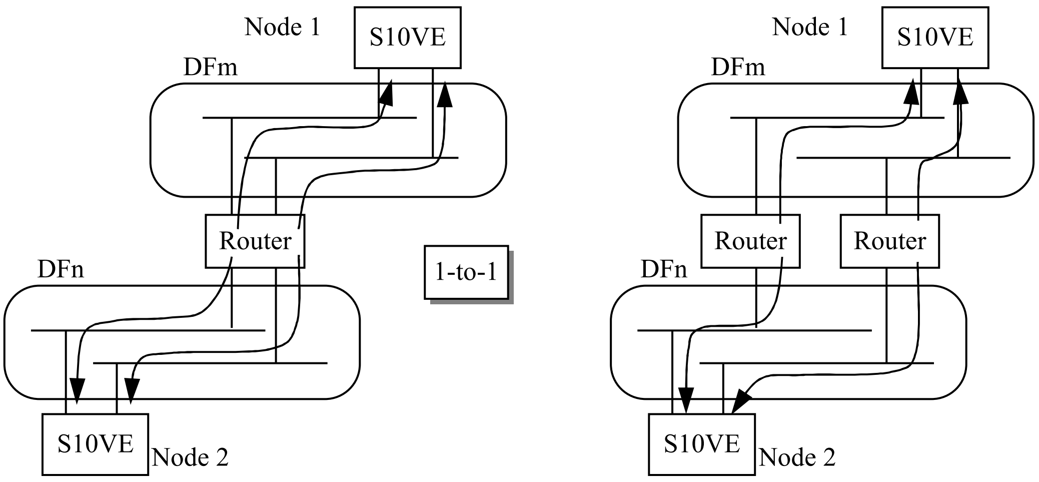


Figure 2-32 Duplexed LANs and Remote Data Field (1)

Also in duplexed configuration, LANs used in remote and local data fields must correspond 1-to-1. Using multiple path setting is not allowed.

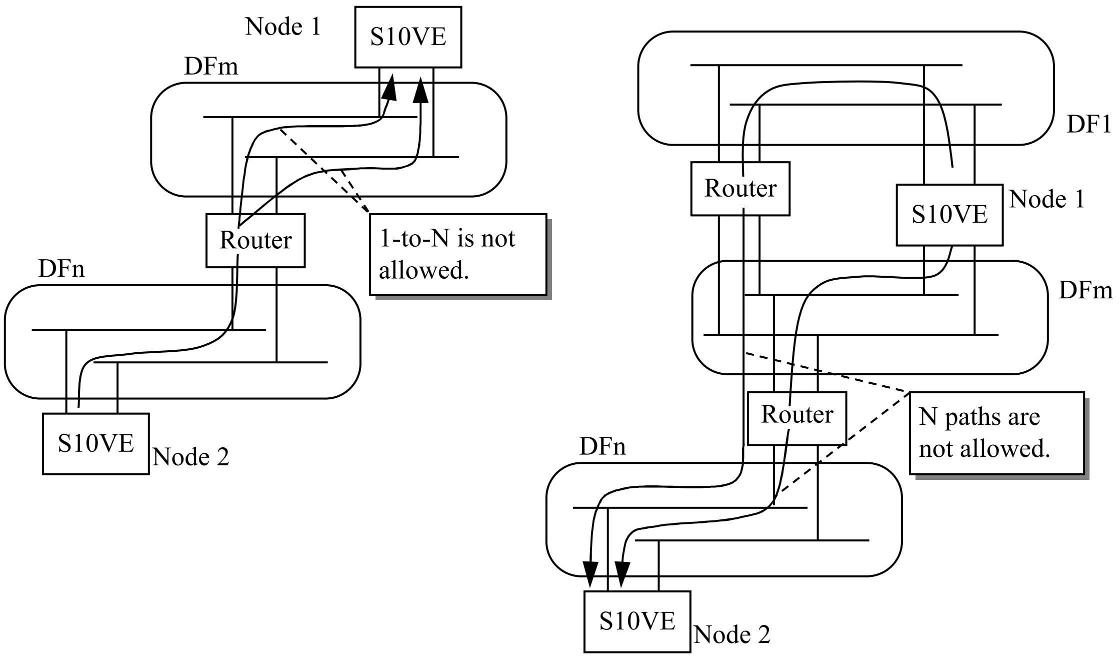


Figure 2-33 Duplexed LANs and Remote Data Field (2)

It is possible to use duplexed LANs in only one of the local and remote data fields. However, the following restrictions apply.

- If the number of LANs is different between the local data field and remote data field, each message must be sent the same number of times as the number of LANs in the destination data field. If the remote data field uses duplexed LANs and the local data field uses a single LAN, each message must be sent twice.

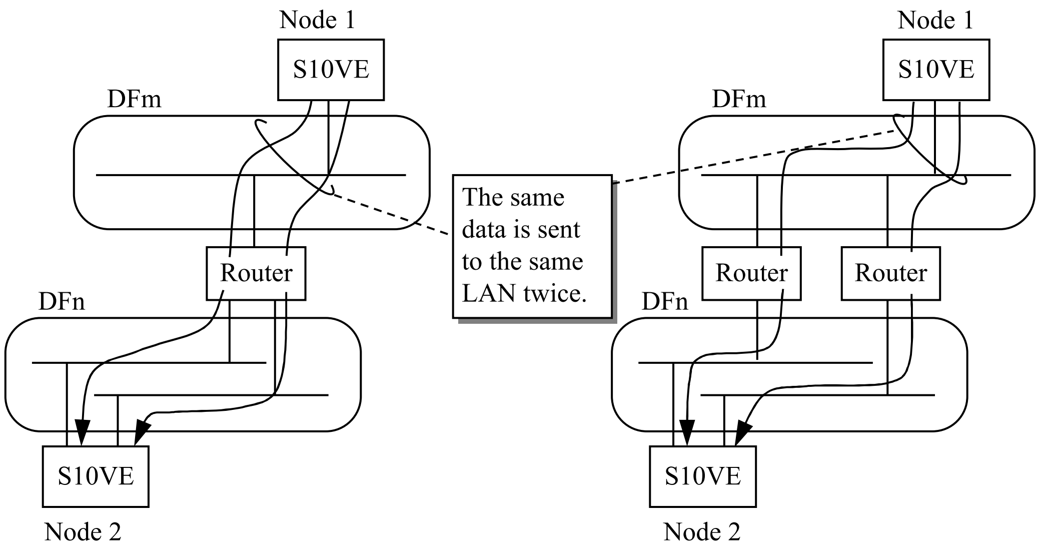


Figure 2-34 Duplexed LANs and Remote Data Field (3)

If the local data field uses duplexed LANs and the remote data field uses a single LAN, the LAN defined first in the construction information for the local data field (LAN defined for "UNO1" in the data field definition information) will correspond to the LAN in the remote data field.

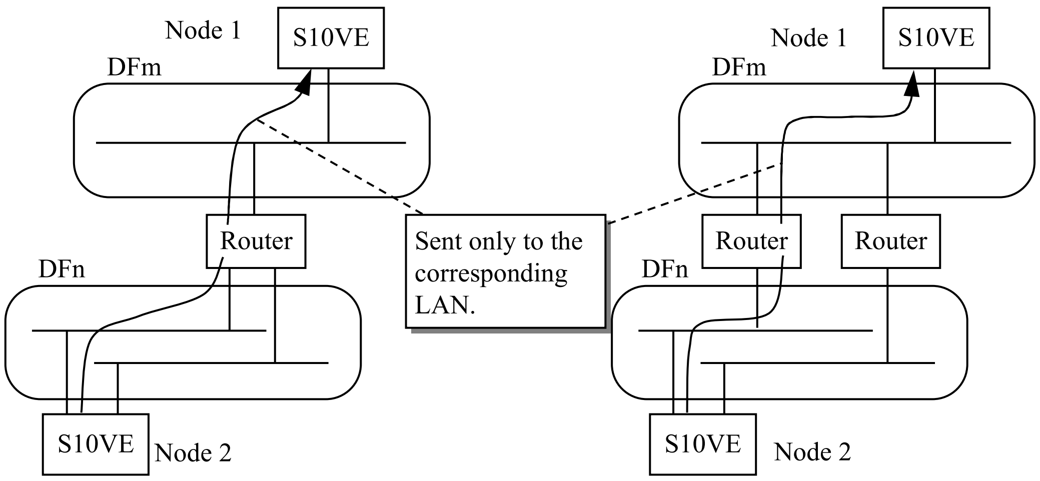


Figure 2-35 Duplexed LANs and Remote Data Field (4)

3.2 Alive Signal Transmission and Reception on Duplexed LANs

When a data field is constructed as duplexed LANs, alive signals are sent to both LANs. Other nodes are monitored for each LAN, and as a result, this function allows you to determine whether a failure is generated in the nodes or in the network.

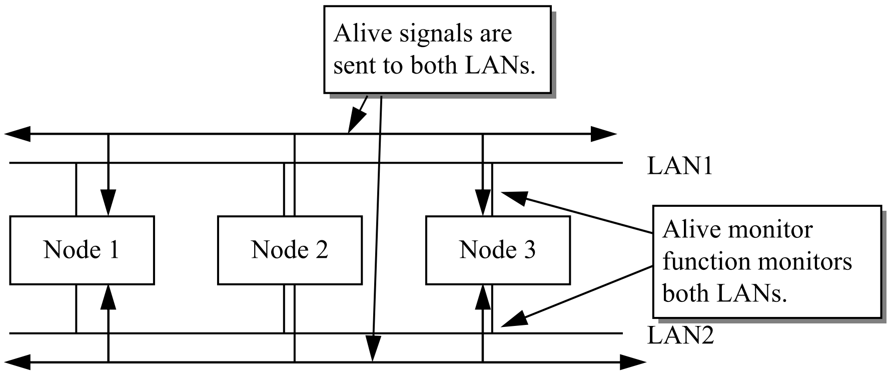

Alive signals are sent to both LANs.
LAN1
Node 1
Node 2
Node 3
Alive monitor function monitors both LANs.
LAN2

Figure 2-36   Duplexed LANs and Alive Signals

(1) Sending an alive signal
When an alive signal is sent to duplexed LANs, an alive signal with the same format and data as an alive signal for a single LAN is sent to both LAN1 and LAN2.

(2) Status monitor and change monitor
Status monitor and change monitor are performed for each LAN.
Consequently, even if alive signals are constantly coming from LAN1, when alive signals from LAN2 are cut off and the alive signal timeout monitoring time has passed without receiving any alive signals from LAN2, “Node N on LAN2 has changed from alive to dead” is notified to a user. Also “dead to alive” is notified for each LAN.
Consequently, when status change occurs on a node, the status change is normally reported to a user twice, one from LAN1 and the other from LAN2. However, note that due to the possibility of packet loss or other reasons, status change notification is not guaranteed.
For details about the format of the status change notification, see “APPENDIX C   NODE STATUS CHANGE NOTIFICATION FORMAT”.

CHAPTER 4 TEST FUNCTION

The test function is a support function that facilitates testing of user programs in a variety of test configurations. As you gradually expand the system, you can test each node independently without changing existing components that are running online and without creating a system only for a test.
In NXACP, to allow testing without affecting existing components, a mode is assigned to both nodes and messages, and message I/O is controlled on a per-node basis by using the combination of these two mode types. By limiting messages input to and output from nodes on a per-node basis, a limited number of components can run in the system, while logically being isolated from the rest of the system.
Note that the functions described in this chapter can be used only for multicast communication.

4.1 Message I/O Control

A user must specify message I/O control modes at system construction.
The possible combinations of node mode and message modes for message I/O control are as follows.

Table 2-12 Node Mode and Message Mode

Node mode	Message mode of sending messages	Message mode of messages that can be received
Online	Online	Online
Test	Test	Test
Test	Test	Online

An online node sends only online messages and can receive only online messages. The ports for test messages are not initialized and test messages are discarded at the OS level so that applications are not disturbed by test messages.
A test node sends only test messages. You can select one of the following options for the mode of messages that can be received by the node.
• Only test messages
• Only online messages
Regardless of which input option is selected, the ports for the mode unnecessary for reception are not initialized and messages in the unnecessary mode are discarded at the OS level to prevent messages in the unnecessary mode from disturbing applications.

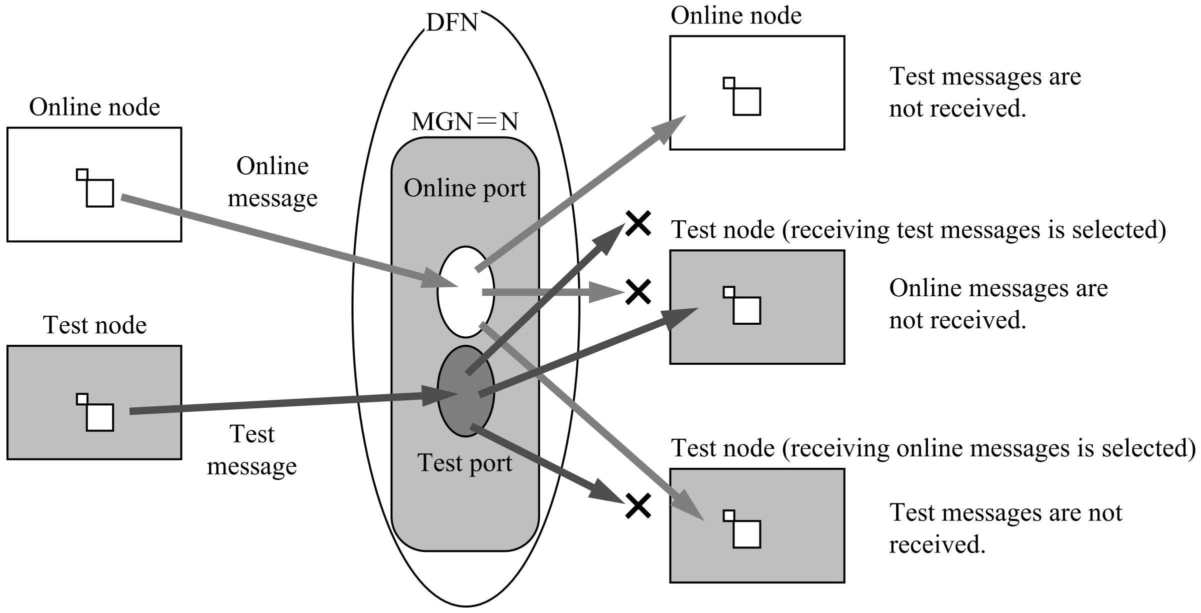

DFN
Online node
Test messages are not received.
Online node
MGN＝N
Online message
Online port
Test node (receiving test messages is selected)
Online messages are not received.
Test node
Test message
Test port
Test node (receiving online messages is selected)
Test messages are not received.

Figure 2-37 Message I/O Control

In Figure 2-37, when the online node sends an online message to Multicast group N, NXACP selects the online port number corresponding to Multicast group N as the destination send port number and sends the message to the port. Therefore, when a user specifies a node to be an online node at system construction, the user must define an online port number corresponding to Multicast group N for transmission.
Similarly, when a node receives an online message from Multicast group N, a user must define an online port number corresponding to Multicast group N for reception.
Note that the online port number corresponding to Multicast group N described above must be the same across a data field.
In NXACP, one multicast group corresponds to (is mapped to) a pair of port numbers: one for online, and the other for test.
A combination of message I/O control modes is specified in a parameter of nx_dfup(). To allow necessary messages to be received and unnecessary messages to be discarded at the OS level, you must ensure that the same mapping from multicast groups to online/test mode port numbers is defined on all nodes in a data field so that the same combinations of mode, multicast group number and port number are used across the data field.
Only the ports for the mode specified by a user in a parameter of nx_dfup() are actually used by NXACP.

Table 2-13 Mode and Port Number

MGN	Online mode port number	Test mode port number
1	55001	57001
23	55023	57023
99	55099	57099
101	55101	57101

The port numbers used by NXACP for transmission and reception are the ones for the mode specified in a parameter of nx_dfup().

This correspondence must be used in all nodes in the data field.

4.2 Test Configuration

The following shows examples of test configurations that use message I/O control.

(1) Test by receiving online messages

When you add or update a user program for the S10VE, you can test the program using existing online messages, without the need to build a new system only for testing.

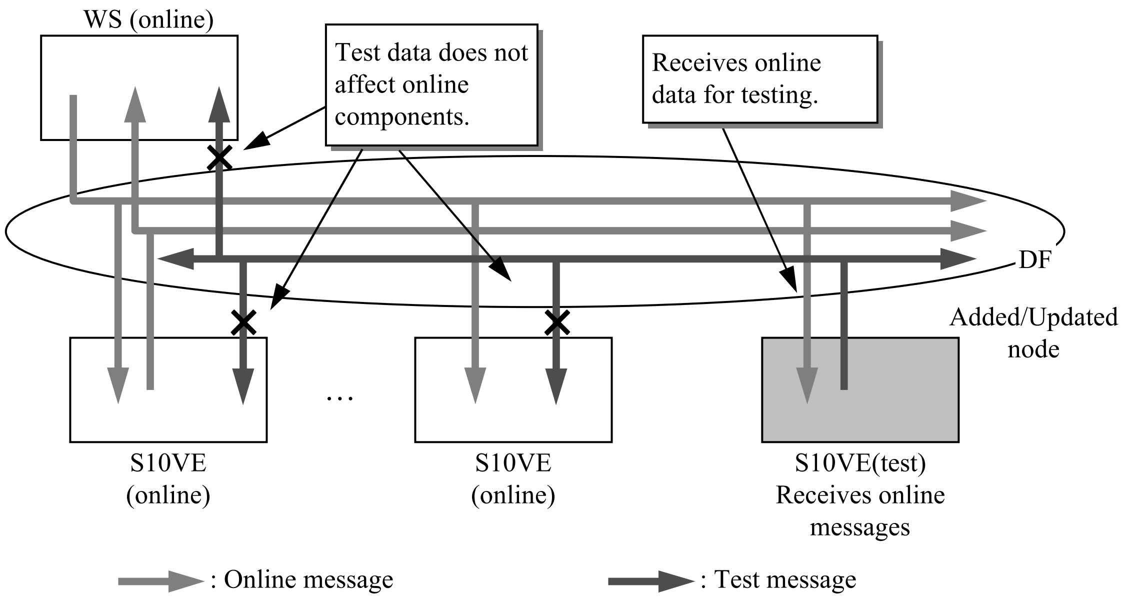

WS (online)
Test data does not affect online components.
Receives online data for testing.
DF
Added/Updated node
…
S10VE (online)
S10VE (online)
S10VE(test) Receives online messages
: Online message
: Test message

Figure 2-38 Test Configuration (1)

(2) Test by receiving test messages

For example, when you want to add a new S10VE and WS and perform a test where the S10VE receives test messages from the WS and the S10VE returns the responses to the WS, you can directly connect the test WS and S10VE to the online network, and then test them without setting up a new test environment.

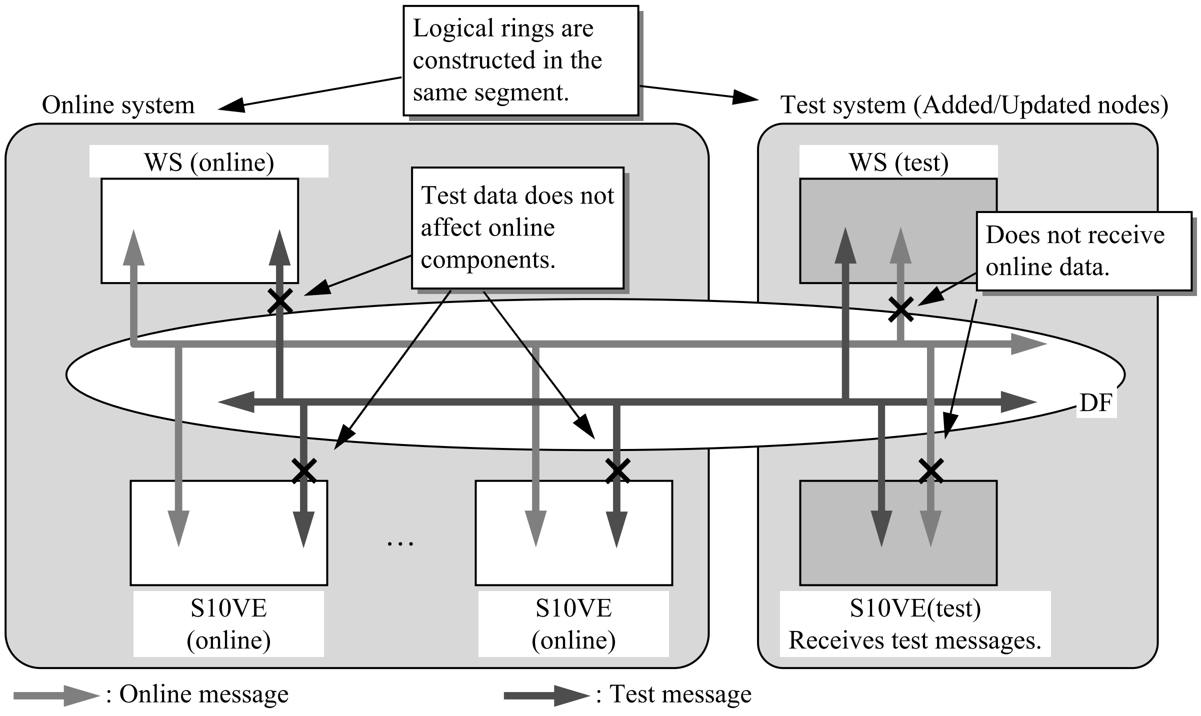

Logical rings are constructed in the same segment.
Online system
Test system (Added/Updated nodes)
WS (online)
Test data does not affect online components.
WS (test)
Does not receive online data.
DF
…
S10VE (online)
S10VE (online)
S10VE(test) Receives test messages.
: Online message
: Test message

Figure 2-39 Test Configuration (2)

(3) Test by receiving both test and online messages
When you want to add a new S10VE and WS, you can perform a test where the S10VE receives test messages from an online WS and the S10VE returns the responses to the test WS. If you use the journaling feature of NX Dlink, you can easily compare the processing result between the online S10VE and the test S10VE.

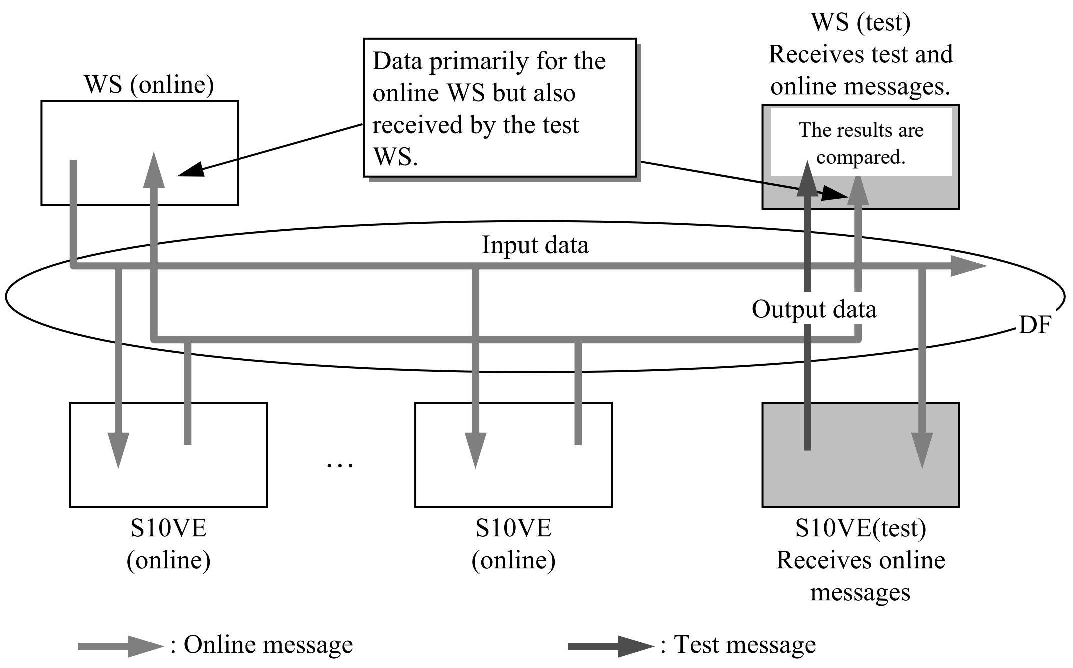


Figure 2-40 Test Configuration (3)

4.3 Remote Data Field and Mode

You can select a node mode uniquely across one data field. Consequently, you can select the mode of messages sent to a data field uniquely across the data field.
For example, in the following figure, if you set the node mode of the local data field (DF1) to online and set the node mode of the remote data field (DF2) to test, the message mode of a message broadcast to DF1 becomes online, and the message mode of a message broadcast to DF2 becomes test.

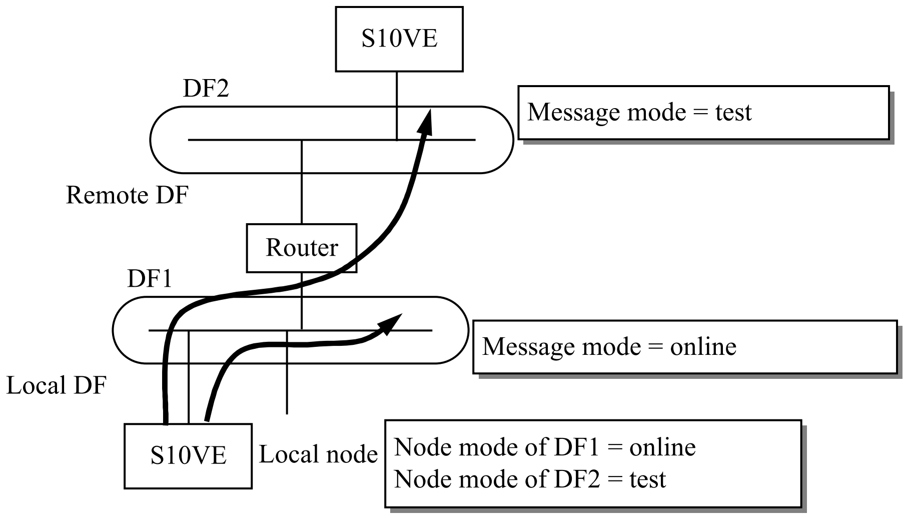


Figure 2-41 Remote Data Field and Mode (1)

At a node that receives messages from a remote data field, the message mode of the messages to be received at the local mode must be set to the same mode as the node mode set by the sender node in the remote data field (that is, the message mode of the messages sent from the sender node).

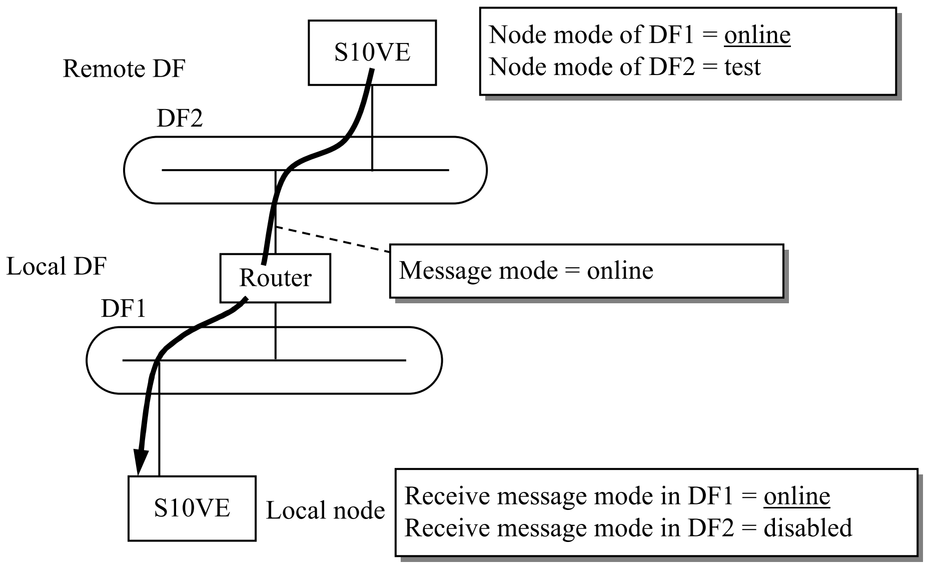

Remote DF
S10VE
Node mode of DF1 = online
Node mode of DF2 = test
DF2
Local DF
Router
Message mode = online
DF1
S10VE
Local node
Receive message mode in DF1 = online
Receive message mode in DF2 = disabled

Figure 2-42 Remote Data Field and Mode (2)

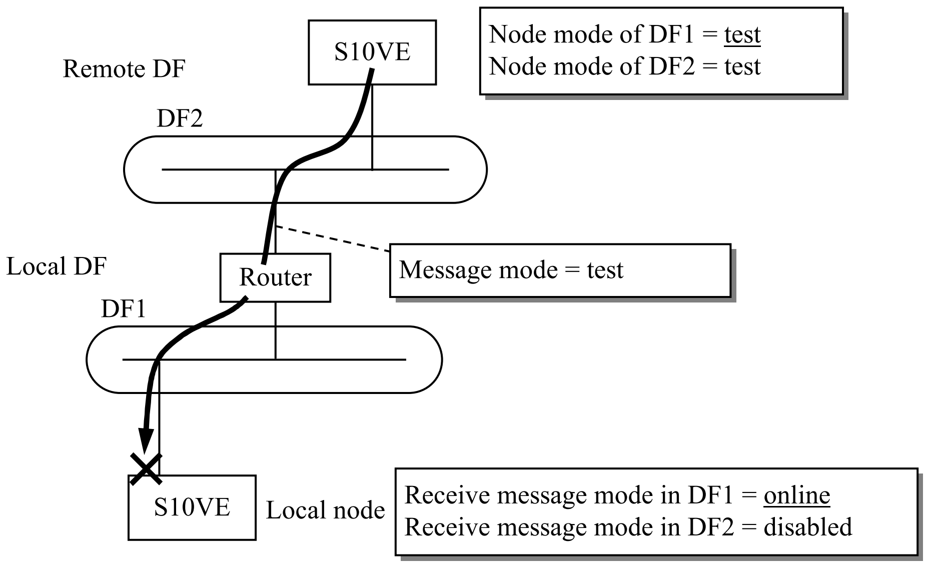


Figure 2-43 Remote Data Field and Mode (3)

CHAPTER 5 SYSTEM MANAGEMENT FUNCTION

When NXACP detects a failure and status change of a node, a description and detailed information of the failure are reported to EAS. This function is called the failure notification function. NXACP also provides a user task management function, which monitors the status of a user task that receives messages, and if the task cannot receive messages, NXACP discards those messages. Collectively, these functions are called the system management function.
In addition to the above, the system management function includes the DHP code trace function and the control trace function.
For information about the DHP trace, see “APPENDIX D DHP RECORD LIST”. For information about the control trace, see “APPENDIX E CONTROL TRACE”.

5.1 Failure Notification Function

When one of the following events occurs, NXACP passes detailed information of the event to EAS or the dedicated IRSUB.
For details about EAS, refer to *S10VE Software Manual CPMS General Description and Macro Specifications* (manual number SEE-3-201).

Table 2-14 EAS Notification Event List

Event name	Event description
Node status change	The status of the node changed (alive to dead or dead to alive).
Buffer status report	The usage ratio of the send/receive buffer exceeded the threshold or the buffer overflowed.
Protocol error	An error was detected in the NX header of the received message.
Socket failure	A socket-related failure was detected.
Transfer area duplication error	For the same area in the transfer memory, the write definition is detected in 2 or more nodes. (For software transfer only).

For details about each event, see “APPENDIX B LOG FORMAT” and “APPENDIX C NODE STATUS CHANGE NOTIFICATION FORMAT”.

[Note]
Failures generated in the Built-in Ethernet network module are notified to EAS by RCTLNET.
For information about network module failures and how to troubleshoot them, refer to the *S10VE User’s Manual General Description* (manual number SEE-1-001).

(1) Node status change

Node status change is notified to a user only when alive signals are received from other nodes (specified at system construction). Node status change is reported if the status of one of the other nodes is changed from alive to dead or from dead to alive.
When duplexed LANs are used, a user is also notified of the LAN number of the LAN where change is detected. This allows a user to detect which LAN has failed.
For this notification, the detection timing and others are different depending on the type of status change.

- Dead to alive

  [Condition]

  Notified if the first alive signal is received after the node starts, or if an alive signal is received again after the node is detected as dead.

  [Detection timing]

  Detected in the node status change check routine launched when an alive signal is received from the network.

- Alive to dead

  [Condition]

  Notified if an alive signal for scheduled SHUTDOWN or scheduled maintenance is received after a normal mode alive signal is received, or if a timeout occurs in the alive signal timeout monitor function after a normal mode alive signal is received.

  [Detection timing]

  An alive signal for scheduled SHUTDOWN or scheduled maintenance is detected by the node status change check routine launched when an alive signal is received from the network. A timeout is detected by the node status change check routine in the node status change monitor task activated every one second.

  [Caution]

  In NXACP, the status monitor function for other nodes is optional and must be enabled by a user (in construction information) to use the function. The purpose of this is to allow you to reduce the load of the controller for receiving alive signals and monitoring the status change of nodes. Status change notifications are sent to the dedicated IRSUB (#332). If it is not required for a node to monitor the status change of other nodes, we recommend running the node with the node connection status monitoring being turned off for the node.
  If WSs or PCs are connected to the network system, we recommend, if possible, using WSs or PCs for monitoring the status change of nodes. If you must monitor the status change of nodes when WSs or PCs are not connected to the network system, we recommend assigning specific controllers to monitor the status change of nodes and not letting the other controllers receive alive signals.
  We also recommend that the controllers in charge of monitoring the status change of nodes are dedicated to this purpose and are given a sufficient margin in the system design to accommodate the load of devices to be added in the future.

In a system with duplexed LANs, this change notification allows you to determine whether the failure location is isolated in a node or in the network, and to monitor the whole system. Table 2-15 shows examples of system monitoring.
Failure monitor patterns in the system configuration in Figure 2-44 are shown as examples.

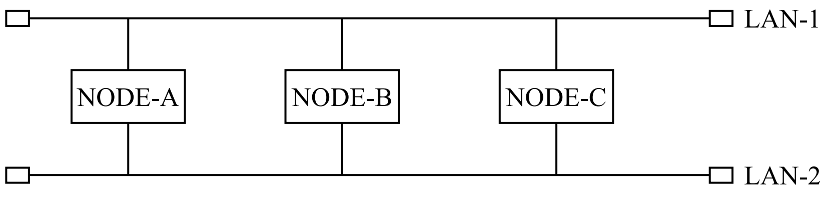
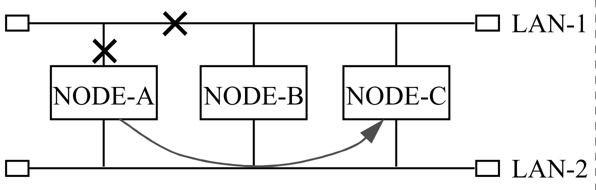
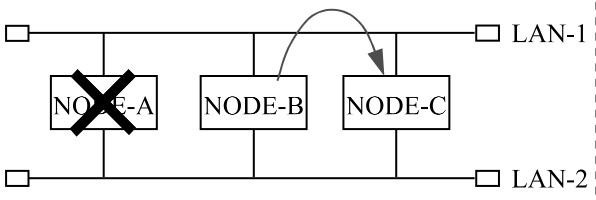
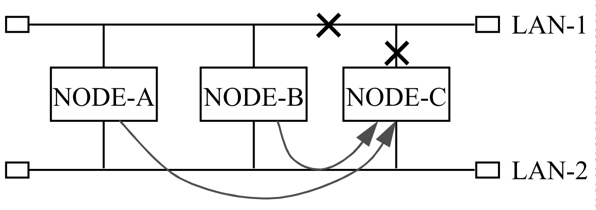


Figure 2-44　Example Configuration

Table 2-15　Examples of Failure Monitor

Failure information (information reported by NXACP)	Failure location	
	Failure diagram	Failure diagram
• The status of NODE-A (LAN1) has changed to dead. • The status of NODE-A (LAN2) stays alive (no change notifications). • The status of nodes next to NODE-A (LAN1) stays alive (no change notifications).		• Built-in Ethernet of NODE-A at LAN-1 • LAN transmission line (including bridges and repeaters)
• The status of NODE-A (both LAN1 and LAN2) has changed to dead. • The status of another node (Node-B) stays alive (no change notifications).		• NODE-A
• The status of NODE-A and B (LAN1) has changed to dead. All nodes have changed to dead at LAN1. • The status of NODE-A and B (LAN2) stays alive (no change notifications).		• Built-in Ethernet of NODE-C at LAN-1 • LAN transmission line (including bridges and repeaters) * You can determine whether the failure location is in the LAN transmission line or in the Built-in Ethernet using failure notifications from the Built-in Ethernet.

The failure information here means the information notified to NODE-C.

(2) Buffer status report

A buffer means message storage memory used for sending and receiving messages. For each data field, buffers for transmission and buffers for reception are managed separately. By managing buffers separately in this manner, you can limit the area of buffers used up by an erroneous user task, and minimize the disturbance on other business tasks by the error.

The number of cases in each buffer is specified by a user at system construction.

In NXACP, if one of the below buffers overflows, or if the usage ratio goes over (or under) a threshold, the event is logged, and the log data is reported to a user every one second. The following shows the timing of buffer status reports.

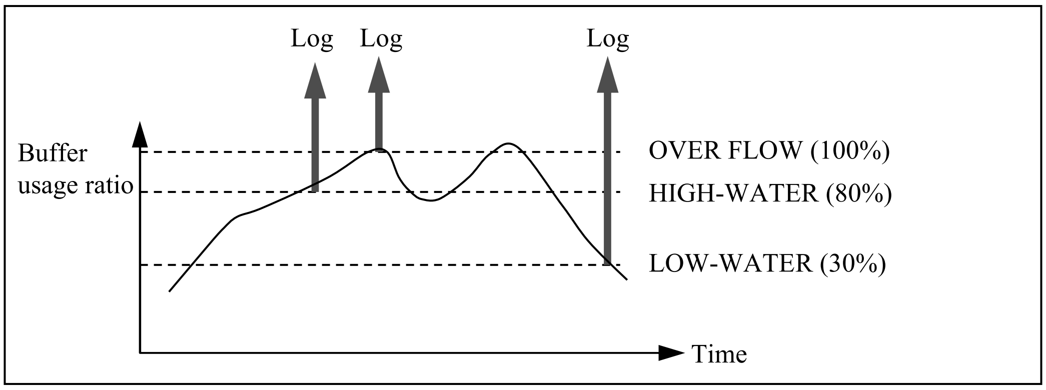


Figure 2-45 Buffer Status Output Timing

[Condition]

Overflow: Logged if allocating space in the buffer fails for the first time after the usage ratio goes over LOW-WATER.

HIGH-WATER: Logged if the usage ratio goes over HIGH-WATER for the first time after the usage ratio goes over LOW-WATER.

LOW-WATER: Logged if the usage ratio goes under LOW-WATER for the first time after HIGH-WATER, or Overflow occurs.

[Detection timing]

Send buffer: Detected when a message transmission request is received from a user and space is attempted to be allocated in the buffer.

Receive buffer: Detected when a message arrives from the network, and space is attempted to be allocated in the receive buffer.

[Action of NXACP when the condition is met]

Even when the condition for HIGH-WATER or LOW-WATER is met, NXACP operation does not change. However, when an overflow occurs, NXACP takes the following actions.

Send buffer: All transmission requests from user tasks are returned with an error because space cannot be allocated in the buffer.

Receive buffer: The process for receiving messages from the network skips reading messages. As a result, messages that arrive during receive buffer overflow are discarded.

[User action]

Reduce the traffic, or increase the buffer capacity.

[Note] Threshold values for HIGH-WATER and LOW-WATER can be changed in the construction information.

(3) Protocol error

A protocol error means an error in an NX header added to each message when sent. This error may occur due to construction errors in send/receive nodes.

When this error is detected, NXACP reports the detected header information (as it is) to a user.

[Condition]

Notified if an error is detected in the header information.

[Detection timing]

Detected in the header check routine launched when a message is received from the network.

[Action of NXACP when the condition is met]

The message with the error is discarded by NXACP after the error is reported to EAS.

[User action]

Check and correct the construction information of the sender node.

(4) Socket failure

Only hardware failures are reported to EAS by RCTLNET, and socket-interface-related errors are reported with detailed information back to the caller of the socket macro. NXACP reports this detailed information to EAS.

At the moment, this error is caused only by an error in the network address definition in the remote data field.

[Condition]

Notified if the socket macro returns with an error (sendto(): ENETUNREACH).

[Detection timing]

Detected by the return code from the socket macro (sendto()) during the online process.

[Action of NXACP when the condition is met]

The transmission request that generated this error is canceled.

[User action]

Check and correct the construction information (network address in the remote data field).

(5) Transfer area duplication error

This error is notified if a write area is defined by multiple nodes for the same address in the transfer memory. A user must specify whether to enable the duplication check at initialization.

[Condition]

Notified if multiple nodes write to the same address in the transfer memory.

[Detection timing]

Detected when software transfer data sent by another node is received. Note that this error is checked after (not before) the transmission/reception has started, or the reception process has started for the software transfer of the local node.

[Action of NXACP when the condition is met]

If duplication is detected for the data received from software transfer, the data is invalidated and not copied to the transfer memory.

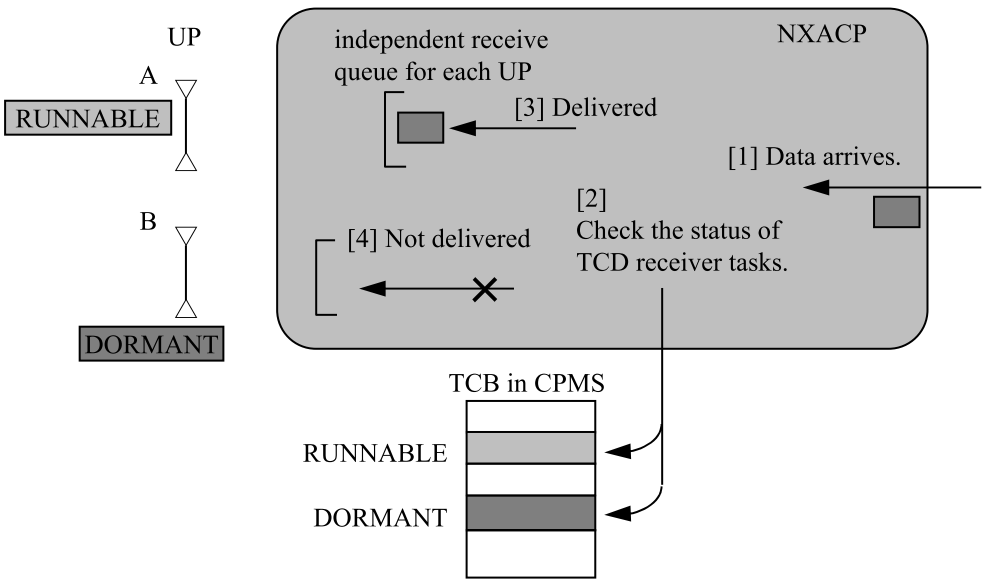
[User action]

Check and reconfigure how the software transfer memory is divided into areas.

5.2 User Task Management Function

Space in the receive buffer is allocated when data is received from the network and released when a user task issues nx_get() and receives the data. That means, if a user task does not issue nx_get(), the receive buffer will overflow.
To prevent receive buffer overflow, NXACP monitors the state of a user task, and if the task is in a state that does not allow the task to call nx_get(), NXACP discards the messages waiting to be received by and that arrive for the task.

(1) Check at message arrival
When a message arrives, NXACP checks the state of each task that accepts the transaction code of the message, and if the state of a task is either DORMANT, IDLE, or NON-EXIST, the message is not delivered to the task. If the state of all such tasks is one of the states above, the message is discarded.



The message is delivered to Task A in the RUNNABLE state (waiting for a message in nx_get() or processing message reception), but not delivered to Task B in the DORMANT state.

Figure 2-46 Task State Check at Message Arrival

(2) Check at transition to DORMANT
When a user task transitions to DORMANT, NXACP checks undelivered messages for the user task and purges any undelivered messages in the receive queue. In addition, a message is discarded when all user tasks that are to accept the message have already completed reception or have entered the DORMANT state.

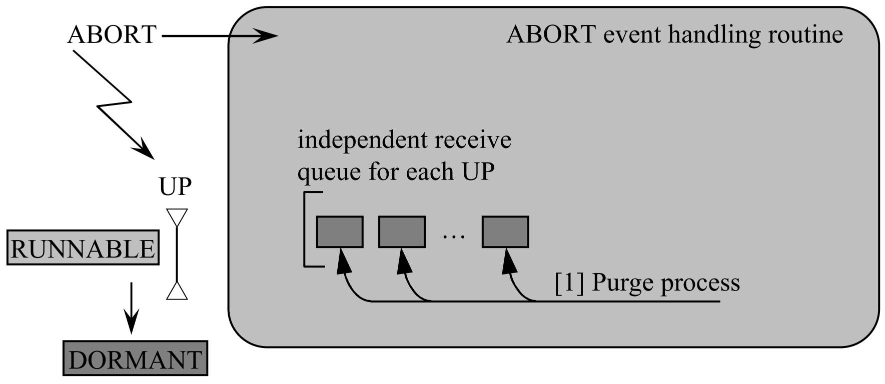

ABORT
ABORT event handling routine
UP
RUNNABLE
DORMANT
independent receive queue for each UP
…
[1] Purge process

Figure 2-47 Post-ABORT Processing

Purging messages in the queue as described above is not executed when a user task transitions to IDLE. The rationale behind this is as follows.

- Normally, a user task that calls nx_get() is constructed as a daemon type program structure, and consequently, does not transition to the IDLE state.
- If this processing is built into the system built-in subroutine for EXIT, the load increases significantly.
- When the next message arrives, the queue will be purged at the timing described in "(1) Check at message arrival".

CHAPTER 6 OPERATION MANAGEMENT FUNCTION

The following figure shows the operation management steps of NXACP. This chapter explains the steps after “Starting NXACP”.

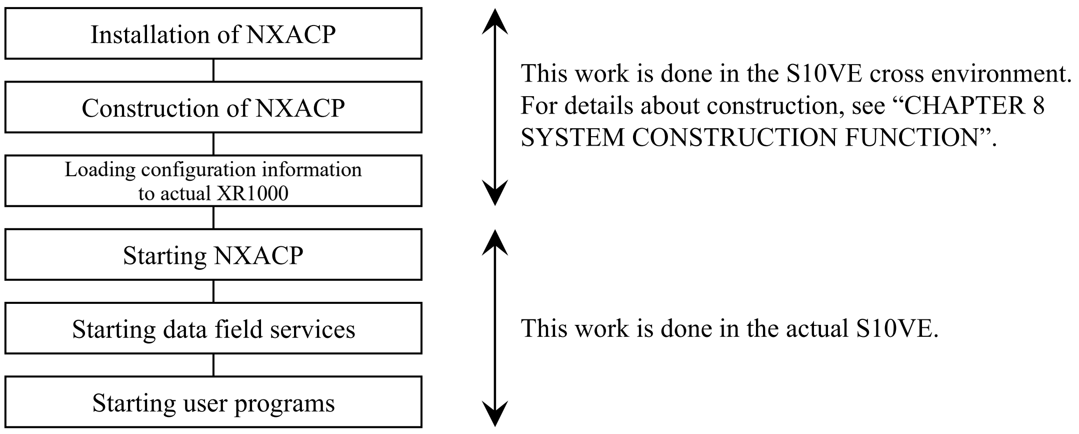

Installation of NXACP
Construction of NXACP
Loading configuration information to actual XR1000
Starting NXACP
Starting data field services
Starting user programs
This work is done in the S10VE cross environment. For details about construction, see “CHAPTER 8 SYSTEM CONSTRUCTION FUNCTION”.
This work is done in the actual S10VE.

Figure 2-48 NXACP Operation Management

The following operation management work is done in the actual S10VE.
• Starting and stopping NXACP
• Starting and stopping a service for each data field
• Removing and inserting a network module
At the step “Starting NXACP”, you must start NXACP (main program) before starting data field services. Along with starting NXACP (main program), when you start the system, perform initialization, and then start data fields just once before starting user programs.
At the step “Starting data field services”, start the transmission and reception process for each data field. The data fields you defined at system construction will start receiving a request for the transmission and reception process when you start the data field services. Issue a service start request to each of the data fields you defined.
You can stop NXACP and data field services as necessary.

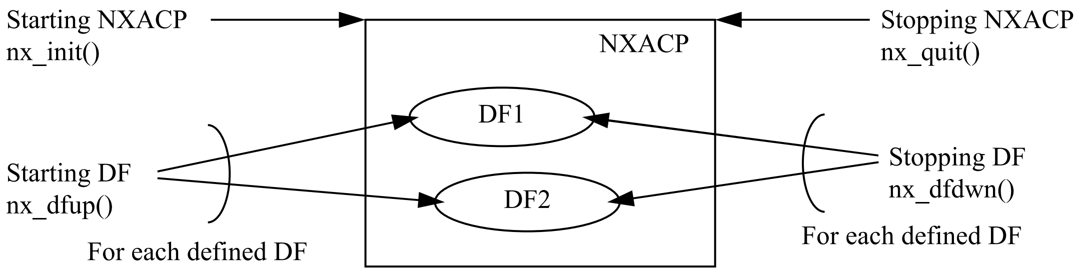


Figure 2-49 Operation Management of NXACP and Data Fields

6.1 Operation Management of NXACP and Data Fields

To start and stop NXACP, use the nx_init() and nx_quit() macros respectively.
After you issue the nx_init() macro, you can start using the data field. Note that you cannot send data to and receive data from data fields by issuing the nx_init() macro alone; you must also issue the nx_dfup() macro for each data field after you issue the nx_init() macro.
When you issue the nx_quit() macro, all NXACP services are stopped. All running data fields are also stopped, and consequently, the message transmission/reception for all data fields is stopped. If you want to stop each data field individually, use the nx_dfdwn() macro.

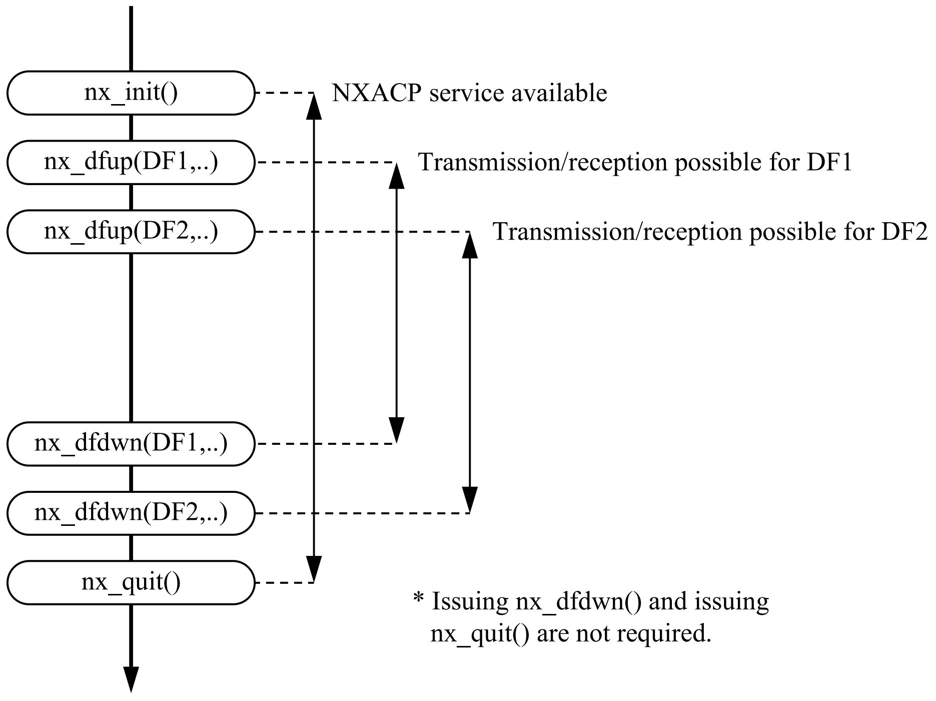

nx_init()
NXACP service available
nx_dfup(DF1,..)
Transmission/reception possible for DF1
nx_dfup(DF2,..)
Transmission/reception possible for DF2
nx_dfdwn(DF1,..)
nx_dfdwn(DF2,..)
nx_quit()
* Issuing nx_dfdwn() and issuing nx_quit() are not required.

Figure 2-50 Macros and Operation Management

6.1.1 Starting data fields

In order for a user task to send and receive messages through a data field, you must start the data field by using nx_dfup().
NXACP supports the following three types of data fields.
• Data field for local node communication
• Local data field
• Remote data field
Regardless of the type of a data field, use nx_dfup() for starting the data field.
When nx_dfup() is issued for a data field, the following routines are executed in the data field.

Table 2-16 Data Field Startup Routines

	DF for local node communication	Local DF	Remote DF
Preparing for message transmission to the DF	Y	Y	Y (*)
Starting message reception from the DF	Y	Y	Y (*)
Starting alive signal transmission to the DF	–	Y	–

Y: Executed –: Not executed
(*) You can send data to and receive data from a remote data field only when the local data field is up and running.

To transmit messages between user tasks within a local node, the data field for local node communication is used, and you must start the data field for local node communication (DF#=0).
If you want to use a remote data field, you must first start the local data field (called a source data field) through which the remote data field is connected, and then start the remote data field. If the local data field has not been started, a message from the remote data field will be discarded by NXACP.
In addition, a transmission request to the data field is treated as an error, and processing of the transmission request is aborted.

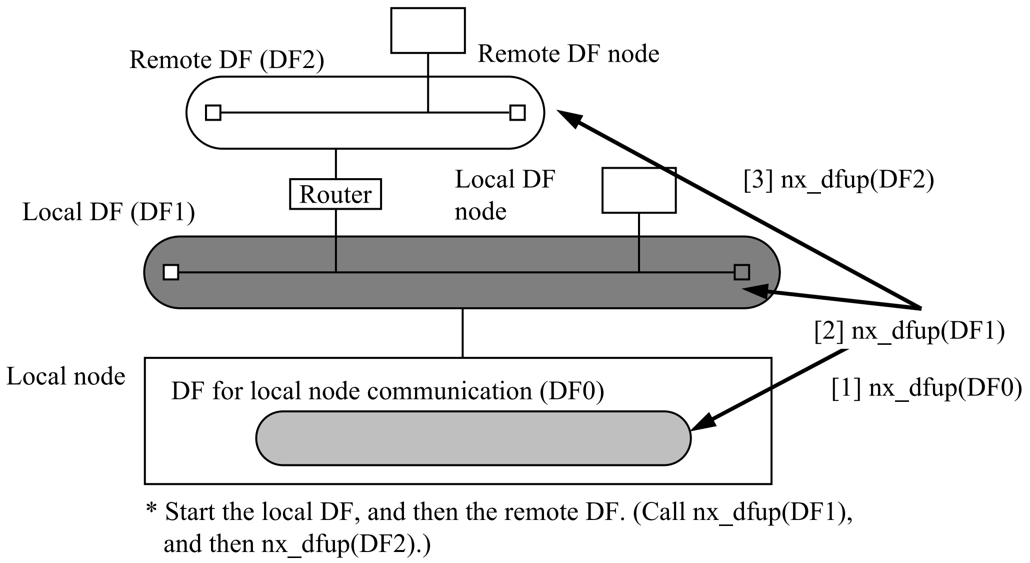

Remote DF (DF2)
Remote DF node
Router
Local DF node
[3] nx_dfup(DF2)
Local DF (DF1)
[2] nx_dfup(DF1)
Local node
DF for local node communication (DF0)
[1] nx_dfup(DF0)
* Start the local DF, and then the remote DF. (Call nx_dfup(DF1), and then nx_dfup(DF2).)

Figure 2-51 Order of Initialization for Data Fields

[Caution] Handling a failure in starting a data field

When a data field is constructed as a single LAN and an error such as a port generation/configuration error is detected, starting a data field fails and nx_dfup() returns an error. When the error is detected, the startup process for the data field is aborted, the network-related settings are reset to the settings used before nx_dfup() was issued (the ports generated so far are deleted), the data field is put back to the stopped state, and then control is returned to a user.

The error is neither reported to EAS nor recorded in the error log. If a failure is generated in Built-in Ethernet, RCTLNET reports the failure to EAS. You can analyze the location of the failure based on the return code of nx_dfup() and the error log information.

When a data field is constructed as duplexed LANs and a configuration error is generated for both LANs, the startup process for the data field is aborted, the network-related settings are reset to the settings used before nx_dfup() was issued (the ports generated so far are deleted), the data field is put back to the stopped state, and then control is returned to a user.

Even when nx_dfup() returns an error, other running data fields are not affected.

6.1.2 Stopping data fields

In order to stop sending and receiving messages through a data field while the data field is running online, you must stop the data field.
If you want to change the mode of a node and the mode of messages to be received while keeping user programs running, stop the data field temporarily, and then restart the data field.
Regardless of the type of data field, use nx_dfdwn() to stop it.
Note that stopping NXACP by using nx_quit() stops all data fields together at the same time. An alive signal for scheduled SHUTDOWN is not sent when NXACP is stopped by using the construction information loading command to update construction data. If you want to notify other nodes of scheduled SHUTDOWN, stop NXACP by using nx_quit().
When nx_dfdwn() is issued for a data field, the following shutdown routines are executed in the data field.

Table 2-17 Data Field Shutdown Routines

	DF for local node communication	Local DF	Remote DF
Stopping message transmission to the DF	Y	Y	Y
Stopping message reception from the DF	Y	Y	Y
Stopping alive signal transmission to the DF	–	Y (*)	–

Y: Executed –: Not executed
(*) Stops alive signal transmission after an alive signal in the scheduled shutdown mode is sent a couple of times.

There are no restrictions on the order of data fields when you stop them.
If you stop a local data field, the remote data fields that are connected through the local data field are also stopped. Then a user does not have to issue nx_dfdwn() to these remote data fields.
Note that, when you restart the local data field, these remote data fields are not automatically restarted. If you want to restart these remote data fields, you must issue nx_dfup() to the remote data fields.
When nx_dfdwn() is issued, all messages waiting to be processed are discarded. The sending and receiving messages for any other data field that has not received a shutdown request are processed normally without being affected. As an exception, a sending message already passed to RCTLNET at the timing of calling nx_dfdwn() will be transmitted to the network without being discarded.
If the data field specified in nx_dfdwn() is a local data field, messages reporting scheduled shutdown are sent (which takes about 10 seconds). As a result, when you issue nx_dfdwn() successively to stop multiple data fields, it takes about 10 seconds before starting the shutdown process for the second and subsequent data fields.
You may want to use nx_quit() when you stop multiple data fields at once. Also when nx_quit() is used, messages reporting scheduled shutdown are sent to the local data fields that have been running. Note that NXACP is also stopped when nx_quit() is used, and you have to reissue nx_init() when you want to use NXACP again.

[Caution]

- Handling a failure in stopping a data field
  The nx_quit() and nx_dfup() macros do not return with an error when an error other than a parameter check error is detected.
  These macros just stop data fields and then return control to a user. All errors in nx_quit() and nx_dfdwn(), other than parameter errors and errors associated with stop requests to data fields that are not running, must be caused by failures in Built-in Ethernet. A failure generated in Built-in Ethernet is reported to EAS by RCTLNET, allowing you to locate the failure. In addition, the details of the failure can be analyzed based on the error log information.
- Effect on remote data fields when stopping a local data field
  If you stop a local data field, the message transmission/reception process for remote data fields that are connected through the local data field is also stopped.

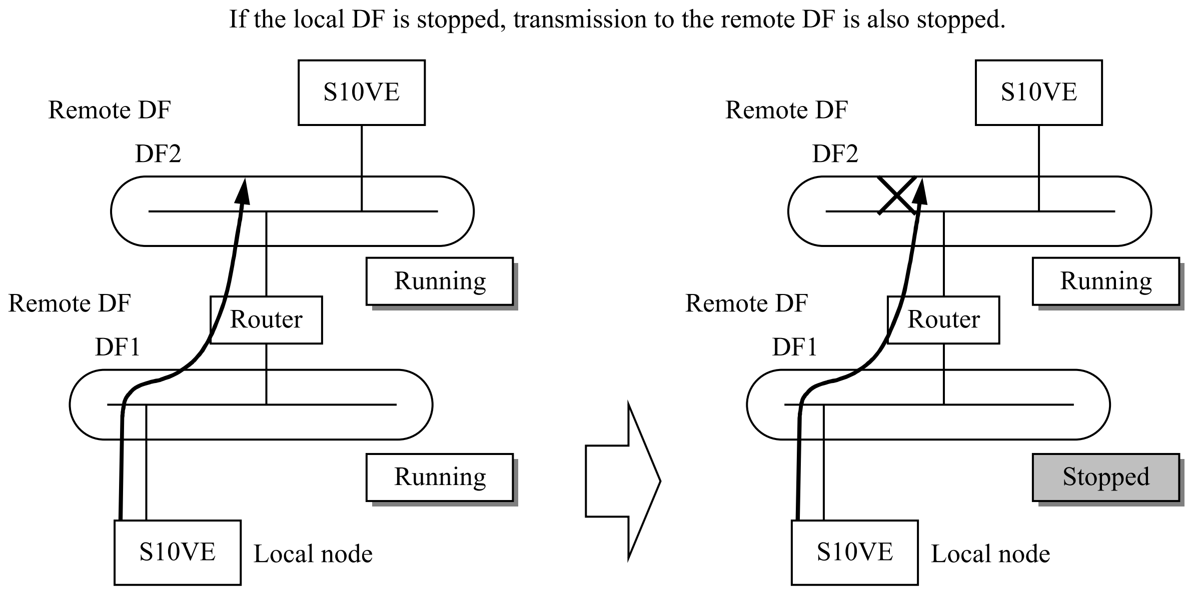


Figure 2-52 Effect on Remote Data Fields (1)

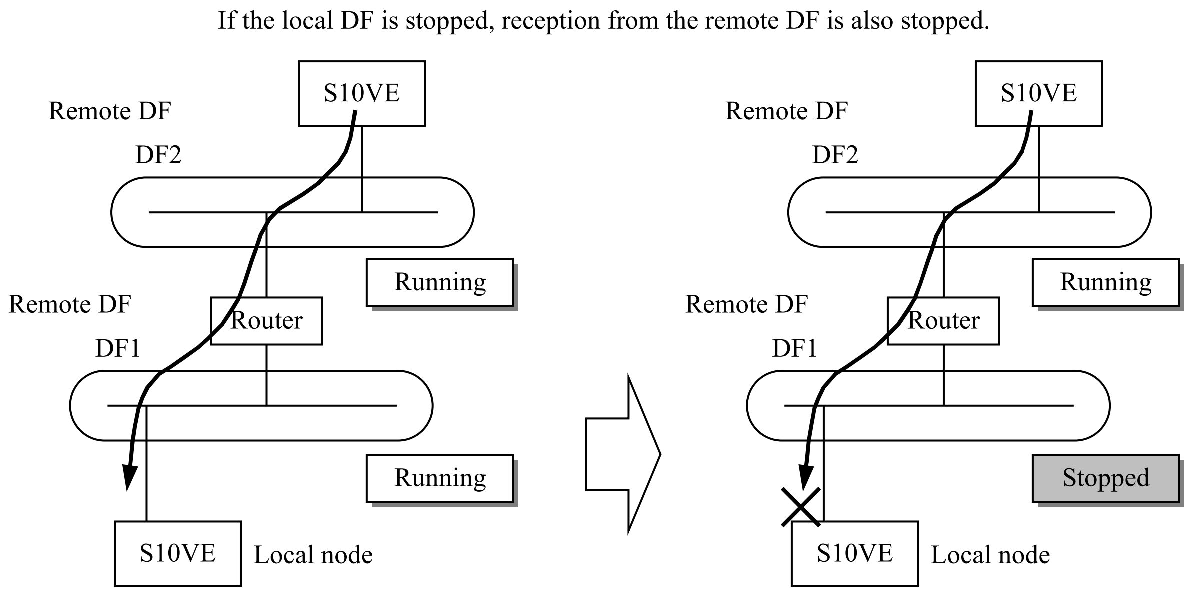


Figure 2-53 Effect on Remote Data Fields (2)

6.1.3 Setting and updating a mode

You can set the node mode of a data field and the message mode of messages to be received on a per-data-field basis. Specify which modes to be used for a data field in a parameter when nx_dfup() is issued. The modes specified in the parameter are valid as long as the data field is running, but you cannot change the modes while the data field is running.
To change the modes, you must stop the data field by using nx_dfdwn(), and then restart the data field by using nx_dfdup() with new modes specified.
When you issue nx_dfdwn(), you must specify both a node mode and the message mode of messages to be received. You must stop the data field even when you change only one of the modes.
The purpose of using two macros, nx_dfdwn() and nx_dfup(), to change the modes is to allow a variety of user task settings associated with the mode change to be configured easily. Actions associated with the mode change, such as initialization of tables and user tasks, can be executed between calling nx_dfdwn() and calling nx_dfup(). During this period, message reception at the data field from the network is stopped, and you can prepare for mode change without concern of buffer overflow. In addition, all sending and receiving messages before mode change waiting to be serviced are discarded, and messages that arrive before nx_dfup() is issued are also discarded. For this reason, you can resume message transmission/reception with the data field in the new modes without concern of status before nx_dfup() is issued.

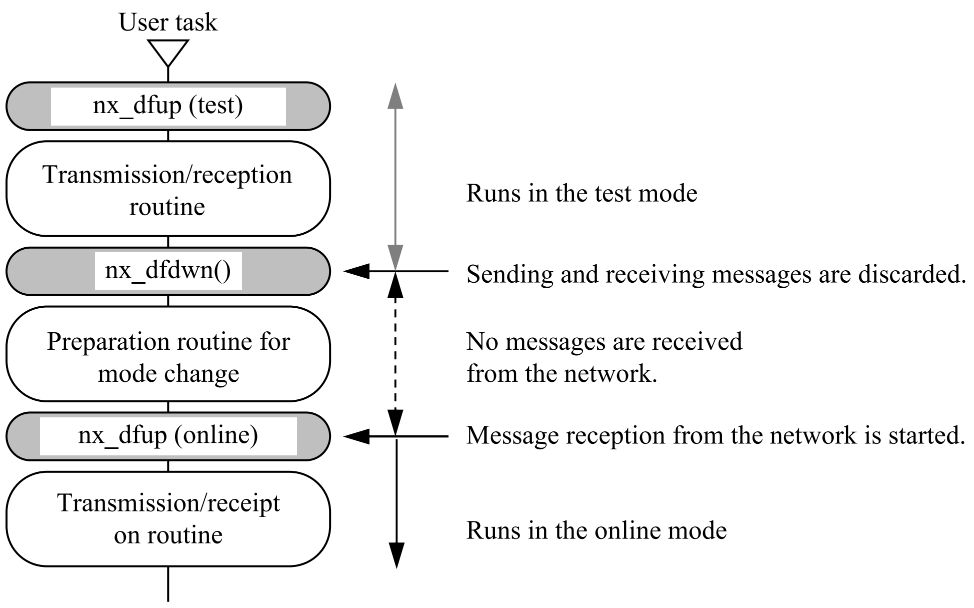

User task
nx_dfup (test)
Transmission/reception routine
nx_dfdwn()
Preparation routine for mode change
nx_dfup (online)
Transmission/receipt on routine
Runs in the test mode
Sending and receiving messages are discarded.
No messages are received from the network.
Message reception from the network is started.
Runs in the online mode

Figure 2-54 Mode Change Procedure

[Caution]
- The setting for the node mode is invalid for the local node data field.
- The setting for the message mode of messages to be received is valid only for local data fields.

Table 2-18 Valid Range of Mode Settings

Data field type	Setting for node mode	Setting for message mode of receiving messages
Data field for local node communication	N	N
Local data field	Y	Y
Remote data field	Y	N

Y: Valid, N: Invalid

6.1.4 Updating construction information

NXACP must be stopped before you update its construction information.
To reload the construction information from the POC, use the tblldnxsv command to load to the S10VE site.
While loading, NXACP is automatically stopped. When the loading process is completed, restart NXACP by using nx_init().
The tblldnxsv command forcibly stops NXACP, and consequently, alive signals for scheduled shutdown are not sent. If you need to send alive signals for scheduled shutdown to other nodes, a user must issue nx_quit() before starting reloading.

CHAPTER 7 NETWORK-SHARED MEMORY FUNCTION

The network-shared memory function (hereinafter denoted as "transfer memory function") periodically copies the data written to transfer memory to the read areas in the other nodes. The same address can be written by only one node but read by multiple nodes. The node that writes to the address is fixed statically by the system.
There are two types of transfers: hardware transfer and software transfer. Both types use the same user interface, and a user can use the transfer memory function without taking care of communication methods.
Note that S10VE does not support hardware transfer.

7.1 Terminology

The following definition of terms is used for the transfer memory function.

(1) Transfer memory
Transfer memory consists of transfer areas. A transfer area consists of 64-byte blocks.
The size of a transfer area depends on the transfer type.

(2) Software transfer
The global area is used to implement transfer memory, and multicast communication provided by NXACP is in charge of actual transfer. The size of a transfer area is variable, and can be specified up to 16384 blocks.
Note that the maximum number of blocks is user-specified when transfer memory is initialized.
The transfer cycle is in milliseconds.
There are two types of software transfers: periodic transmission type (data is transmitted at each transfer cycle), and split transmission type (data is split and transmitted within the transfer cycle at fixed intervals).
For periodic transmission, you can enable mutual exclusion for a transmitting message.
However, for split transmission, data is split when transmitted, and you cannot enable mutual exclusion.

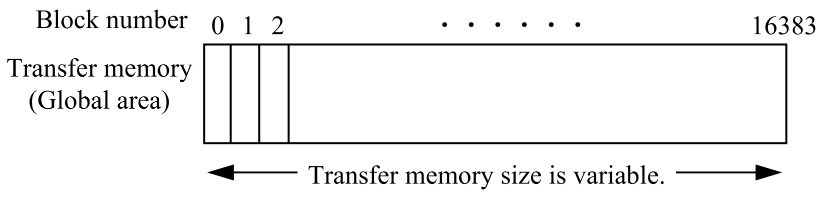


Figure 2-55 Software Transfer Memory

(3) Transfer memory identifier (Transfer Memory ID)

You can define multiple transfer memory areas for each data field. Each transfer memory area is identified by a transfer memory identifier (TMID).
For each data field, four TMIDs can be defined for software transfers.
When accessing transfer memory, a user must specify a TMID.

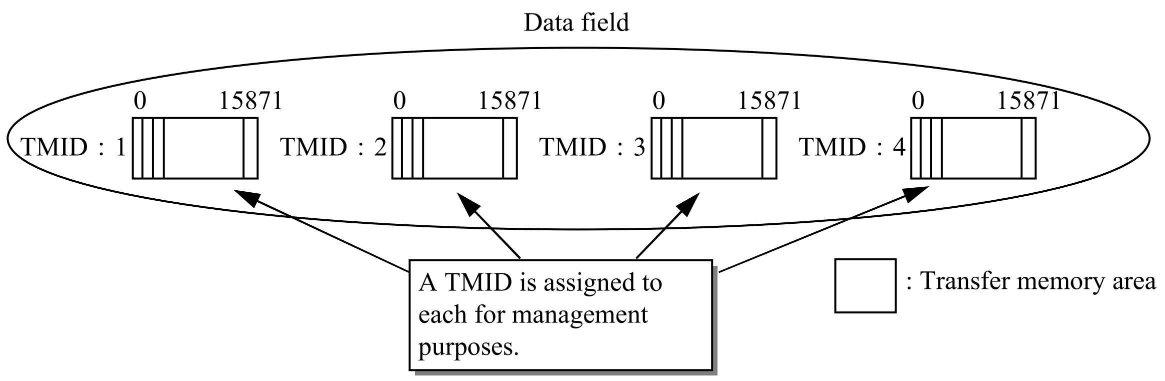


Figure 2-56 Transfer Memory Identifiers (TMID)

(4) Write area/Read area

A transfer memory area is divided into write areas and read areas.
Data written to a write area by the local node is transmitted to the data field. A write area is defined when transfer memory is initialized. Up to 9 cases per TMID can be defined as a write area.
Data received from other nodes can be read from the read area. You do not have to define a read area explicitly. Transfer memory area not used for write areas is used for read areas.
Note that “case” here means consecutive blocks.

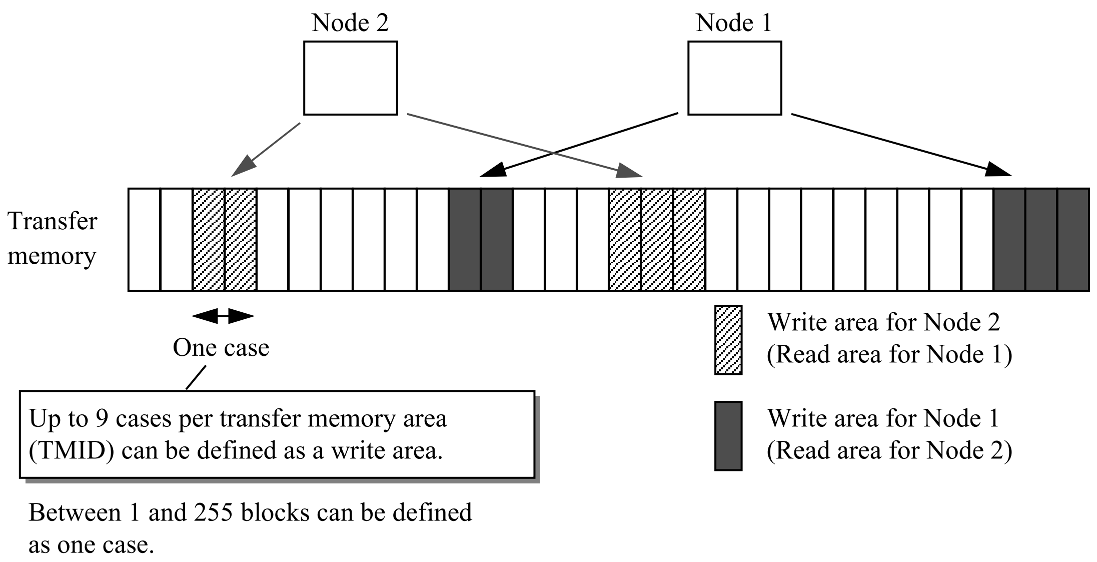

Node 2
Node 1
Transfer memory
One case
Up to 9 cases per transfer memory area (TMID) can be defined as a write area.
Between 1 and 255 blocks can be defined as one case.
Write area for Node 2 (Read area for Node 1)
Write area for Node 1 (Read area for Node 2)

Figure 2-57 Write/Read Area

7.2 Specifications of Transfer Memory

(1) Identifier and unit of management for transfer memory

Transfer memory is a set of transfer memory areas defined for each data field. Each transfer memory area has a transfer memory identifier (TMID) unique in each data field. The following shows the attribute information of transfer memory.

Table 2-19 Attributes for Each Transfer Type

Type / Attribute	Software transfer
Transfer memory areas per data field	4
Size per transfer memory area	64 bytes to 1MB
The number of write areas for one node per transfer memory area	Up to 9
Transfer cycle per write area	(*3)
Duplication check for write areas	Detected at data reception. An error is reported to EAS.
Controller of memory transfer	NXACP
Protection	All readable and writable (*1)
Mutual exclusion	16320 bytes (*2)
Mapping area for transfer memory	Global area CM area (*4)
Encapsulation of duplexed LANs	NXACP

(*1) Data written to a read area will be invalidated because data is periodically written to the area by another node.

(*2) If memory is directly accessed, data consistency for four bytes is guaranteed.
If mutual exclusion for transfer memory is used, data consistency for up to 16320 bytes is guaranteed.
(Fast software transfer does not support mutual exclusion.)

(*3) 100 ms to 86400 s for periodic transmission type software transfer. 10 ms to 86400 s for split transmission type software transfer.

(2) Correspondence between TMIDs and MCGs

Software transfer uses multicast communication provided by NXACP. You must define multicast groups used for software transfer in the construction information of NXACP. To reduce the transfer load, use a dedicated multicast group for each TMID.

<Example of using software transfer>

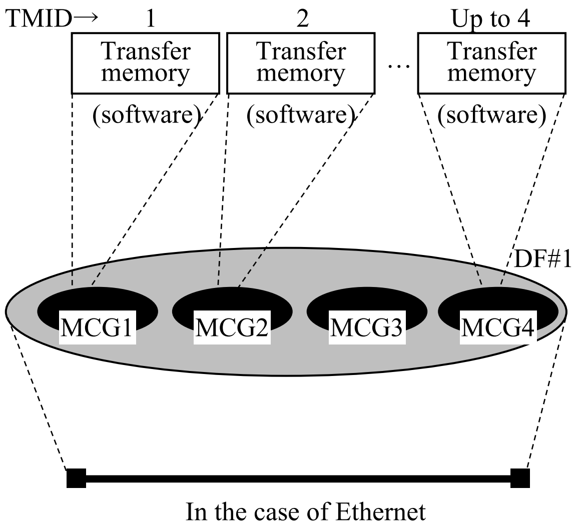


Figure 2-58 Perception of Transfer Memory and Multicast Group

(3) Mutual exclusion

You can enable mutual exclusion for transfer memory as an option of an initialization request.

The mutual exclusion here means mutual exclusion for transfer memory between a read/write request from a user and the transmission/reception routine in NXACP. By using this mutual exclusion, data consistency up to 16320 bytes is guaranteed.

Note that mutual exclusion is executed on a per-TMID basis.

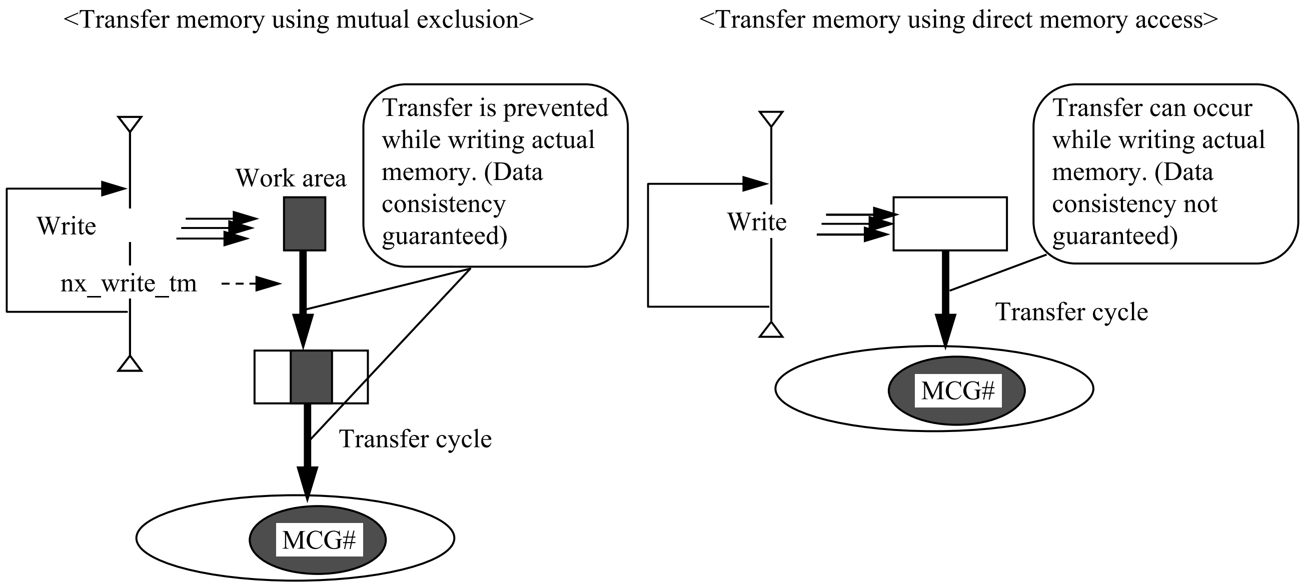

<Transfer memory using mutual exclusion>
<Transfer memory using direct memory access>
Write
nx_write_tm
Work area
Transfer is prevented while writing actual memory. (Data consistency guaranteed)
Transfer cycle
MCG#
Write
Transfer can occur while writing actual memory. (Data consistency not guaranteed)
Transfer cycle
MCG#

Figure 2-59 Mutual Exclusion for Software Transfer

[Caution]

Mutual exclusion is optional. If mutual exclusion is not used, data consistency for up to 4 bytes is guaranteed.

Note that, if mutual exclusion is not used, data half updated by a user can be written to the transfer memory in another node.

For fast software transfer, to guarantee timely periodic update, mutual exclusion is not supported.

(4) Duplication check for write areas

In memory transfer, by defining the write area of each node so that they do not overlap in a transfer area, a user can control write and read timing. NXACP uses the following method for duplication check of the write area of each node.

For software transfer, a user must specify whether to perform a duplication check of the write area as an option of an initialization request.

- Software transfer

  Duplication is checked when data is received from another node. When duplication is detected, the error is reported to EAS. When duplication is detected, received data is discarded. Even after duplication is detected, a write request from a node that caused duplication does not return with an error.

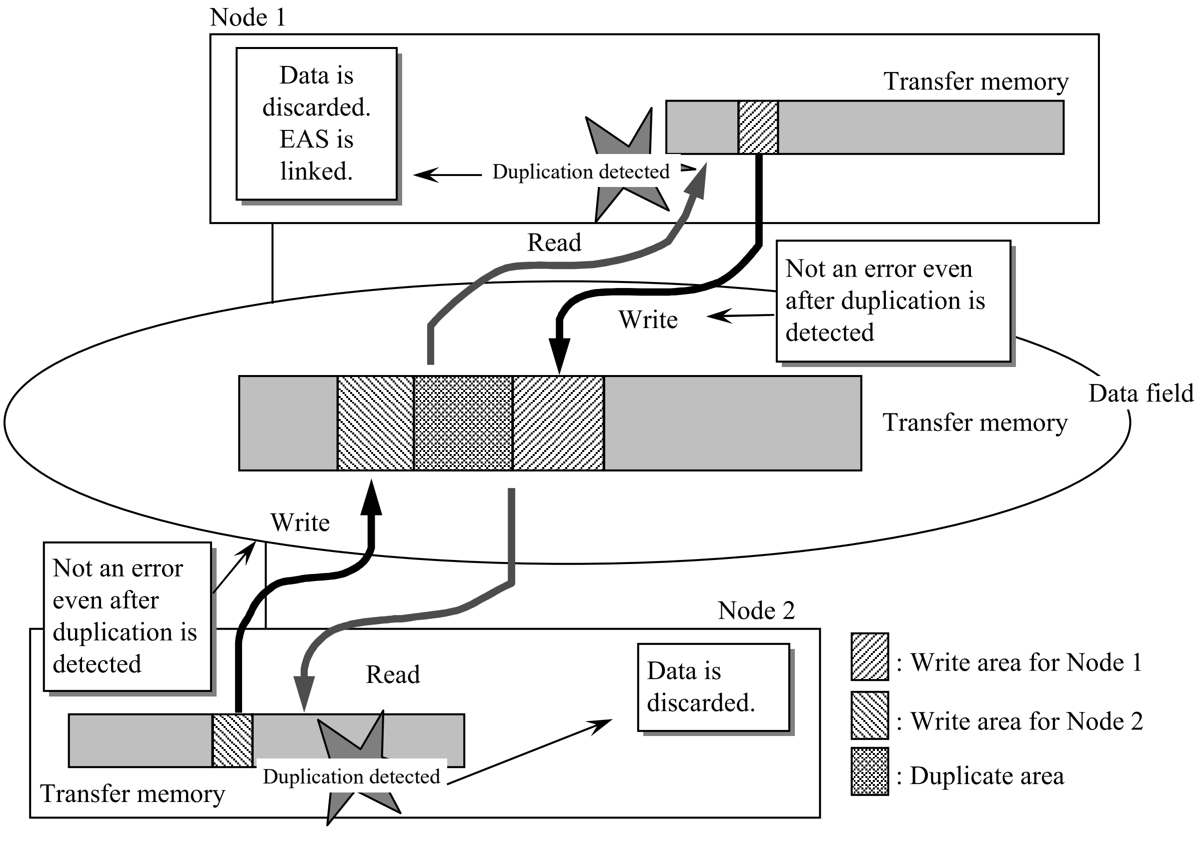


Figure 2-60 Duplication Check for Software Transfer

Duplication between write area cases of the local node is checked at the initialization request, and the result is reported in the return code.

If an error occurs, the initialization request is canceled.

(5) Features of software transfer

There are two types of software transfers: periodic transmission type (data is transmitted at each transfer cycle), and split transmission type (data is split and transmitted within the transfer cycle in order to reduce the network load).

You can select the type in a parameter of the nx_ini_tm macro.

- Periodic transmission type software transfer: Specify TMEM_SOFT for m_type.
- Split transmission type software transfer: Specify TMEM_FAST for m_type.

<Periodic transmission type software transfer>

For messages [1] to [4] (up to 16320 bytes), data consistency is guaranteed by mutual exclusion.

For the minimum transfer cycle, you can specify a value from 100 milliseconds.

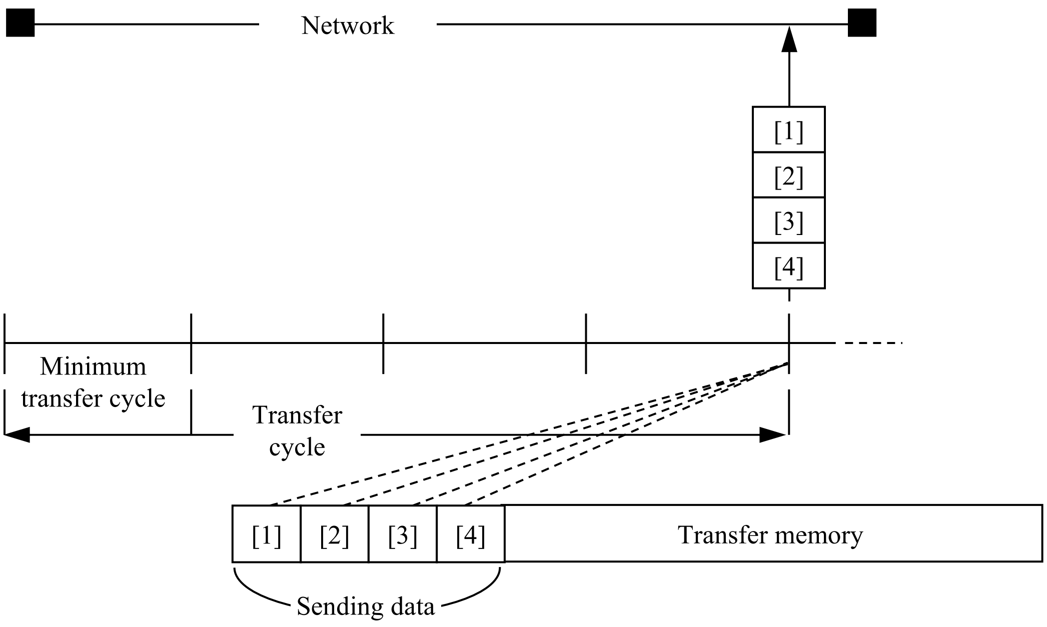


Figure 2-61 Processing of Periodic Transmission Type Software Transfer

<Split transmission type software transfer>

Mutual exclusion is not supported. (Data consistency up to two bytes is guaranteed.)

For the minimum transfer cycle, you can specify a value of 10 milliseconds or longer.

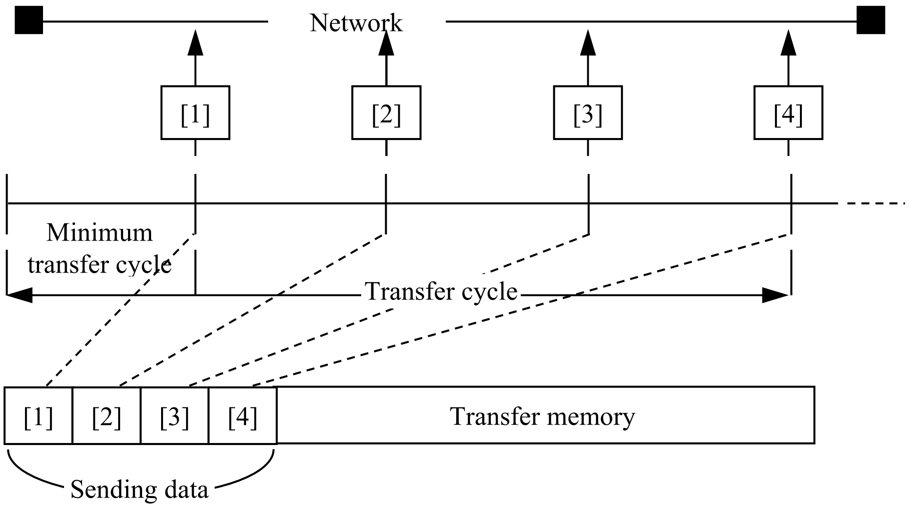


Figure 2-62 Processing of Split Transmission Type Software Transfer

When split transmission type software transfer is used, you can receive both the periodic transmission type and split transmission type.

Table 2-20 List of Transfer Support Models

	Hardware transfer	Periodic transmission type software transfer	Split transmission type software transfer
S10VE	–	Y	Y
S10V	Y	Y	Y
S10mini	–	–	–
Models equipped with NX Dlink (*)	Y	Y	Y

Y: Supported
–: Not supported (cannot be used)
(*) For information about models that support NX Dlink, refer to the *NX Dlink REFERENCE* (manual number RS90-63-X051).
Note that split transmission type software transfer is denoted as "fast software transfer" for NX Dlink.

(6) Setting the minimum transfer cycle
Because the most efficient setting for the minimum transfer cycle is given when data is sent (once) in every cycle, the following setting is recommended.

[Periodic transmission type software transfer]
Transfer data is sent immediately, and consequently, uses the following formula.
(Minimum transfer cycle) = (Transfer cycle)
When you use multiple transfers, set the minimum transfer cycle to the least common multiple of their transfer cycles.

[Split transmission type software transfer]
In fast software transfer, the whole transfer data must be sent within the transfer cycle while 21 blocks of the transfer data is sent at a time at the interval of the minimum transfer cycle. Then the minimum transfer cycle is calculated as follows.

$$\underbrace{\text{(Minimum transfer cycle)}}_{\text{Rounded down to the nearest integer}} = \text{(Transfer cycle)} / (\underbrace{\text{Number of transfer blocks / 21}}_{\text{Rounded up to the nearest integer}})$$

- When you use multiple transfers, set the minimum transfer cycle to the least common multiple of the values calculated by using the formula above for each transfer.
- Due to limited precision, we recommend specifying a multiple of 10 ms (10 ms, 50 ms, 100 ms, etc.) for the minimum transfer cycle, even when you need the finest precision.

7.3 Cautions on Construction

Software transfer is at a higher layer than multicast communication. Even when multicast communication is not used, you must set up the definitions necessary for multicast communication.
The following data in particular must be handled with caution, as described below.

[Number of cases in a buffer (dfN: SBUFCNT and RBUFCNT)]
When you define these values, the number of cases used for software transfer must be taken into account.
For multicast communication, a user must specify the number of cases in a buffer used for storing the messages that cannot be serviced in a timely manner at peak traffic. For software transfer, you must allocate cases necessary for covering the number of write areas in a send buffer, and also allocate cases necessary for covering the number of read areas in a receive buffer.

7.4 Overview of Macros

(1) List of transfer memory macros

When you issue any macro listed below, you must specify a TMID (transfer memory is managed on a per-TMID basis).

Table 2-21 shows an overview of transfer memory macros.

For details, see "PART 3 MACRO SPECIFICATIONS".

Table 2-21 Transfer Macro List

Macro name	Summary of the function	Status that allows calling the macro	Note
nx_init_tm	Initialization of transfer memory	Undefined status	Initializes the transfer memory.
		Stopped status	Reconfigures the attributes and control information of the transfer memory.
nx_ctl_tm	Starting transfer memory	Stopped status or Only reception is ongoing	Starts transmission and reception.
	Starting the reception process of the transfer memory	Stopped status	Starts only reception.
	Stopping transfer memory	Transfer status	Stops transmission and reception.
nx_get_tm	Reading the control information of transfer memory	Stopped or Transfer status	
nx_write_tm	Writing to transfer memory	Stopped or Transfer status	Returns an error if issued in the Undefined status.
nx_read_tm	Reading from transfer memory	Stopped or Transfer status	Returns an error if issued in the Undefined status.

(2) Status transition of transfer memory

Status transitions occur only when nx_init_tm and nx_ctl_tm return normally.

No status transitions occur by calling nx_get_tm, nx_read_tm, or nx_write_tm, regardless of whether the macro returns an error or not.

The following figure shows the status transition of transfer memory.

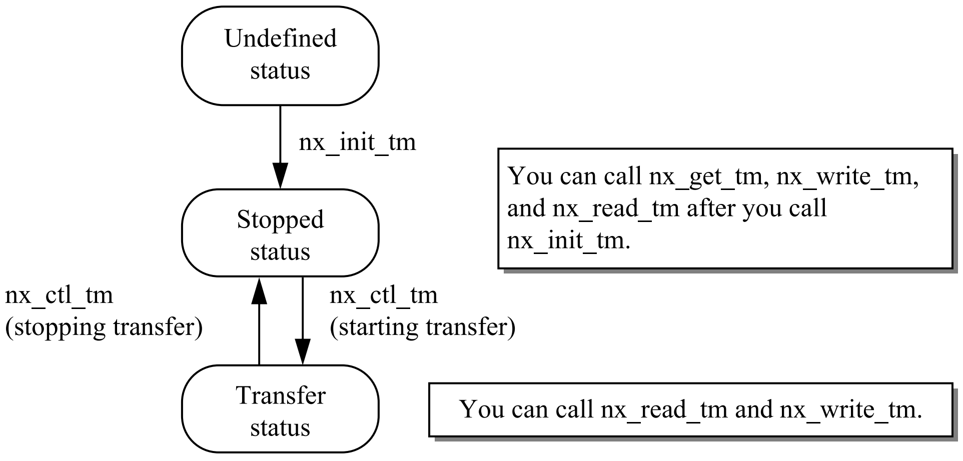

Undefined status
nx_init_tm
Stopped status
You can call nx_get_tm, nx_write_tm, and nx_read_tm after you call nx_init_tm.
nx_ctl_tm (stopping transfer)
nx_ctl_tm (starting transfer)
Transfer status
You can call nx_read_tm and nx_write_tm.

Figure 2-63 Status Transition of Transfer Memory

[Note]
If the site configuration of the S10VE is a multiprocessor type (each processor has its own site), each site has its own status of transfer memory. Consequently, you must change the status of transfer memory in both sites.

(3) Condition on a data field status
The status of a data field is either CLOSE or OPEN.
The transfer memory function can be used only when the status of the data field is OPEN, regardless of whether the type of transfer is software or hardware.

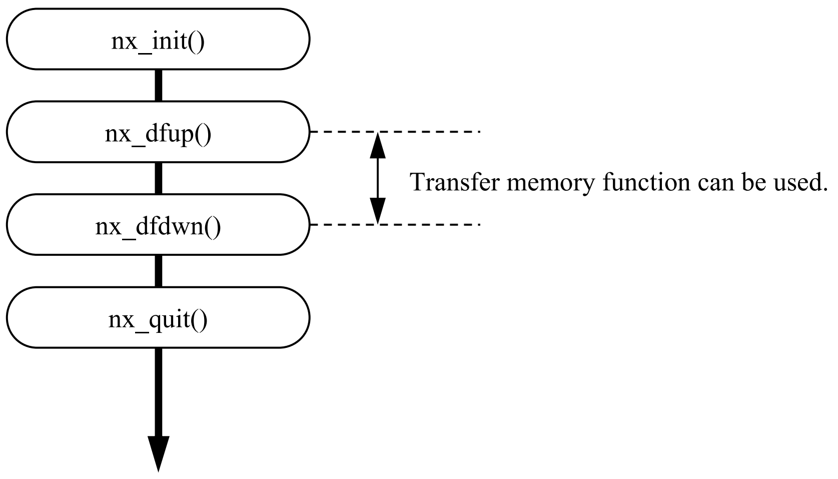


Figure 2-64 Transfer Memory and Data Field Status

CHAPTER 8 SYSTEM CONSTRUCTION FUNCTION

This chapter describes the procedure for installation and system construction of NXACP.
NXACP runs on the task in charge of communication control (hereinafter denoted as NXACP task) and the IRSUBs in charge of processing the user interface (hereinafter denoted as NXACP macros).
The objects of NXACP are provided in a CD-ROM.
To install the main part of NXACP, run an RPDP command on a computer that has the site information of the target S10VE.
Note that, before you use NXACP, you must set up the operation environment by using construction commands.
Set up the environment also on the computer that has the site information of the target S10VE and load the settings (hereinafter denoted as NXACP table) to the GLB area of the S10VE.
The following shows the system construction procedure for NXACP and the name of the command used for each step. Note that the site information must be constructed before you start installation and system construction.

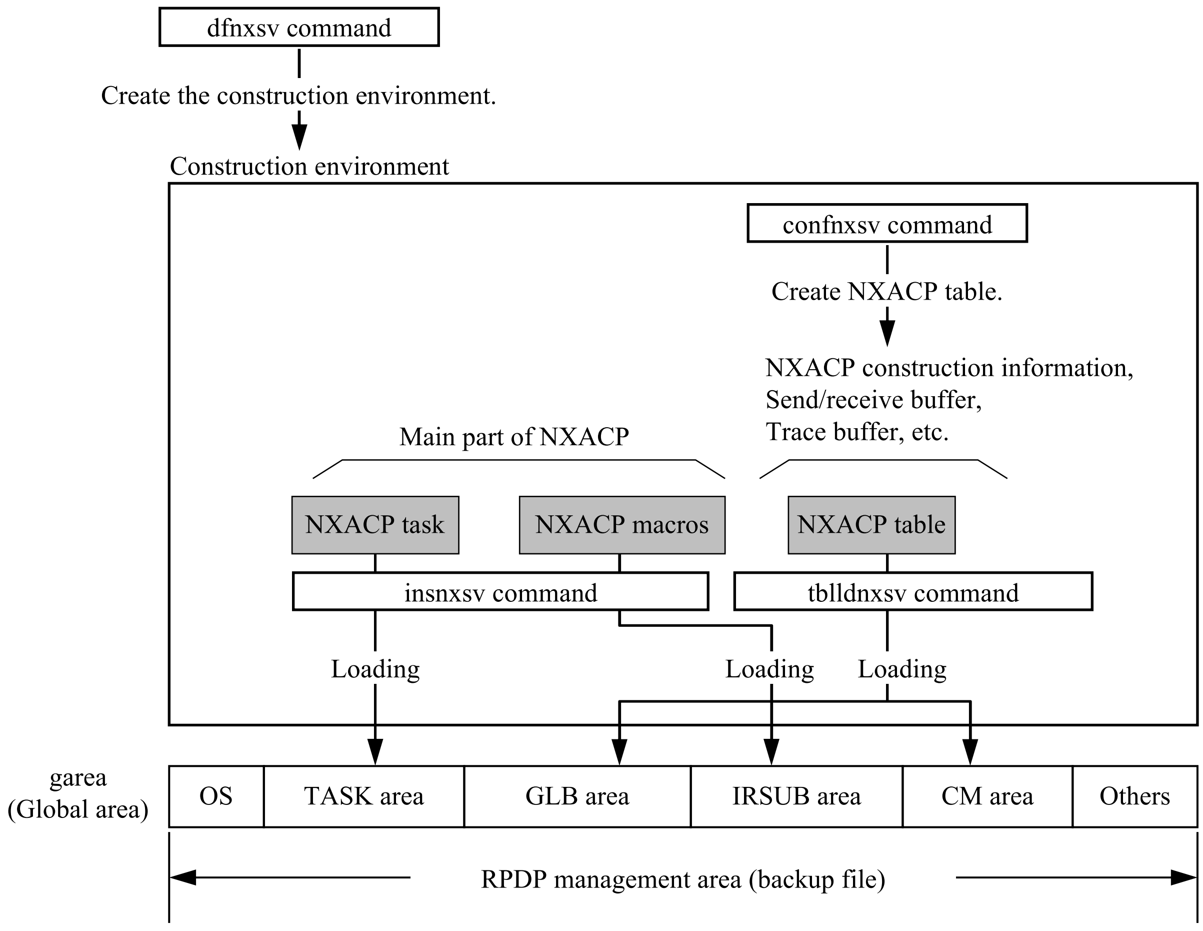


Figure 2-65 Deployment of Resources

[Remark] Each construction command suppresses a signal while running. The tblldnxsv command displays the execution state.

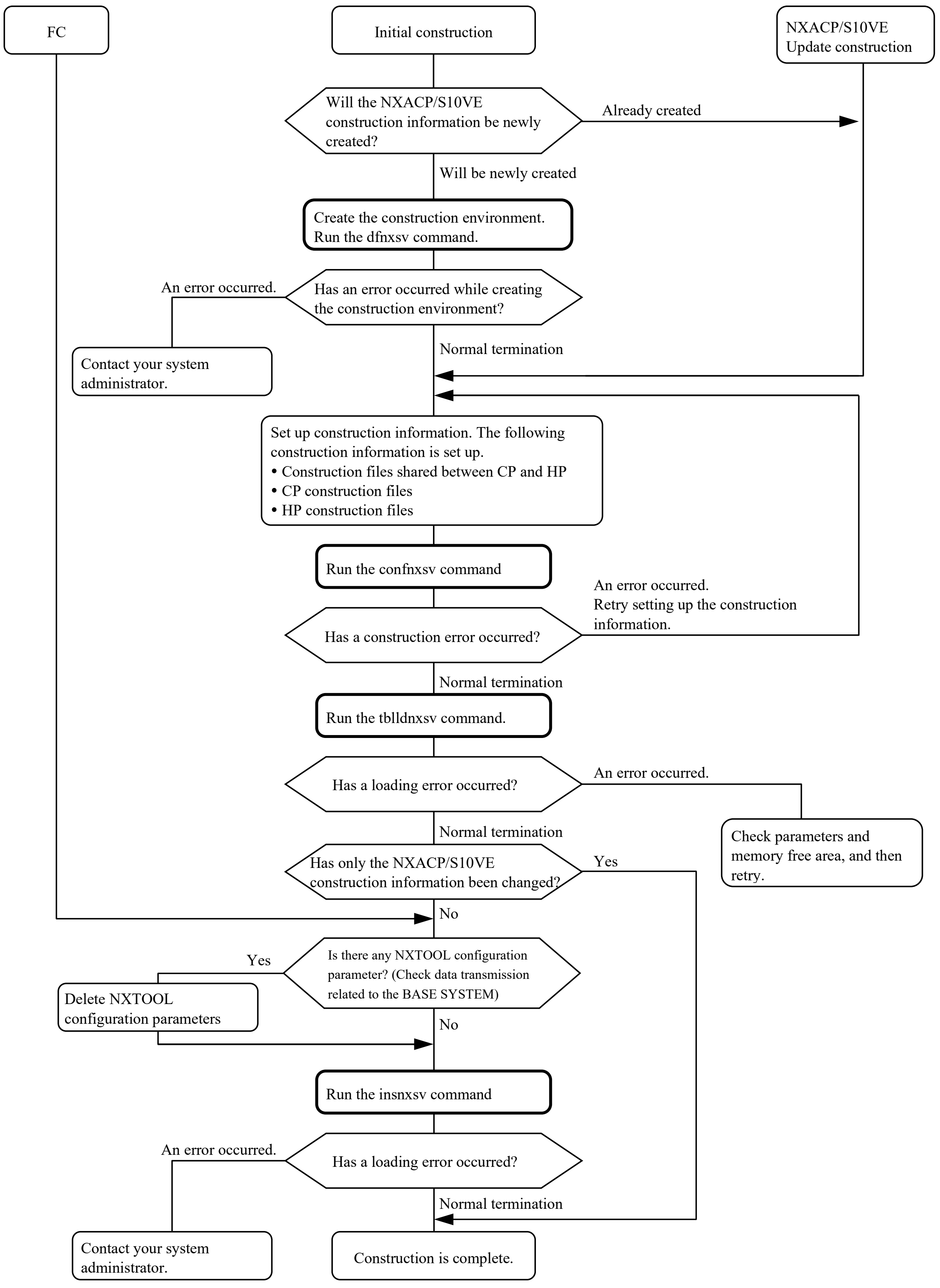

FC
Initial construction
NXACP/S10VE Update construction
Will the NXACP/S10VE construction information be newly created?
Already created
Will be newly created
Create the construction environment. Run the dfnxsv command.
An error occurred.
Has an error occurred while creating the construction environment?
Contact your system administrator.
Normal termination
Set up construction information. The following construction information is set up.
• Construction files shared between CP and HP
• CP construction files
• HP construction files
Run the confnxsv command
An error occurred. Retry setting up the construction information.
Has a construction error occurred?
Normal termination
Run the tblldnxsv command.
Has a loading error occurred?
An error occurred.
Check parameters and memory free area, and then retry.
Normal termination
Has only the NXACP/S10VE construction information been changed?
Yes
No
Is there any NXTOOL configuration parameter? (Check data transmission related to the BASE SYSTEM)
Yes
Delete NXTOOL configuration parameters
No
Run the insnxsv command
An error occurred.
Has a loading error occurred?
Contact your system administrator.
Normal termination
Construction is complete.

Figure 2-66 NXACP Construction Procedure

8.1 Loading the Main Part of NXACP

To load the main part of NXACP (including the NXACP task and NXACP macros) to the backup file under the site information, run the following command.

```
# insnxsv site_name [Enter]
```

Specify the site name of the CP for site_name.
By running this command, the main part of NXACP is loaded to the backup file of the S10VE specified as the site name. To copy (download) the main part of NXACP to the target S10VE, use the remote loading command provided by RPDP.

<Error output on the monitor and its contents>

Parameter count error: The following message is displayed.

```
parameter number out of range
usage:insnxsv site
```

Error generated during execution: If an RPDP-related error is displayed and the execution is aborted, refer to *S10VE Software Manual Operation RPDP for Windows*® (manual number SEE-3-133).

<Output on the monitor during execution and its contents>

If the command is executed without an error, the following messages are output by the command.

```
*** NXACP/S10VE INSTALL START(CPU_name) ***
                      |    ◄──────────  Shows the commands executed on CP.
*** CP(site_name) INSTALL OK !! ***
                      |    ◄──────────  Shows the commands executed on HP.
*** HP(site_name) INSTALL OK !! ***
*** NXACP/S10VE INSTALL END ***
```

[For reference] The following shows the resource allocation of NXACP.

Table 2-22 Resource Allocation

Program name	Task number	IR subroutine number	UL subroutine number	Task level	Storage area name (Capacity)
nx_memac	209	–	–	7	task/nxacpt (288 KB)
nx_cycsnd	210	–	–	7	
(reserve)	211	–	–	–	
(reserve)	212	–	–	–	
nx_operation	213	–	–	4	
nx_snd1	214	–	–	6	
nx_snd2	215	–	–	6	
nx_snd3	216	–	–	6	
nx_snd4	217	–	–	6	
nx_snd5	218	–	–	6	
nx_snd6	219	–	–	6	
nx_htim	220	–	–	5	
nx_ltim	221	–	–	17	
nx_upexe	222	–	–	6	
nx_purcv	223	–	–	6	
(reserve)	224	–	–	–	
nx_init	–	301	–	–	sub/nxacpm (144 KB)
nx_quit	–	302	–	–	
nx_put	–	303	–	–	
nx_get	–	304	–	–	
nx_dfup	–	305	–	–	
nx_dfdwn	–	306	–	–	
nx_init_tm	–	307	–	–	
nx_ctl_tm	–	308	–	–	
nx_get_tm	–	309	–	–	
nx_write_tm	–	310	–	–	
nx_read_tm	–	311	–	–	
nx_trc	–	312	–	–	
nx_cdoff	–	313	–	–	
nx_cdon	–	314	–	–	
nx_puni	–	315	–	–	
(reserve)	–	316 to 332	–	–	
(reserve)	–	–	2	–	
nx_ins	–	–	5 (INS)	–	
nx_exs	–	–	6 (EXS)	–	
nx_abs	–	–	7 (ABS)	–	
nx_ctl	–	–	15 (MODES)	–	

[Note]
Task numbers from 209 to 224 must be reserved for NXACP.
IR subroutine numbers from 301 to 332 are reserved by NXACP.
UL subroutine number ENTRY No.2 is reserved by NXACP.

[Caution]
The task level of any user task that uses NXACP must be lower than the task level of NXACP task.
The task number of any user task that uses NXACP must be set to a value from 1 to 208.

[Warning]
An erroneous operation might occur if parameter information configured with NXTOOLS SYSTEM/S10VE remains.
If the data transmission function of BASE SYSTEM/S10VE was used before the loading operation to configure parameter information that was configured with NXTOOLS SYSTEM/S10VE, and that information (namely "NX (type 4)", "NX (type 5)", or "NX (type 6)") remains, such information must be deleted.
For details on how to start BASE SYSTEM/S10VE and use the data transmission function, see *S10VE User's Manual General Description* (manual number SEE-1-001).

8.2 System Construction and Loading

To use NXACP, you must set up the operation environment for each site.

8.2.1 Initialization

Initialize the definition information of each site by using the command below. Note that the command should be executed only once at the initial construction, after a site is created by the basic system.
Specify the site name of the CP for site_name.

```
# dfnxsv site_name [Enter]
```

<Error output on the monitor and its contents>
Parameter count error: The following message is displayed.

```
parameter number out of range
usage:dfnxsv site
```

<Construction environment>
A site is created for HP and CP separately under the CPU name due to the processor structure of the S10VE, in the same way as the site environment of RPDP. To meet the specifications of RPDP, a unique name (which must differ from that of the R700, R900) is given to a site. If the same name is used, an error is generated during system generation. In addition, the site (CP) name is used as the CPU name.
To allow coexistence with the construction environment of a R700, R900, the structure of the construction environment of the R700, R900 is as follows.

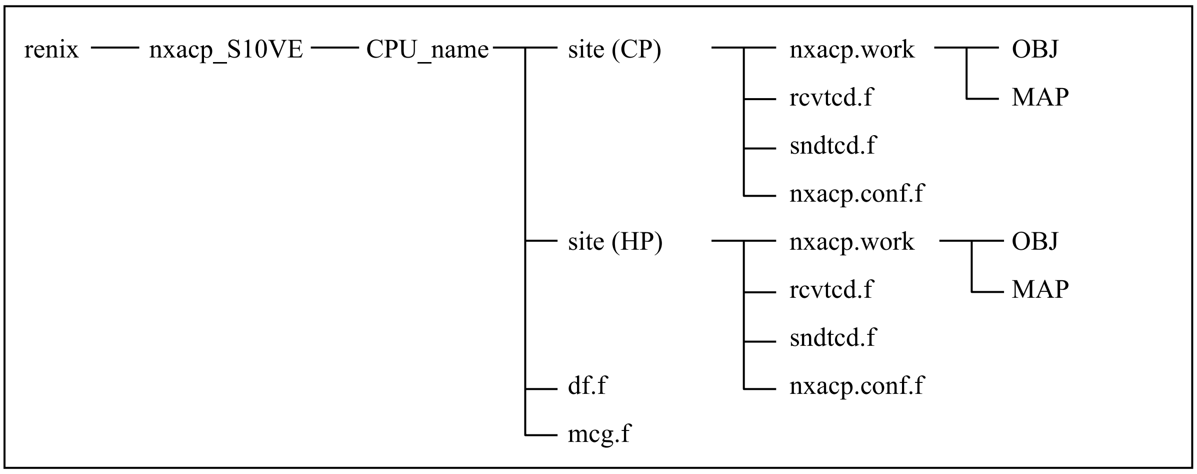


Figure 2-67 Construction Environment

8.2.2 Setting up construction information

When you set up definition information files for each site, change “original file name” of each definition file stored by the dfnxsv command in the directory for the site to “creating file name”, and then set up construction information in each “creating file” by using, for example, an editor command (notepad.exe).

Table 2-23 Configuration File List

Category	Original file name	Creating file name	Items managed by the database
NXACP common definition	nxacp.conf.f	nxacp.conf	Manages the basic settings of NXACP.
DF definition	df.f	dfN	Manages the settings of each data field. Define information such as the LAN address of a node. Specify a data field number from 0 and 255 for N.
MGN definition	mcg.f	mcgN	Manages the online/test UDP destination port numbers of each multicast group number. Specify a data field number from 1 to 255 for N.
TCD definition	rcvtcd.f sndtvd.f	rcvtcdN sndtcdN (*)	Manages the settings of each transaction code. Define the user task number to receive TCD messages for rcvtcdN. Specify a data field number from 0 to 255 for N. Note that when transmission and reception are performed on the HP side, the definition of the HP side must be applied to the CP side, regardless of whether the definition is used on the CP side.

(*) The task numbers of the user tasks that send messages of each TCD are defined in sndtcdN.
Note that this file is optional. Create this file only when you want to use “TCD transmission right check” described later.

Each definition information file is explained below. Before these explanations, how to describe the definitions in a definition file is explained.
Each definition file must be created according to the following rules.

<Creation rules>

- Handling of a definition in multiple lines
  Each definition of information must be included in one line and must not span multiple lines. (You cannot concatenate lines by using “\”.)
- Characters available for definition
  Only one-byte ASCII alphanumeric characters can be used in definitions (numbers must be in decimal notation). Spaces, tabs, and blank lines are skipped.
  If the first letter of a line is “#” (pound sign), the line is interpreted as a comment.
  Signs and two-byte characters cannot be used in definitions.
- Order of definitions
  The order of definitions can be random and does not have to be ascending or descending.

(1) Definition information in the common definition file (nxacp.conf)
The following information is defined in the common definition file.
If a specific definition is not required, specify the value shown in the "If not used" column for the definition, delete the identifier, or insert "#" (pound sign) before the identifier. If the value shown in the "If not used" column for a definition is "–", you cannot delete the definition.

Identifier	Description	Configurable range	If not used
MAXDFCNT	Total number of registered data fields	0 to 16 (*1)	–
TRCCS	Trace case count (Number of cases)	0, 128 to 65535 (multiple of 128)	0
TMSCYC	Software transfer operation cycle (Specify a multiple of 10 milliseconds. Specify 10 if software transfer is not used.)	10 to 85400000 (multiple of 10)	1000
DFNO	Data field number	0 to 255 (*2)	–

(*1) Set "0" (no data field to be used) for the "MAXDFCNT" (total number of registration data fields) definition information of nxacp.conf on the HP side. Note that MAXDFCNT cannot be set to "0" on the CP side.
(*2) You must create "dfN" for each data field number defined here.
In the following "definition example", you must create "df0" for "DFNO 0" and "df1" for "DFNO 1".

<Definition example>

```
MAXDFCNT    2
TRCCS       512
TMSCYC      100
DFNO        0
DFNO        1
```

(2) Definition information in a data field definition file (dfN)
The following information is defined in a data field definition file.
The I, L, and R columns on the right side indicate the type of a data field (I: local node, L: local, R: remote) and whether the definition is necessary for each type (√: Definition necessary, ×: Definition not necessary, –: Follows the definition of the local DF).
If a specific definition is not required, specify the value shown in the "If not used" column for the definition, delete the identifier, or insert "#" (pound sign) before the identifier. If the value shown in the "If not used" column for a definition is "–", you cannot delete the definition.

Identifier	Description	Configurable range	If not used	L	R	I
DFTYPE	Data field type (=0: local, ≠0: remote (Source data field number))	0, 1 to 255	–	√	√	×
NODENAME	Local node name	9 or less ASCII characters	–	√	×	×
NODENO	Local node number	1 to 4095	–	√	×	×
SSPORT	Send local port number	1 to 65535	–	√	–	×
ALPORT	Alive signal destination port number	1 to 65535	–	√	×	×
ALSCYC	Alive signal transmission interval (in seconds)	1 to 3600	–	√	×	×
ALTOUT	Alive signal timeout time (in seconds)	1 to 3600	–	√	×	×
ALRCVFG	Alive signal reception on/off flag (0: Not received, 1: Received)	0 and 1	0	√	×	×
UNO1	UNO of LAN1	1 to 23	–	√	×	×
UNO2	UNO of LAN2	2 to 23	0	√	×	×
MINSNDMCG	Minimum send multicast group number	1 to 255	0	√	√	×
MAXSNDMCG	Maximum send multicast group number	1 to 255	0	√	√	×
SNDMCGCNT	Total number of send multicast groups	1 to 128	0	√	√	×
MINRCVMCG	Minimum receive multicast group number	1 to 255	0	√	×	×
MAXRCVMCG	Maximum send multicast group number	1 to 255	0	√	×	×
RCVMCGCNT	Total number of receive multicast groups	1 to 32	0	√	×	×
MINUSETCD	Minimum transmission/reception TCD number	1 to 59999	–	√	√	√
MAXUSETCD	Maximum transmission/reception TCD number	1 to 59999	–	√	√	√
USETCDCNT	Total number of transmission/reception TCDs	1 to 3328	–	√	√	√
SBUFSZ	Size of one case of send buffer	512 to 1408	0	√	–	×
SBUFCNT	Number of cases in a send buffer	1 to 1024	0	√	√	×
RBUFSZ	Size of one case of receive buffer	512 to 1408	0	√	–	√
RBUFCNT	Number of cases in a receive buffer	1 to 1024	0	√	√	√
BUFHWRATE	HIGH WATER alarm buffer usage ratio	2 to 100	–	√	√	√
BUFLWRATE	LOW WATER alarm buffer usage ratio	1 to 99	–	√	√	√
MINNODENO	Minimum node number	1 to 4095	0	√	√	×
MAXNODENO	Maximum node number	1 to 4095	0	√	√	×
R_INA1	Network address of LAN1	XX.XX.XX.XX	–	×	√	×
R_NMASK1	Netmask of LAN1	XX.XX.XX.XX	–	×	√	×
R_INA2	Network address of LAN2	XX.XX.XX.XX	0	×	√	×
R_NMASK2	Netmask of LAN2	XX.XX.XX.XX	0	×	√	×
CYCFG	Network-shared memory use flag (0: Not used, 1: Used)	0 and 1	0	√	×	×
SYSPORT	System multicast communication port number	1 to 65535	0	√	√	×
UNIMCGNO	Send unicast group number	MINSNDMSG to MAXSNDMSG	0	√	√	×

<Advice on efficient use of memory>
For each node, we recommend using consecutive numbers for reception TCDs and MGNs.
For each data field, we recommend using consecutive numbers for node numbers.

[Caution]

The following shows cautionary notes for each identifier.

SBUFSZ,
RBUFSZ: Specify the user data size in one buffer case in bytes. The specified number must be a multiple of 4.
To obtain the actual size of one buffer case, you must add 64 bytes (NX header) and 4 bytes (buffer management header) to the specified value.

MINUSETCD,
MAXUSETCD,
USETCDCNT: Define these values for transmission and reception TCDs. However, if transmission TCD check is not used, define them for reception TCDs only.

R_INA: Data defined only for a remote data field. Use dot-decimal notation.

R_NMASK: Data defined only for a remote data field. Use dot-decimal notation.

CYCFG: Use macros to define detailed settings of network-shared memory.

SBUFCNT,
RBUFCNT: You cannot specify 0 for RBUFCNT in the local node data field.
If you specify 0 for SBUFCNT and RBUFCNT in a remote data field, an error is not generated, but actual transmission and reception are stopped.
If you specify 0 for SBUFCNT and RBUFCNT in a local data field, sending only alive signals is permitted.

[Caution] Continued

ALSCYC,
ALSTOUT: When you define these two parameters, make sure that ALSCYC < ALTOUT.

<Definition example for a local data field>

```
DFTYPE          0

NODENAME        OOMIKA
NODENO          3

SSPORT          1025
ALPORT          600

ALSCYC          10
ALTOUT          30
ALRCVFG         0

UNO1            3
UNO2            0

MINSNDMCG       1
MAXSNDMCG       4
SNDMCGCNT       4
MINRCVMCG       5
MAXRCVMCG       8
RCVMCGCNT       4

MINUSETCD       1
MAXUSETCD       64
USETCDCNT       32

SBUFSZ          1408
SBUFCNT         128
RBUFSZ          1408
RBUFCNT         128
BUFHWRATE       80
BUFLWRATE       30

MINNODENO       1
MAXNODENO       64

CYCFG           0
SYSPORT         0
```

<Definition example for the data field for local node communication>

```
DFTYPE          0

MINUSETCD       1
MAXUSETCD       64
USETCDCNT       32

RBUFSZ          1408
RBUFCNT         128

BUFHWRATE       80
BUFLWRATE       30
```

<Definition example for a remote data field>

```
DFTYPE          1

MINSNDMCG       1
MAXSNDMCG       4
SNDMCGCNT       4

SBUFCNT         128
RBUFCNT         128

BUFHWRATE       80
BUFLWRATE       30

MINNODENO       1
MAXNODENO       64

R_INA1          192.168.0.1
R_NMASK1        255.255.255.0
R_INA2          0
R_NMASK2        0
```

(3) Definition information for a multicast group definition file (mcgN)
The following information is defined in this file.
The I, L, and R columns on the right side indicate the type of a data field (I: local node, L: local, R: remote) and whether the definition is necessary for each type (√: Definition necessary, ×: Definition not necessary, –: Follows the definition of the local DF).
If a specific definition is not required, specify the value shown in the "If not used" column for the definition, delete the identifier, or insert "#" (pound sign) before the identifier. If the value shown in the "If not used" column for a definition is "–", you cannot delete the definition.

Identifier	Description	Configurable range	If not used	L	R	I
SMGN	Send multicast group number	1～255 (*1)	–	√	√	×
RMGN	Receive multicast group number	1～255 (*1)	–	√	×	×
OPORT	Online mode port number (*2)	1～65535	0	√	√	×
TPORT	Test mode port number (*2)	1～65535	0	√	√	×

(*1) The maximum number of send multicast groups that can be registered is SNDMCGCNT defined in dfN. The maximum number of receive multicast groups that can be registered is RCVMCGCNT defined in dfN.
(*2) Defines the send destination port number for transmission and the port number bound by bind() for reception.
You must define online/test mode port numbers independently for transmission and for reception.

<Definition example>

```
#MGNTYPE    MGN     OPORT    TPORT
    SMGN      1     55001    57001
    SMGN      2     55002    57002
    SMGN      3     55003    57003
    SMGN      4     55004    57004

#MGNTYPE    MGN     OPORT    TPORT
    RMGN      5     55005    57005
    RMGN      6     55006    57006
    RMGN      7     55007    57007
    RMGN      8     55008    57008
```

If you want to use only the online mode or the test mode, specify 0 for the port number of the unused mode. (If you specify 0 for the port number of one mode and issue nx_dfup() in that node mode, nx_dfup() terminates with a socket macro error.)

[Caution]
- Note that, even if you specify (for a receive multicast group) a multicast group number to which the local node sends messages, messages sent from the local node still cannot be received.
- The port numbers defined as SSPORT, ALPORT, and SYSPORT in dfN must not be used in the definitions in mcgN.
- To enable communication with NX Dlink (communication package for a WS, PC, and control server) by using a test mode port, specify "0" for LNSYSTYPE in the dfieldN definition information for NX Dlink.
- To enable communication with NX Dlink, specify "1" for RCVLEVEL in the dfieldN definition information for NX Dlink.

(4) Definition information for a reception transaction code definition file (rcvtcdN)
The following information is defined in this file.
The I, L, and R columns on the right side indicate the type of a data field (I: local node, L: local, R: remote) and whether the definition is necessary for each type (√: Definition necessary, ×: Definition not necessary, –: Follows the definition of the local DF).
If a specific definition is not required, specify the value shown in the “If not used” column for the definition, delete the identifier, or insert “#” (pound sign) before the identifier. If the value shown in the “If not used” column for a definition is “–”, you cannot delete the definition.

Identifier	Description	Configurable range	If not used	L	R	I
TCDNO	Reception transaction code number	1 to 59999 (*1)	–	√	×	√
BUFCNT	Maximum number of cases in a receive buffer	4 to 1024 (*2)	–	√	×	√
TN	Receive task number	1 to 208 (*3)	–	√	×	√

(*1) The maximum number of TCDNO definitions that can be registered is USETCDCNT defined in dfN.
For TCDNO, you can specify a value between MINUSETCD and MAXUSETCD defined in dfN.
(*2) The maximum number of cases in a receive buffer (BUFCNT) must not exceed RBUFCNT defined in dfN. If you specify 0 for BUFCNT, RBUFCNT defined in dfN is actually used as BUFCNT. If remote DFs are connected, the sum of RBUFCNT defined in dfN for each remote DF is added to the actual maximum number of cases in a receive buffer.
(*3) The maximum number of tasks that can be registered for one TCDNO is 8.

<Definition example>

```
# RCV   TCDNO   BUFCNT   TN1   TN2   TN3   TN4   TN5   TN6   TN7   TN8
TCD         1   16         1    11    21    31    41    71
TCD         2   16         2   112   142
TCD         3   16        16    32    48    64    80    96   112   128
TCD         4   16        64   128
```

(5) Definition information for a transmission transaction code definition file (sndtcdN)
The following information is defined in this file. If TCD transmission right check is not required, you do not have to create this file.
The I, L, and R columns on the right side indicate the type of a data field (I: local node, L: local, R: remote) and whether the definition is necessary for each type (√: Definition necessary, ×: Definition not necessary, –: Follows the definition of the local DF).
If a specific definition is not required, specify the value shown in the "If not used" column for the definition, delete the identifier, or insert "#" (pound sign) before the identifier. If the value shown in the "If not used" column for a definition is "–", you cannot delete the definition.

Identifier	Description	Configurable range	If not used	L	R	I
TCDNO	Transmission transaction code number	1 to 59999 (*1)	–	√	√	√
TN	Send task number	1 to 208 (*2)	–	√	√	√

(*1) The maximum number of TCDNO definitions that can be registered is USETCDCNT defined in dfN.
For TCDNO, you can specify a value between MINUSETCD and MAXUSETCD defined in dfN.
(*2) The maximum number of tasks that can be registered for one TCDNO is 8.

<Definition example>

```
#SND    TCDNO   TN1   TN2   TN3   TN4   TN5   TN6   TN7   TN8
TCD         1     1    11    21    31    41    71
TCD         2     2   112   142
TCD         3    16    32    48    64    80    96   112   128
TCD         4    64   128
```

[NOTE]
If you register a task, you must define the mapping between the task and transmission TCDs for all data fields the task sends messages to. You cannot turn on or off transmission check on a per-data-field basis.

8.2.3 Compiling construction information

Use the following command to compile the definition information for each site.
Execute the following command also when you update the construction information.

```
# confnxsv site_name [Enter]
```

<Error output on the monitor and its contents>
Parameter count error: The following message is displayed.

```
parameter number out of range
usage:confnxsv site
```

Specify the site name of the CP for site_name.
If an error message such as a compiler error message is output while the confnxsv is executed, check and correct the construction information. If any abnormality occurs when construction information on the HP side is compiled, set "0" to MAXDFCNT, which is definition information in `nxacp.conf` on the HP side.

<Output on the monitor during execution and its contents>
If the command is executed without an error, the following messages are output by the command.

```
*** NXACP/S10VE GENERATION START(CPU_name) ***
                          |<-------- Checks the construction information of CP.
*** CP(site_name) CONFIGURATION OK !! ***
                          |<-------- Checks the construction information of HP.
*** HP(site_name)GENERATION OK !! ***
*** NXACP/S10VE GENERATION END ***
```

[Note]
In the S10VE, if the same configuration information is used on another node, you can copy the site information to the node. To copy the site information, simply copy all configuration files to the node, change the node name and node number (in dfN), and then execute the command above. (Perform this after you delete all object files under the OBJ directory in the construction environment.)

8.2.4 Loading configuration information

Use the tblldnxsv command to load the information output from the confnxsv command to the specified site as control tables.

```
# tblldnxsv site_name ON/OFF [Enter]
```

By running this command, the configuration information is loaded to the actual environment in the machine.
If you specify the ON option, the construction information of NXACP is loaded to the actual environment in the S10VE specified as the site name. Note that NXACP is forcibly stopped temporarily while the construction information is loaded.
If you specify the OFF option, the construction information of NXACP is loaded to a memory image file in the S10VE specified as the site name. When loading the memory image file to the actual environment in the S10VE, use the remote loading command (svrpl) provided by RPDP to load the whole construction information at the same time.

<Error output on the monitor and its contents>

Parameter count error: The following message is displayed.

```
parameter number out of range
usage:tblldnxsv site ON/OFF
```

Error generated during execution: If an RPDP-related error is displayed and the execution is aborted, refer to *S10VE Software Manual Operation RPDP for Windows*® (manual number SEE-3-133).

<Output on the monitor during execution and its contents>

If the command is executed without an error, the following messages are output by the command.

```
*** NXACP/S10VE TABLE LOAD START(CPU_name)***
                        |      ◄────────  Checks the table load for CP.
*** CP(site_name)TABLE LOAD OK !! ***
                        |      ◄────────  Checks the table load for HP.
*** HP(site_name) TABLE LOAD OK !! ***
*** NXACP/S10VE TABLE LOAD END ***
```

[Note]

When you run the tblldnxsv command after all construction files are copied, make sure that you delete all obj files under the OBJ directory before you run the command. (The tblldnxsv command compares obj files, and consequently, if construction files are newly copied, the tblldnxhr command thinks that the construction information is being loaded to the site backup file. As a result, tables are not loaded correctly.)

8.3 Estimating Required Memory Capacity

In NXACP, you can configure maximum limits, table counts, and other settings as system configuration parameters to adapt to various system configurations. When you specify these parameters, use the following method to calculate the memory capacity necessary for constructing NXACP. The identifiers shown below are those for construction information. When memory is actually allocated, in a page unit (4096 bytes) is allocated to fit the total capacity for each processor.

Table 2-24 Capacity Calculation Sheet

Area name		Size of one case in bytes (corresponding construction file name)	Case count (corresponding construction file name)	Total (*2) (in bytes)
Common fixed block				53,248
Local node	Receive buffer	RBUFSZ(df0)+228	RBUFCNT(df0)	
	TCD management block	4	MAXUSETCD(df0) - MINUSETCD(df0) + 1	
		64	USETCDCNT(df0)	
Local DF	Send buffer	SBUFSZ(dfN)+100 (*3)	SBUFCNT(dfN)	
	Receive buffer	RBUFSZ(dfN)+228 (*4)	RBUFCNT(dfN)	
	TCD management block	4	MAXUSETCD(dfN) - MINUSETCD(dfN) + 1	
		64	USETCDCNT(dfN)	
	Send MCG management block	16	MAXSNDMCG(dfN) - MINSNDMCG(dfN) + 1	
	Receive MCG management block	1	MAXRCVMCG(dfN) - MINRCVMCG(dfN) + 1	
		32	RCVMCGCNT(dfN)	
	Node management block	32	MAXNODENO(dfN) - MINNODENO(dfN) + 1	
		32	RCVMCGCNT(dfN) × (number of defined nodes) (*1)	
Remote DF	Send buffer	SBUFSZ (*5) +100 (*3)	SBUFCNT(dfN)	
	Receive buffer	RBUFSZ (*5) +228 (*4)	RBUFCNT(dfN)	
	TCD management block	4	MAXUSETCD(dfN) - MINUSETCD(dfN) + 1	
		64	USETCDCNT(dfN)	
	Send MCG management block	16	MAXSNDMCG(dfN) - MINSNDMCG(dfN) + 1	
	Node management block	32	MAXNODENO(dfN) - MINNODENO(dfN) + 1	
		32	RCVMCGCNT (*3) × (number of defined nodes) (*1)	
Trace block		32	TRCCS	
Total				

(*1) (number of defined nodes) means MAXNODENO(dfN) - MINNODENO(dfN) + 1.
(*2) The total is calculated as (Size of one case in bytes) × (Case count).
(*3) 100 here refers to the size of send buffer management information.
(*4) 228 here refers to the size of receive buffer management information.
(*5) Source DF specified for DFTYPE in dfN.

If memory transfer is used for a local data field, the table capacity is calculated by adding 768 bytes to the numeric value calculated by the method shown above. If an alive signal is received, the table capacity is calculated by adding 1536 bytes to the numeric value calculated by the method shown above.

This Page Intentionally Left Blank.

PART 3   MACRO SPECIFICATIONS

CHAPTER 1 INTRODUCTION

1.1 Macro Types and Macro List

NXACP provides macros in the form of IRSUBs. The stack size necessary for issuing a macro must be allocated by a user task. The return code is directly returned as a value by a macro. You do not have to use geterrno() to read the value.
Table 3-1 shows a list of NXACP macros.

Table 3-1 Macro List

<Multicast communication macros>

Macro name	Summary of function	Stack size (in bytes)	Issuability	
			CP	HP
nx_put	Send a multicast message	2048	Y	N
nx_get	Send a multicast message	4096	Y	N

<Operation management macros>

Macro name	Summary of function	Stack size (in bytes)	Issuability	
			CP	HP
nx_init	Start NXACP	4096	Y	N
nx_dfup	Start a data field	4096	Y	N
nx_dfdwn	Stop a data field	4096	Y	N
nx_quit	Stop NXACP	4096	Y	N

<Shared memory macros>

Macro name	Summary of function	Stack size (in bytes)	Issuability	
			CP	HP
nx_init_tm	Initialize transfer memory	512	Y	N
nx_ctl_tm	Start/stop transfer for the transfer memory	512	Y	N
nx_get_tm	Get the control information of the transfer memory	512	Y	N
nx_write_tm	Write data to the transfer memory	512	Y	N
nx_read_tm	Read data from the transfer memory	512	Y	N

Y: Can be issued.
N: Cannot be issued.

[Caution]

- You must always specify “0” for the items currently not used in the macro interface because some new functions may be assigned to these items in the future.
- Only user tasks can issue NXACP macros. A user task here means a task created with “RSUTYP=u” specified.

CHAPTER 2 MULTICAST COMMUNICATION MACROS

2.1 nx_put

[Name]

nx_put - Send a message

[Syntax]

```
#include <nxacp.h>
long     nx_put (msg, msglen, ginfo);

char       *msg;
long       *msglen;
nx_ginfo   *ginfo;
```

[Parameters]

msg[in]	Specifies the start address of the user data to be sent.
msglen[in]	Specifies the size of the user data to be sent (in bytes).
nx_ginfo[in]	Specifies the following transmission information.

```
typedef struct   {
   unsigned char   dfn;       /* Transmission destination data field number */
   char   fu1;                /* For future use (Fixed to 0) */
   unsigned short   mgn;      /* Transmission destination multicast group number */
   unsigned short   tcd;      /* Transmission transaction code */
   short   fu2;               /* For future use (Fixed to 0) */
   unsigned long   sa;        /* This area is used for nx_get() and not used for nx_put().
                                 (Fixed to 0) */
   long   rcvlen;             /* This area is used for nx_get() and not used for nx_put().
                                 (Fixed to 0) */
}nx_ginfo;
```

[Description of the function]

The nx_put() macro only supports multicast one-way communication. By issuing nx_put() once, one message is broadcast into one data field.
Specify the start address of the transmission message area for “msg” and the length of transmission data in bytes for “msglen”. Up to 16 KB of data can be transmitted at one time.
If you want to send a message to the local node, specify “0” for dfn as a destination data field. As a result, NXACP does not send the message to the network, and internally sends the message back to the local node.
If you want to send a message to other nodes, specify a value other than “0” for dfn. As a result, NXACP does not send the message back to the local node.
For this reason, you cannot send a message to both the local node and other nodes by issuing nx_put() once. If you want to send the same message to the local node and also other nodes, you must issue nx_put() multiple times with different destination data field numbers.
The nx_put() macro returns when the macro stores the transmission data in the queue in NXACP.
Only a non-blocking call is supported by nx_put(). If allocating space for the transmission data in the queue fails due to the queue being full or other reason, the macro returns with an error without retrying allocating space in the queue.
NXACP determines and controls the priority levels and data modes of transmission messages.

[Return value]

When nx_put finishes normally, 0 is returned.
If nx_put terminates with an error, the following value is returned.

0x001: NXACP has not been initialized.
0x003: Construction information has not been loaded.
0x102: The data field specified by dfn has not been defined.
0x103: The data field specified by dfn has not been initialized.
0x108: The data field of another PU specified for dfn has not been initialized (including the period when the state is transitioning).
0x111: The value of mgn is out of the configurable range (1 to 255).
0x112: The multicast group specified by mgn has not been defined.
0x121: The value of tcd is out of the configurable range (1 to 59999) or has not been defined.
0x131: The value of msglen is out of the configurable range (0 to 16384).
0x201: Allocating space for the request in the send buffer failed.
0x211: All network modules have failed or are in the CARDOFF state.
0x231: An error has been detected in the inter-PU communication.
0x301: The task number of the task that issued the request is out of range (1 to 208).
0x303: The value of tcd has not been defined. (This code is used only when the TCD transmission permission check is enabled.)

[Note]

The information about the data fields and multicast groups to be specified as a transmission destination must be set up in the data field construction information and multicast group construction information respectively.
Only information of the errors that occur before connecting to the send queue in NXACP can be returned as a return code of nx_put().
If an error occurs in the built-in Ethernet, the error is reported to EAS. Additionally, even though nx_put() is normally terminated, if a network error occurs before transmission processing, or if processing stops at nx_quit() or nx_dfdwn(), no message might be sent. If transmission to the network becomes impossible, transmission messages connected to the send queue are discarded by NXACP.
NXACP guarantees that the transmission requests from tasks at the same level are processed and sent to the network in the order of requests. If you want to be absolutely sure that messages are sent in the order of requests, confirm nx_put() has returned before issuing the next data transmission request nx_put().

2.2 nx_get

[Name]

nx_get - Receive a message

[Syntax]

```
#include <nxacp.h>
int      nx_get (buf, buflen, offset, nx_ginfo, time);

char        *buf;
long        *buflen,*offset;
nx_ginfo    *ginfo;
long         *time;
```

[Parameters]

buf[out] Specifies the start address of the area used for storing the reception data. Received data is stored in this area.

buflen[in] Specifies the size of the area used for storing the reception data.
This value is used as a requested reception size if the size of the arrived message is larger than this value.

offset[in] Specifies the relative address (the start address of the whole message is 0) of the part of the message to be received (if only a part of the message is received). If the whole message is received, specify “0”.

nx_ginfo[out] Specifies the following transmission information.

```
typedef struct    {
   unsigned char    dfn;        /* Reception destination data field number */
   char    fu1;                 /* For future use (Fixed to 0) */
   unsigned short    mgn;       /* Reception destination multicast group number */
   unsigned short    tcd;       /* Reception transaction code */
   short    fu2;                /* For future use (Fixed to 0) */
   unsigned long    sa;         /* Sender address (*) */
   long    rcvlen;              /* Received data size */
                                /* xFFFFFFFF if a timeout occurs. */
}nx_ginfo;
```

time[in] Specifies the reception timeout monitoring time (in seconds).

(*) The meaning of the sender address is the same as SA in “APPENDIX F MESSAGE HEADER FORMAT”.

[Description of the function]

The nx_get() macro receives one message that has the TCD specified at system construction.
Specify the start address of the reception message area for “buf” and the size of the area for “msglen”.
Specify the relative address (the start address of the whole message is 0) of the part of the message to be received (if only a part of the message is received).
When data is received, the received data is stored in buf, and the size of the received data is stored in rcvlen.
Only the part of the received data specified by msg_offset is stored. If the size of the received message is larger than buflen, the first buflen bytes of the message starting from the relative address specified by offset are stored in buf.
Up to 16 KB of data can be received by one call of nx_get().
The detailed information of the received data including the sender data field number, sender node number, destination multicast group, and transaction code is stored in nx_ginfo.
For time, specify the duration of time that nx_get() waits for data to arrive. You can specify a value between -1 and 3600 (in seconds).
If you specify “0” for time, nx_get() waits indefinitely for a message to arrive.
If you do not want to wait for a message when a message has not arrived yet, specify “-1” for time. When you specify “-1” for time, nx_get() processes the received message if a message has already arrived. If not, nx_get() returns with return code 0, and “-1” is set in rcvlen.
Also when a timeout occurs, nx_get() returns control to the user task with return code 0 because nx_get() exits normally as a routine in NXACP, and “-1” is set in rcvlen.
The messages to be received are specified at system construction. The nx_get() macro cannot receive a specific message select.
Received messages are connected to user task queues in the order of reception from the network. Then the messages are serviced first-in first-out (FIFO order by each user task.

[Return value]

When nx_get finishes normally, 0 is returned, and the size of the received data (0 or larger) is set in rcvlen.
If a timeout occurs while waiting for a message to arrive, “-1” is set in rcvlen.
If nx_get terminates with an error, the following value is returned.

0x001: NXACP has not been initialized (nx_init() has not been issued) or has been stopped (nx_quit() has been issued).
This code is also generated when the control information is swapped.
0x003: Construction information has not been loaded.
0x131: A negative value has been set for buflen.
0x141: The value of offset is out of the configurable range (0 to 16384).
0x151: The value of time is out of the configurable range (-1 or between 0 and 3600).

0x202: The receiving buffer was purged when nx_dfdwn() was called.

0x301: The task number of the task that issued the request is out of range (1 to 208).
0x302: Reception TCDs have not been defined for the caller user task.

[Note]

The TCDs of the messages to be received must be defined in advance at system construction.
Only the information of the errors that occur before a message is received from the receive queue in NXACP can be returned as a return code of nx_get(). If an error occurs in the built-in Ethernet, the error is reported to EAS.
If you stop a data field using nx_dfdwn(), or if you stop NXACP using nx_quit(), the messages waiting to be processed are all purged.
In addition, when nx_quit() is issued, nx_get() is aborted and returns control to a user.

CHAPTER 3 OPERATION MANAGEMENT MACROS

3.1 nx_init

[Name]

nx_init - Start NXACP

[Syntax]

```
#include <nxacp.h>
long   nx_init (info)

long   info[3];
```

[Parameters]

info[out]	Specifies the start address of the detailed information storage area used when initialization fails.

[Description of the function]

The nx_init() macro starts the process of NXACP and initializes shared tables.
A user must run this macro first when starting the system.

[Return value]

When nx_init() finishes normally, 0 is returned.
If nx_init() terminates with an error, the following value is returned. For information about the relationship with the error code stored in the detailed information storage area, see “APPENDIX A RETURN CODE DETAILS”.

0x002:	NXACP has already been initialized.
0x003:	Construction information has not been loaded.
0x004:	The NXACP system tasks have not been loaded (including a task startup failure).
0x1e1:	The network module type or IP address is invalid.
0x1e3:	An error has been detected in the network module type in the duplexed LAN configuration.
0x1f1:	An error has been detected in the socket macro.
0x2e1:	The NXACP state is currently transitioning.

[Note]

- You can issue nx_get() immediately after you issue nx_init().
- In NXACP, an installation check for the network module is performed in nx_init(). Therefore, when the uninstalled UNO is defined, processing stops as an error.
- Even when a data field is constructed as duplexed LANs, nx_init() aborts the initialization process when an error is detected in either one of the duplexed LANs.

3.2 nx_dfup

[Name]

nx_dfup - Start a data field

[Syntax]

```
#include <nxacp.h>
long  nx_dfup (dfn, nmode, mmode, info)

long  *dfn;
long  *nmode;
long  *mmode;
long  info[3];
```

[Parameters]

dfn[in]	Specifies the data field number of the data field to be initialized.
nmode[in]	Specifies a node mode.
mmode[in]	Specifies a message mode.
info[out]	Specifies the start address of the detailed information storage area used when initialization fails.

[Description of the function]

The nx_dfup() macro puts the specified data field into the OPEN state. To receive/send a message from/to a data field, you must issue nx_dfup() to put the data field into the OPEN state. Transmission of alive signals also starts when nx_dfup() is issued.
The following explains the values you can specify for nmode and mmode and their meanings.

- nmode
  0: Online mode
  1: Test mode
- mmode (valid only when nmode is Test mode (1).)
  0: Online messages only
  1: Test messages only

You can start only one data field by issuing nx_dfup() once. You must issue nx_dfup() the same number of times as the data fields you want to use.
If nx_dfup() terminates with an error, the specified data field remains in the CLOSE state.
When nx_dfup() terminates with an error, nx_dfup() releases all the resources, such as ports that have been allocated, until the error is detected.
If the error is caused by an error in the construction information, check and correct the construction information. Especially, make sure you use port numbers different from other subsystems.
When duplexed LANs are used for a data field, nx_dfup() terminates with an error if both LANs cannot be initialized normally. The data field cannot be put into the OPEN state if only one of the duplexed LANs can be initialized normally. Remove the error immediately, and then retry the operation.

[Return value]
When nx_dfup() finishes normally, 0 is returned.
If nx_dfup() terminates with an error, the following value is returned. For information about the relationship with the error code stored in the detailed information storage area, see “APPENDIX A RETURN CODE DETAILS”.

0x001:	NXACP is not initialized.
0x003:	Construction information is not loaded.
0x004:	The NXACP system tasks are not loaded (including a task startup failure).
0x101:	The value of dfn is out of the configurable range.
0x102:	The data field specified by dfn is not defined.
0x104:	The data field specified by dfn is already initialized.
0x106:	The source data field is not defined.
0x107:	The source data field is not initialized.
0x161:	An nmode setting error.
0x162:	An mmode setting error. (Specify “0” for mmode when nmode is 0 (online mode). If you specify a non-zero value, the value is invalid.)
0x1e6:	All modules of the UNOs for the data field specified in the construction information failed.
0x1f1:	An error was detected in the socket macro.
0x2e1:	The NXACP state is currently transitioning.
0x2f1:	The state of the data field dfn is currently transitioning.
0x2f2:	The state of the source data field is currently transitioning.

[Note]
Immediately after nx_dfup() is issued, receiving messages from the network starts.
The later nx_get() starts being issued after nx_dfup() is issued, the higher the possibility of buffer overflow. (While a message receiver user task is in the IDLE or DORMANT state, received messages for the task are discarded by NXACP.)
You must register a subroutine to EAS before issuing nx_dfup(). If an error is detected while nx_dfup() is executed, nx_dfup() terminates and returns an error. At the same time, the network-related settings revert back to the original settings used before nx_dfup() is issued, and the control is returned to the user. If an error occurs in the built-in Ethernet, the error is reported to EAS. We recommend making it possible to analyze the location of the failure based on not only the error return code from nx_dfup(), but also its error log information.
If nx_dfup() is issued again to the same data field without issuing nx_dfdwn() to the data field in between, an error is returned (error code: 0x104). The mode information specified in the arguments of the second nx_dfup() are ignored, and the mode information in the first nx_dfup() continues to be used.
Before you start local node communication, you must OPEN data field #0 dedicated to local node communication.

[Note] Continued

- Even when a data field is constructed as duplexed LANs, nx_dfup() aborts the initialization process when an error is detected in either one of the duplexed LANs.
- If you issue this macro for a data field when it is specified in the multicast group definition file (mgnN) that ports are not used (=0) for the data field, this macro terminates and returns error code 0x1f1.

3.3 nx_dfdwn

[Name]

nx_dfdwn - Close a data field

[Syntax]

```
#include <nxacp.h>
long   nx_dfdwn (dfn)

long   *dfn;
```

[Parameters]

dfn[in] Specifies the data field number of the data field to be stopped.

[Description of the function]

The nx_dfdwn() macro changes the state of the specified data field from OPEN to CLOSED. Note that, when a local data field is specified, the remote data fields that are connected through the local data field are also put into the CLOSED state.
When nx_dfdwn() is issued to a data field, the data field stops accepting message transmission requests from user tasks, stops the reception process for the messages targeted to itself, and purges the messages waiting to be processed (waiting to be sent, waiting to be received, or being reconstructed). At the same time, if a task has issued nx_get() to the data field, the nx_get() terminates and returns error code 0x202 to the task. For this reason, the task can reissue nx_get() to get a message from other data fields.
The nx_dfdwn() macro sends alive signals for scheduled shutdown to notify the status change to other nodes.
You can stop only one data field by issuing nx_dfdwn() once. You must issue nx_dfdwn() the same number of times as the data fields in the OPEN state.

[Return value]

When nx_dfdwn() finishes normally, 0 is returned.
If nx_dfdwn() terminates with an error, the following value is returned.

0x001: NXACP has not been initialized or has been stopped.
0x003: Construction information has not been loaded.

0x101: The value of dfn is out of the configurable range (0 to 255).
0x102: The data field specified by dfn has not been defined.
0x103: The data field specified by dfn has not been initialized.

0x2e1: The NXACP state is currently transitioning.
0x2f1: The state of the data field dfn is currently transitioning.
0x2f2: The state of the source data field is currently transitioning.
0x2f3: The state of a remote data field is currently transitioning.

[Note]

Even when an error is detected in the closing process of ports, etc., while nx_dfdwn() is executed, nx_dfdwn() does not terminate with an error.
The nx_dfdwn() macro sends an alive signal in the SHUTDOWN mode three times at constant intervals, and as a result, takes 5 to 10 seconds to finish.

3.4 nx_quit

[Name]
nx_quit - Stop NXACP

[Syntax]
```
#include <nxacp.h>
long   nx_quit()
```

[Parameters]
None

[Description of the function]
The nx_quit() macro stops NXACP.
If you issue nx_quit() without issuing nx_dfdwn() beforehand, all data fields are stopped.
When you issue nx_quit() without issuing nx_dfdwn(), nx_quit() stops all data fields from accepting message transmission requests from user tasks, stops the reception process for the messages targeted to all data fields, and sends alive signals in the SHUTDOWN mode to notify the status change to other nodes.

[Return value]
When nx_quit() finishes normally, 0 is returned.
If nx_quit() terminates with an error, the following value is returned.

0x001: NXACP is not initialized.
0x003: Construction information is not loaded.

0x2e1: The NXACP state is currently transitioning.
0x2f1: At least one data field is currently transitioning. “Currently transitioning” means that either one of the following events are being processed.
nx_dfup() and nx_dfdwn() are being processed.

[Note]
Even when an error is detected in the closing process of ports, etc., while nx_quit() is executed, nx_quit() does not terminate with an error.
Like nx_dfdwn(), nx_quit() sends an alive signal in the SHUTDOWN mode three times at a constant intervals, and as a result, takes 5 to 10 seconds to finish.

CHAPTER 4 SHARED MEMORY MACROS

4.1 nx_init_tm

[Name]

nx_init_tm - Initialize transfer memory

[Syntax]

```
#include   <nxacp.h>
long   nx_init_tm( df, tmid, tmmap)

long          *df;
long          *tmid;
struct tmmap   *tmmap;
```

The following shows the details of the tmmap structure.

```
struct tmmap {
    long   m_type;
    long   m_mcg;
    long   m_cflags;
    long   m_memid;
    long   m_mincyctm;
    long   m_caseno;
    long   m_maxblk;
    struct   {
        long   m_cyctm;
        long   m_blkno;
        long   m_blkcnt;
    }m_send[m_caseno];
} tmmap;
```

[Parameters]

df: Specifies a data field number between 1 and the maximum number of data fields.
You cannot select a data field for local node communication (DF=0) or a remote data field.

tmid: Specifies a transfer memory ID between 1 and 4.

m_type: Specifies a memory transfer type.
TMEM_SOFT(0): Periodic transmission type software transfer
TMEM_FAST(2): Split-transmission type software transfer

m_mcg: Specifies a multicast group number between 1 and the maximum number of multicast groups.

m_cflags: Specifies transfer memory control flags. Specify this item when you use the following options. If you do not use the following options, specify "0" for this item.
CMF_ACL(1): Enables mutual exclusion.
(You cannot enable this flag for high-speed software transfer.)
CMF_COLERR(2): Enables duplication check for write areas.
If you want to enable both, specify these two flags with "or" (CMF_ACL | CMF_COLERR).

m_memid: Specifies the address of CM or GLB assigned to the transfer memory when software transfer is used.

m_mincyctm: Specifies the minimum cycle of cyclic communication (minimum of the m_cyctm values described below). Specify the same value for a TMID on all computers that use the TMID.
For software transfer, this value is not used, and you must specify "0" for this item.

m_caseno: Specifies the number of valid transmission areas (that is, the number of elements in m_send[N] described below).
If transmission is disabled for transfer memory from this CPU, specify "0" for this item. Then, you do not have to set up m_send[].

m_maxblk: Specifies the maximum block number used by transfer memory.

m_send[m_caseno]: Specifies the transmission area information of transfer memory.

```
    struct    {
        long    m_cyctm;  Specifies the transfer cycle for each transfer area.
                          For software transfer, specify a value in milliseconds.
                          (Example: 1 second = 1000)
        long    m_blkno;  Specifies the start block number of the transfer area relative to the
                          beginning of the transfer memory.
                          For software transfer:0 to 16383
        long    m_blkcnt; Specifies the number of blocks in the transfer area.
                          For hardware transfer: 1 to 15872
                          For software transfer: 1 to 255
    }m_send[m_caseno];
```

[Description of the function]
This macro specifies the attributes of the transfer memory and allocates physical memory to the transfer memory.
Issue this macro when the system starts or when new transfer memory is created.

[NOTE]
If you specify improper values for the number of transfer cases, transfer size, transfer cycle, or the number of connected nodes when you use memory transfer, the CPU load may increase. You must choose appropriate values for these items when you design the system.

[Return value]
When nx_init_tm() finishes normally, 0 is returned.
If nx_init_tm() terminates with an error, the following value is returned.

0x001: NXACP has been stopped.
0x003: Control tables have not been loaded.

0x101: The data field number is out of range.
0x102: The data field has not been defined.
0x103: The data field has not been initialized.
0x105: The data field is a remote data field.
0x111: The multicast group number is out of range.
0x112: The multicast group has not been defined.

0x501: The TMID is out of range.
0x502: The order of issuing macros is not correct.
0x503: The type of the transfer memory is not correct.
0x504: An error has been found in the specified transfer memory control flags.
0x505: The transfer memory cycle is not correct.
0x506: The number of transmission areas is out of range.
0x507: The block number is out of range.
0x508: The number of blocks is out of range.
0x509: The specified address is not on a two-byte boundary.
0x50A: Transmission areas in the local node are overlapped.
0x511: A transmission area is defined outside the transfer memory (mismatch between m_blkno and m_blkcnt).
0x513: Transfer memory is outside the global area.

[Note]
The following table shows the differences among software transfer parameters.

Table 3-2 Parameter List

Transfer type / Parameter	Periodic transmission type software transfer	Split transmission type software transfer
Parameter	Configurable range	
df	1 to 255	
tmid	1 to 4	
m_type	TMEM_SOFT	TMEM_FAST
m_mcg	1 to 255	
m_cflags	1 to 3	0 and 2
m_memid	GLB address	
m_mincycle	–	
m_caseno	0 to 9	
m_maxblk	1 to 16384	
m_cyctm	milliseconds (*)	
m_blkno	0 to 16383	
m_blkcnt	1 to 255	

The parameter indicated as "-" is invalid.
Specify 0 for the parameter.

(*) Specify a value between 100 ms and 86400000 ms for periodic-transmission type software transfer.
Specify a value between 10 ms and 86400000 ms for split-transmission type software transfer.

4.2 nx_ctl_tm

[Name]

nx_ctl_tm - Control transfer memory

[Syntax]

```
#include  <nxacp.h>
long   nx_ctl_tm(df, tmid, cmd, hrtn)

long   *df, *tmid,*cmd,*hrtn;
```

[Parameters]

df: Data field number
tmid: Transfer memory identifier
cmd: Memory transfer type

TM_START(2): Starts transmission and reception for the specified TMID.
TM_READ(1): Starts only reception for the specified TMID.
TM_STOP(0): Stops the transfer for the specified TMID.

hrtn: Area used for storing an error cause code generated during hardware transfer (No value can be stored because hardware transfer is not supported.)

[Description of the function]

This macro has the following transfer memory control functions.
Before you start or stop transfer, you must initialize the transfer memory using the nx_init_tm macro.

- Transfer start
  Starts both transmission and reception for transfer memory.
- Reception start
  Starts only reception for transfer memory.
- Transfer stop
  Stops transfer.

[Return value]
When nx_ctl_tm() finishes normally, 0 is returned.
If nx_ctl_tm() terminates with an error, the following value is returned.

0x001: NXACP was stopped.
0x003: Control tables are not loaded.

0x101: The data field number is out of range.
0x102: The data field is not defined.
0x105: The data field is a remote data field.0x501: The TMID is out of range.

0x502: The order of issuing macros is not correct.
0x504: An error was found in the specified transfer memory control flags.

4.3 nx_get_tm

[Name]

nx_get_tm - Get the transfer memory information

[Syntax]

```
#include    <nxacp.h>
long    nx_get_tm(df, tmid, tmmap)

long    *df, *tmid;
struct tmmap    *tmmap;
```

[Parameters]

df: Data field number of the data field you want to get the information for (between 1 and maximum DF).

tmid: Transfer memory identifier of the transfer memory area you want to get the information for (between 1 and maximum TMID).

tmmap: Start address of the area used for storing the retrieved information

[Description of the function]

This macro retrieves the attributes of the transfer memory to the specified area.
For information about the tmmap structure, see “4.1 nx_init_tm”.

[Return value]

When nx_get_tm() finishes normally, the retrieved values are stored in tmmap, and 0 is returned.
If nx_get_tm() terminates with an error, the following value is returned.

0x001: NXACP has stopped.
0x003: Control tables are not loaded.

0x101: The data field number is out of range.
0x102: The data field is not defined.
0x105: The data field is remote.

0x501: The TMID is out of range.
0x502: Transfer memory ID has not been initialized.

4.4 nx_write_tm

[Name]

nx_write_tm - Write to transfer memory

[Syntax]

```
#include   <nxacp.h>
long   nx_write_tm(df, tmid, offset, da, dc, hrtn)

long    *df;
long    *tmid;
long    *offset;
long    *da;
long    *dc;
long    *hrtn;
```

[Parameters]

df: Data field number of the data field for the transfer memory you want to write to.
tmid: Transfer memory identifier (TMID) of the transfer memory area you want to write to.
offset: Write position in the transfer memory area measured in bytes from the beginning (The position must be at a two-byte boundary.)
da: User data area address (The address must be at a 2 byte boundary.)
dc: User data area size
hrtn: Area used for storing an error cause code generated during hardware transfer (No value can be stored because hardware transfer is not supported.)

[Description of the function]

This macro writes the data specified by da and dc to the transfer area specified by df and tmid.
If software transfer is used with mutual exclusion enabled, data consistency for up to 16320 bytes is guaranteed.
If software transfer is used with mutual exclusion disabled, or if high-speed software transfer is used, data consistency for four bytes is guaranteed only if offset, da, and dc are all multiples of four.

[Return value]
When nx_write_tm() finishes normally, 0 is returned.
If nx_write_tm() terminates with an error, the following value is returned.

0x001: NXACP has been stopped.
0x003: Control tables are not loaded.

0x101: The data field number is out of range.
0x102: The data field is not defined.
0x105: The data field is remote.
0x141: There is an error in the specified offset size.

0x501: The TMID is out of range.
0x502: The order of issuing macros is not correct.
0x50B: There is an error in the specified da.
0x50C: There is an error in the specified dc.
0x531: Failed to allocate transfer memory.

4.5 nx_read_tm

[Name]

nx_read_tm - Read from transfer memory

[Syntax]

```
#include   <nxacp.h>
long   nx_read_tm(df, tmid, offset, da, dc, hrtn)

long     *df;
long     *tmid;
long     *offset;
long     *da;
long     *dc;
long     *hrtn;
```

[Parameters]

df: Data field number of the data field for the transfer memory you want to read from.
tmid: Transfer memory identifier (TMID) of the transfer memory area you want to read from.
offset: Read position in the transfer memory area measured in bytes from the beginning (The position must be at a two-byte boundary.)
da: User data area address used for reading (The address must be at a 2-byte boundary.)
dc: User data area size used for reading.
hrtn: Area used for storing an error cause code generated during hardware transfer (No value can be stored because hardware transfer is not supported.)

[Description of the function]

This macro reads the data specified by da and dc from the transfer area specified by df and tmid.
If software transfer is used with mutual exclusion enabled, data consistency for up to 16320 bytes is guaranteed. If software transfer is used with mutual exclusion disabled, or if high-speed software transfer is used, data consistency for four bytes is guaranteed only if offset, da, and dc are all multiples of four.

[Return value]
When nx_read_tm() finishes normally, 0 is returned.
If nx_read_tm() terminates with an error, the following value is returned.

0x001: NXACP has stopped.
0x003: Control tables are not loaded.

0x101: The data field number is out of range.
0x102: The data field is not defined.
0x105: The data field is remote.
0x141: There is an error in the specified offset size.

0x501: The TMID is out of range.
0x502: The order of issuing macros is not correct.
0x50B: There is an error in the specified da.
0x50C: There is an error in the specified dc.
0x512: The specified dc means usage outside transfer memory.
0x531: Failed to allocate transfer memory.

APPENDIXES

APPENDIX A RETURN CODE DETAILS

(1) Return codes of multicast communication macros

(1/2)

Code	Description	User action
0x0001	NXACP has not been initialized or has been stopped.	Initialize NXACP.
0x0002	NXACP has already been initialized.	NXACP has already been initialized. (Initialization is not required.)
0x0003	Control tables have not been loaded.	Construct and load construction information.
0x0004	The NXACP tasks have not been loaded or the task startup failed.	Reload the main part of NXACP. (*1)
0x0101	The specified data field number is out of range.	Check the data field number, and then retry.
0x0102	The specified data field is not defined.	Check the data field number, and then retry.
0x0103	The specified data field has not been initialized.	Check the data field number, and then retry.
0x0104	The specified data field has already been initialized.	Check the data field number, and then retry.
0x0105	There is an error in the specified data field.	Check the data field number, and then retry.
0x0106	The source data field has not been defined.	Check the source data field number, and then retry.
0x0107	The source data field has not been initialized.	Check the source data field number, and then retry.
0x0108	The DF at the destination PU of inter-PU communication has not been initialized.	Initialize the DF at the destination PU, and then retry.
0x0111	The specified multicast group number is out of range.	Check the multicast group number, and then retry.
0x0112	The specified multicast group has not been defined.	Check the multicast group number, and then retry.
0x0121	The specified transaction code is out of range.	Check the transaction code, and then retry.
0x0131	The specified message/buffer size is out of range.	Check the message/buffer size, and then retry.
0x0141	The specified offset size is out of range.	Check the specified offset size, and then retry.
0x0151	The specified timeout monitoring time is out of range.	Check the timeout, and then retry.
0x0161	Node mode setting error	Check the node mode, and then retry.
0x0162	Message mode setting error	Check the message mode, and then retry.
0x01E1	Device type or IP address error	Check the unit number and IP address, and then retry.
0x01E3	Network combination error in duplexed LANs.	Check the device for each UNO, and then retry.
0x01E6	All modules have failed.	Replace the modules, and then retry.
0x01F1	Socket macro error	For details, see the General Description manual. (*2)

(2/2)

Code	Description	User action
0x0201	No space available in the send buffer.	Please retry.
0x0202	The data field has been down during the reception process.	Please retry.
0x0211	Transmission not possible.	Replace the module.
0x0231	An error has been detected in inter-PU communication.	Please retry.
0x02E1	The NXACP state is currently transitioning.	Please retry.
0x02F1	The state of the specified data field is currently transitioning.	Please retry.
0x02F2	The state of the source data field is currently transitioning.	Please retry.
0x02F3	The state of a remote data field is currently transitioning.	Please retry.
0x0301	The task number of the task that issued the macro is out of range.	The task number of the task that can issue the macro is between 1 and 208.
0x0302	Reception TCDs have not been defined.	Define reception TCDs for the task.
0x0303	Transmission TCDs have not been defined.	Check and correct the transmission permission check TCD definition, and then retry.

(*1) 0x0004 includes a task startup failure. The task startup involves QUEUEs and TIMERs.
For TIMER, two tasks are registered to the timer in NXACP. Consequently, if the number of tasks users have registered to the timer reaches the system limit before NXACP starts, the task startup is aborted due to this error.

(*2) The return code from RCTLNET is saved in the detailed code storage area. Check this return code in *S10VE User’s Manual General Description* (manual number SEE-1-001).

<Correspondence between return codes and macros>

Code	nx_init	nx_dfup	nx_dfdwn	nx_quit	nx_put	nx_get		INFO1	INFO2	INFO3
0x0001	–	Y	Y	Y	Y	Y		–	–	–
0x0002	Y	–	–	–	–	–		–	–	–
0x0003	Y	Y	Y	Y	Y	Y		–	–	–
0x0004	Y	Y	–	–	–	–		TN	–	–
0x0101	–	Y	Y	–	–	–		–	–	–
0x0102	–	Y	Y	–	Y	–		–	–	–
0x0103	–	–	Y	–	Y	–		–	–	–
0x0104	–	Y	–	–	–	–		–	–	–
0x0105	–	–	–	–	–	–		–	–	–
0x0106	–	Y	–	–	–	–		–	–	–
0x0107	–	Y	–	–	–	–		–	–	–
0x0108	–	–	–	–	Y	–		–	–	–
0x0111	–	–	–	–	Y	–		–	–	–
0x0112	–	–	–	–	Y	–		–	–	–
0x0121	–	–	–	–	Y	–		–	–	–
0x0131	–	–	–	–	Y	Y		–	–	–
0x0141	–	–	–	–	–	Y		–	–	–
0x0151	–	–	–	–	–	Y		–	–	–
0x0161	–	Y	–	–	–	–		–	–	–
0x0162	–	Y	–	–	–	–		–	–	–
0x01E1	Y	–	–	–	–	–		UNO	–	–
0x01E3	Y	–	–	–	–	–		DFNO	–	–
0x01E6	-	Y	–	–	–	–		UNO1	UNO2	–
0x01F1	Y	Y	–	–	–	–		UNO	MCODE	ERRNO
0x0201	–	–	–	–	Y	–		–	–	–
0x0202	–	–	–	–	–	Y		–	–	–
0x0211	–	–	–	–	Y	–		–	–	–
0x0231	–	–	–	–	Y	–		–	–	–
0x02E1	Y	Y	Y	Y	–	–		–	–	–
0x02F1	–	Y	Y	Y	–	–		–	–	–
0x02F2	–	Y	Y	–	–	–		–	–	–
0x02F3	–	–	Y	–	–	–		–	–	–
0x0301	–	–	–	–	Y	Y		–	–	–
0x0302	–	–	–	–	–	Y		–	–	–
0x0303	–	–	–	–	Y	–		–	–	–

Y: This error code can be returned. –: This error code is never returned.
INFO1 to 3 are parameters of nx_init() and nx_dfup(), and contain detailed information for when the macro returns with an error.
TN: Task number
UNO: Unit number (When the code is 0x1E1, the UNOs of two LANs are stored in the upper 2 bytes and the lower 2 bytes.)
MCODE: RCTLNET macro code
SOCKET: 1
BIND: 2
GETSOCKOPT: 3
SETSOCKOPT: 4
ERRNO: RCTLNET macro return code (*)
SLOTNO: Slot number
DFNO: Data field number
(*) For details about the return code of the macro, refer to *S10VE User's Manual General Description* (manual number SEE-1-001).

[Error code system]

The error code system for multicast communication macros is as follows. Note that X denotes any numerical character.

0x00XX: Errors generated when installation has not been finished or installation has failed
0x01XX: Errors generated by a parameter check
Hardware errors are also included for initialization requests (nx_init() and nx_dfup()).
0x02XX: Timing errors These errors may be resolved by simply retrying.
However, be cautious about the timing of retry.
If you immediately retry and the same timing error is generated, an infinite loop might result.
0x03XX: Errors in the user's system design

(2) Return codes of shared memory macros

Code	Description	User action
0x0501	TMID out of range	Check and correct the TMID, and then retry.
0x0502	Macro calling procedure error	Check and correct the calling procedure for each macro, and then retry.
0x0503	Transfer type specification error	Check and correct the transfer type, and then retry.
0x0504	Transfer control flag specification error	Check and correct the transfer control flags, and then retry.
0x0505	Transfer cycle specification error	Check and correct the transfer cycle, and then retry.
0x0506	Transmission area count out of range	Check and correct the number of transmission areas, and then retry.
0x0507	Block number out of range	Check and correct the block number, and then retry.
0x0508	Block count out of range	Check and correct the block number, and then retry.
0x0509	Specified address specification error (odd address)	Check the specified address, and then retry.
0x050A	Overlapped transmission areas in the local node	Check and correct the transmission areas, and then retry.
0x050B	DA specification error	Check and correct the DA, and then retry.
0x050C	DC specification error	Check and correct the DC, and then retry.
0x0531	Failed to allocate transfer memory (rserv failed)	Make sure that the number of tasks that access the transfer area is 15 or less.
0x0511	A transmission area is defined outside the transfer memory.	Check and correct blkno and blkcnt, and then retry.
0x0512	The specified dc means using outside the transfer memory.	Check and correct the DC, and then retry.
0x0513	Transfer memory is outside the global area.	Check the specified address, and then retry.

[Error code system]

The error code system for shared memory macros is as follows. Note that X denotes any numerical character.

0x050X: Errors generated by a parameter check

0x053X: System operational errors

<Correspondence between return codes and macros>

Code	nx_init_tm	nx_ctl_tm	nx_get_tm	nx_write_tm	nx_read_tm
0x0001	Y	Y	Y	Y	Y
0x0003	Y	Y	Y	Y	Y
0x0101	Y	Y	Y	Y	Y
0x0102	Y	Y	Y	Y	Y
0x0103	Y	–	–	–	–
0x0105	Y	Y	Y	Y	Y
0x0111	Y	–	–	–	–
0x0112	Y	–	–	–	–
0x0141	–	–	–	Y	Y
0x0501	Y	Y	Y	Y	Y
0x0502	Y	Y	Y	Y	Y
0x0503	Y	–	–	–	–
0x0504	Y	Y	–	–	–
0x0505	Y	–	–	–	–
0x0506	Y	–	–	–	–
0x0507	Y	–	–	–	–
0x0508	Y	–	–	–	–
0x0509	Y	–	–	–	–
0x050A	Y	–	–	–	–
0x050B	–	–	–	Y	Y
0x050C	–	–	–	Y	Y
0x0511	Y	–	–	–	–
0x0512	–	–	–	–	Y
0x0513	Y	–	–	–	–
0x0531	–	–	–	Y	Y

Y: This error code can be returned.　–: This error code is never returned.

(3) List or error codes of the construction command (confnxsv)

- The message format of the output from the confnxsv command is as follows.

```
Error/Warning: Failure message Failure No.
```

- The details are shown in the following table.

If you execute the tblldnxsv command without resolving the error, tblldnxsv also terminates with an error.

Failure No.	Failure message	Description of the failure
1300	AAA is out of range (file:BBB, line:CCC)	AAA is out of range. (BBB: File name, CCC: Line number)
1302	AAA is multiple defined (file:BBB)	Multiple instances of AAA are defined. (BBB: File name)
1303	AAA is undefined (file:BBB)	AAA is not defined. (BBB: File name)
1330	Illegal format of name (file:BBB, line:CCC) “AAA”	The name “AAA” does not follow the correct format. (BBB: File name, CCC: Line number)
1331	Illegal format of numeric value (file:BBB, line:CCC)	The numerical value does not follow the correct format. (BBB: File name, CCC: Line number)
1332	Not enough entries (file:BBB, line:CCC)	Sufficient entries have not been provided for the line. (BBB: File name, CCC: Line number)
1333	Max over entries (file:BBB) “AAA”	“AAA” exceeds the maximum value. (BBB: File name)
1356	Resource data unmatch (file:BBB) “AAA”	Discrepancies are found in “AAA”. (BBB: File name)
1651	Specified AAA is already defined (file:BBB, line:CCC)	AAA is already defined. (BBB: File name, CCC: Line number)
2653	Specified file is for AAA (file:BBB)	The specified file is used for AAA. (BBB: File name)
9001	Specified size is not on AAA byte boundary (file:BBB, line:CCC)	The specified value is not a multiple of AAA. (BBB: File name, CCC: Line number)
9002	Can’t open file “BBB”	The BBB file could not be opened.

If an error occurs with construction information on the CP side, check the details of the failure, and then retry the operation.
If an error occurs with construction information on the HP side, set “0” to MAXDFCNT (the definition information in `nxacp.conf` on the HP side), and then retry the operation.

APPENDIX B　LOG FORMAT

The following tables show the error information format to be reported to EAS.

(1) Error notification list

Event summary	Code	Contents of error notification	User handling
Buffer status report	0x0401	The send/receive buffer usage reached the threshold value or recovered.	Increase the number of buffer cases.
Protocol error	0x0201	An error was detected in the NX header of the received message and the corresponding message was discarded.	Check the sender node and construction information.
Socket error	0x0501	An error was reported from the socket macro.	Report to the system manager.
Transfer area duplication error	0x0601	Duplication in the transmission area for transfer was detected (only for software transfer).	Review the send address for transfer.

[Note]
The following table shows the check items for reporting a protocol error (0x0201):

Check item	Discard condition	Notification	Remarks
Receive size	1473 bytes or more (maximum)	Yes	(*1)
Received message	Pattern (NUXM) error	Yes	
Transmission destination address	Inconsistent DF	Yes	Check DA
	Inconsistent MGN	Yes	Check DA
Sender address	Undefined sender node	None	(*2)
Receive MCG	Undefined MGN	None	
TCD	Undefined TCD	None	
MODE	Inconsistent port mode	Yes	

(*1) When the received NX header is 63 bytes or less, notification is not performed (only discarded in NXACP).
(*2) Undefined nodes (out of range) are not reported. It is determined that messages from the corresponding nodes do not need to be received.

(2) EAS input data format (ADB)
For details, refer to *S10VE Software Manual CPMS General Description and Macro Specifications* (manual number SEE-3-201).

	Name	Description
0	logno	Error log number
4	timestamp	Time
8	type	Severity indication (depending on the error type)
12	class	Error detection component class (0x2011 for NXACP)
16	retcode	Disabled (fixed to 0 for NXACP)
20	errtype	Error type (fixed to 10 for NXACP)
22	flag	Error message flag (fixed to 0 for NXACP)
24	site[16]	Site name
40	erb[117] (*)	Error block (error report data). Although the area size is fixed to 468 bytes, the size of valid data differs depending on the error type.
512	dhpbuf[128]	DHP data (512 bytes)
1024		

(*) erb is configured in the order of formtype, size, errorcode, detailed data 1, and data 2... from the beginning, with each item being four bytes.

(3) List of error detailed data formats

Name	Buffer status report	Protocol error	Socket error	Transfer area duplication error
formtype in erb	0x0103	0x0102	0x0104	0x0105
size in erb	64	76	16	24
errorcode in erb	0x0401	0x0201	0x0501	0x0601
Detailed data (each information item is four bytes)	DFN	DFN	DFN	DFN
	SPEAK	PORTNO	DADDR	TMID
	RPEAK	NXHD (*)	DPORT	CASENO
	SOVFCCNT			BLKNO
	SHWCCNT			BLKCNT
	SLWCCNT			
	ROVFCCNT			
	RHWCCNT			
	RLWCCNT			
	SOVFTCNT			
	SHWTCNT			
	SLWTCNT			
	ROVFTCNT			
	RHWTCNT			
	RLWTCNT			
type	NOTE	WARNING	NONFATAL	NONFATAL

(*) NXHD is in a 64-byte configuration and indicates the NX header. For details about the configuration, refer to “APPENDIX F　MESSAGE HEADER FORMAT”.

DFN: Data Field Number
SPEAK: Send Buffer Peak Count
RPEAK: Receive Buffer Peak Count
SOVFCCNT: Send Buffer Overflow Current Count
SHWCCNT: Send Buffer High-Water Current Count
SLWCCNT: Send Buffer Low-Water Current Count
ROVFCCNT: Receive Buffer Overflow Current Count
RHWCCNT: Receive Buffer High-Water Current Count
RLWCCNT: Receive Buffer Low-Water Current Count
SOVFTCNT: Send Buffer Overflow Total Count
SHWTCNT: Send Buffer High-Water Total Count
SLWTCNT: Send Buffer Low-Water Total Count
ROVFTCNT: Receive Buffer Overflow Total Count
RHWTCNT: Receive Buffer High-Water Total Count
RLWTCNT: Receive Buffer Low-Water Total Count
PORTNO: Port Number
DADDR: Destination Address
DPORT: Destination Port
TMID: Transfer Memory Identifier
CASENO: Case Number
BLKNO: Block Number
BLKCNT: Block Count
MERRNO: Macro Error Code

(4) Display format

• Buffer status report

```
%NX-I-SOFT-0103   SITE=xxxxxxxxxxxxxxx   RC=00000000   yyyy/mm/dd hh:mm:ss   LOG=xxx
 EC=000000401 Buffer status
 DFN =XXXXXXXX   SPEAK =XXXXXXXX   RPEAK =XXXXXXXX
 SOVFCCNT  =XXXXXXXX   SHWCCNT  =XXXXXXXX   SLWCCNT  =XXXXXXXX
 ROVFCCNT  =XXXXXXXX   RHWCCNT =XXXXXXXX   RLWCCNT  =XXXXXXXX
 SOVFTCNT  =XXXXXXXX   SHWTCNT  =XXXXXXXX   SLWTCNT  =XXXXXXXX
 ROVFTCNT  =XXXXXXXX   RHWTCNT  =XXXXXXXX   RLWTCNT  =XXXXXXXX
～
```

RC: Return Code
EC: Error Code
DFN: Data Field Number
SPEAK: Send Buffer Peak Count
RPEAK: Receive Buffer Peak Count
SOVFCCNT: Send Buffer Overflow Current Count
SHWCCNT: Send Buffer High-Water Current Count
SLWCCNT: Send Buffer Low-Water Current Count
ROVFCCNT: Receive Buffer Overflow Current Count
RHWCCNT: Receive Buffer High-Water Current Count
RLWCCNT: Receive Buffer Low-Water Current Count
SOVFTCNT: Send Buffer Overflow Total Count
SHWTCNT: Send Buffer High-Water Total Count
SLWTCNT: Send Buffer Low-Water Total Count
ROVFTCNT: Receive Buffer Overflow Total Count
RHWTCNT: Receive Buffer High-Water Total Count
RLWTCNT: Receive Buffer Low-Water Total Count

• Protocol error

```
%NX-W-SOFT-0102   SITE=xxxxxxxxxxxxxxx   RC=00000000   yyyy/mm/dd hh:mm:ss   LOG=xxx
EC=000000201 Message frame error
DFN      =XXXXXXXX   PORTNO=XXXXXXXX
H_TYPE  =XXXXXXXX   ML  =XXXXXXXX   SA      =XXXXXXXX   DA =XXXXXXXX
V_SEQ   =XXXXXXXX   SEQ =XXXXXXXX   M_CTL =XXXXXXXX
INQ_ID1 =XXXXXXXX   INQ_ID2 =XXXXXXXX   INQ_ID3 =XXXXXXXX
TCD      =XXXXXXXX   G_TID1  =XXXXXXXX   G_TID2  =XXXXXXXX
MSGMD =XXXXXXXX   BLOCK  =XXXXXXXX   FU1      =XXXXXXXX
～
```

EC: Error Code
DFN: Data Field Number
PORTNO: Port Number

The following information items indicate the NX header.
For details about the configuration, refer to “APPENDIX F　MESSAGE HEADER FORMAT”.
H_TYPE: HEAD Pattern
ML: Message Length
SA: Source Address
DA: Destination Address
V_SEQ: Version Sequence number
SEQ: Send Sequence number
M_CTL: Message Control
INQ_ID1: Inquiry Identifier1
INQ_ID2: Inquiry Identifier2
INQ_ID3: Inquiry Identifier3
TCD: Transaction Code
G_TID1: Global Transaction Identifier1
G_TID2: Global Transaction Identifier2
MSGMD: Message Mode
BLOCK: Message Block
FU1: Future Use

- Socket error

```
%NX-E-SOFT-0104    SITE=xxxxxxxxxxxxxxx    RC=00000000    yyyy/mm/dd hh:mm:ss    LOG=xxx
 EC=000000501 Socket error
 DFN    =XXXXXXXX    DADDR=XXXXXXXX    DPORT=XXXXXXXX
～
```

RC: Return Code
EC: Error Code
DFN: Data Field Number
DADDR: Destination IP Address
DPORT: Destination Port Number

- Transfer area duplication error

```
%NX-E-SOFT-0105    SITE=xxxxxxxxxxxxxxx    RC=00000000    yyyy/mm/dd hh:mm:ss    LOG=xxx
 EC=000000601 Transfer memory address error
 DFN      =XXXXXXXX    TMID      =XXXXXXXX    CASENO =XXXXXXXX
 BLKNO =XXXXXXXX    BLKCNT =XXXXXXXX
～
```

RC: Return Code
EC: Error Code
DFN: Data Field Number
TMID: Transfer Memory Identifier
CASENO: Send Case Number
BLKNO: Send Block Number
BLKCNT: Send Block Count

APPENDIX C NODE STATUS CHANGE NOTIFICATION FORMAT

A status change notified from other nodes links to the notification IRSUB reserved by NXACP. The IRSUB number of the notification IRSUB is 332. By registering a subroutine to IRSUB 332, a user can detect the status change of nodes in realtime. You must register the subroutine before you start NXACP.

(1) Notification format

	Name	Description
0	CODE	Event number
4	DFN	Data field number
8	LNN	Logical node number
12	LANNO	LAN number
16	TYPE	Change cause type
20		

(2) Detailed information

Alive notification: An alive signal is received for the first time, or an alive signal is received again after the node is detected as dead.

Dead notification: No alive signals are received in the predefined time after an alive signal is received.

Scheduled dead notification: An alive signal for scheduled SHUTDOWN or maintenance has been received.

	Alive notification	Dead notification	Scheduled dead notification
0	0x0301	0x0302	0x0302
4	DFN	DFN	DFN
8	LNN	LNN	LNN
12	LANNO (*)	LANNO (*)	LANNO (*)
16	Fixed to 0	Fixed to 2	Fixed to 1
20			

(*) The value is fixed to 1 for single LAN configuration. The value is also fixed to 1 for the local node.
For duplexed LAN configuration, specify 1 for the UNO1 LAN, and specify 2 for the UNO2 LAN.
For the local node, the value is fixed to 1, and reports a notification only once.

[Precaution]
This IRSUB runs as a subroutine of an NXACP system task.
For this reason, from this IRSUB, you must not call a macro that waits inside. The stack size this IRSUB can use is up to 512 bytes.

APPENDIX D DHP RECORD LIST

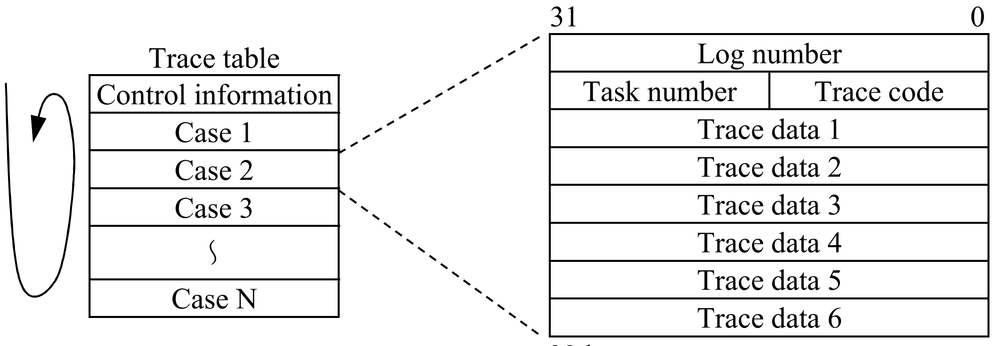
Recording point	Code	Data count	Data 1	Data 2	Data 3	Data 4
nx_init() start	0x00400001	0	–	–	–	–
nx_init() completion	0x00400021	1	RTNCD	–	–	–
nx_dfup() start	0x00400002	3	DFN	NMODE	MMODE	–
nx_dfup() completion	0x00400022	2	DFN	RTNCD	–	–
nx_dfdwn() start	0x00400003	1	DFN	–	–	–
nx_dfdwn() completion	0x00400023	1	RTNCD	–	–	–
nx_quit() start	0x00400004	0	–	–	–	–
nx_quit() completion	0x00400024	1	RTNCD	–	–	–
nx_put() start	0x00400005	4	DFN	MGN	TCD	LEN
nx_put() completion	0x00400025	1	RTNCD	–	–	–
nx_get() start	0x00400006	3	TMOUT	BFSZ	OFSET	–
nx_get() completion (*)	0x00400006	4	RTNCD	SA	DA	TCD/LEN
Completion of data reconstruction after reception	0x00100009	2	POSTA	Fixed to 0	–	–
Incomplete data reconstruction after reception	0x00100009	2	Fixed to 0	Fixed to 0	–	–
Reception of transfer data	0x00100009	2	Fixed to 0	Fixed to 1	–	–
Reception of an alive signal	0x00100009	2	Fixed to 0	Fixed to 2	–	–
Discarding a message due to header error	0x00100009	2	Fixed to 0	Fixed to 3	–	–

The meaning of each data item is as follows:
RTNCD: Processing result
DFN: Data field number
NMODE: Node mode
MMODE: Message mode
MGN: Multicast group number
TCD: Transaction code
LEN: Message size
TMOUT: Timeout monitoring time
BFSZ: Buffer size
OFSET: Offset
SA: Sender address
DA: Transmission destination address
POSTA: POST address
(*) For nx_get completion, data 1 to 4 are set to 0 if a message is not available or if nx_get returns an error.

APPENDIX E CONTROL TRACE

NXACP provides a RAS feature for logging the internal operation of NXACP.
This log is called a “control trace”. You can use this trace data to analyze an NXACP-related failure.
A user can decide whether to use a control trace. You can turn on a control trace during a test. (Disable it during online operation to allow for high-speed execution.) We recommend that you allocate about 512 cases (16 KB) of memory for a control trace.
To display the trace, use the dump command (sd) offered by RPDP.

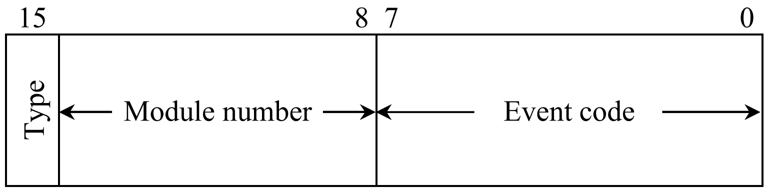
(1) Trace area format
The format of the trace area is as follows. The cases whose number is specified in the construction information are used cyclically.



32 bytes per case

The cases whose number is specified in the construction information are used cyclically.

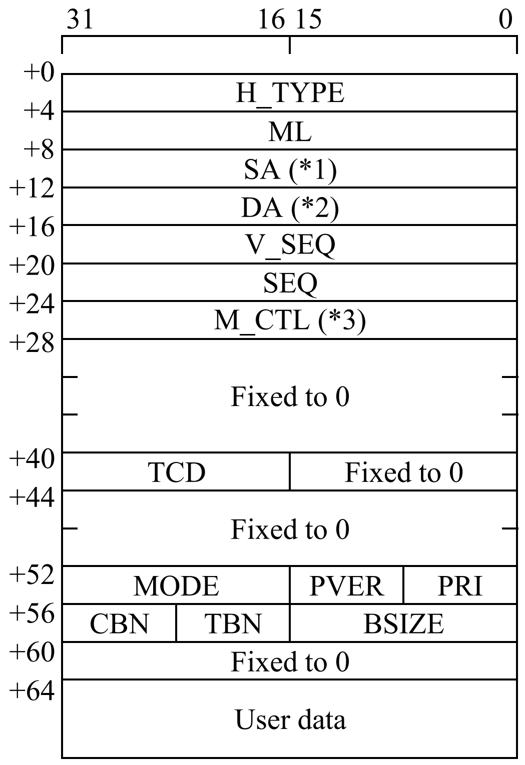
(2) Trace area format



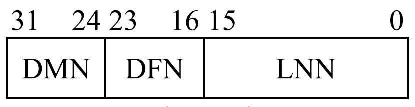
Type: This bit is set during error tracing.
Module number: Indicated as follows.

1: nx_rcv	16: nx_put	32: nx_init
2: nx_upexe	17: nx_get	33: nx_quit
3: nx_htim	18: nx_abs	34: nx_dfup
4: nx_ltim		35: nx_dfdwn
5: nx_dsnd		36: nx_ins
6: nx_memac		37: nx_cdon
7: nx_cycsnd		38: nx_cdoff
8: nx_purcv		39: nx_ctl
10: nx_init_tm		
11: nx_ctl_tm		
12: nx_get_tm		
13: nx_write_tm		
14: nx_read_tm		

APPENDIX F MESSAGE HEADER FORMAT

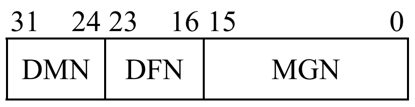


(*1) SA



DMN: Domain number
(Currently fixed to 0)
DFN: Data field number
LNN: Sender node number

(*2) DA



DMN: Domain number
(Currently fixed to 0)
DFN: Data field number
MGN: Multicast group number

(*3) M_CTL: Fixed to 0x80000000
(Multicast communication)

Symbol name	Size (in bytes)	Description
H_TYPE	4	Header type (Set to “NUXM” in ASCII.)
ML	4	Length of the message including the NX header
SA	4	Message sender address See (*1).
DA	4	Message destination address. See (*2).
V_SEQ	4	Version of sequence number (Set to the time when the sequence number is initialized.)
SEQ	4	Message transmission sequence number (0x00000001 to 0x7FFFFFFF)
M_CTL	4	Message transmission control information. See (*3).
TCD	2	Transaction code
MODE	2	Message mode (0: Online mode, 1: Test mode)
PVER	1	NX protocol version (Fixed to 1)
PRI	1	The priority level of messages (Fixed to 0)
CBN	1	Current block number
TBN	1	Total block count
BSIZE	2	User data size in a split packet (Byte count in a fragment block)

APPENDIX G　ALIVE SIGNAL HEADER FORMAT

	31		16 15	0
+0	H_TYPE			
+4	ML (Fixed to 128)			
+8	SA (*1)			
+12	DA (*2)			
+16	V_SEQ			
+20	SEQ			
+24	M_CTL (*3)			
+28	Fixed to 0			
+40	TCD (Fixed to 60003)		Fixed to 0	
+44	Fixed to 0			
+52	MODE		PVER (Fixed to 1)	PRI (Fixed to 1)
+56	CBN (Fixed to 1)	TBN (Fixed to 1)	BSIZE (Fixed to 128)	
+60	Fixed to 0			
+64	AL_ND_NAME			
+76	AL_OS_NAME			
+84	AL_TM_OUT			
+88	AL_PROTOCOL	AL_MODE	AL_MSGSERNO	
+92 +96	Fixed to 0			
+100	AL_CHG_TIME			
+104 +108	AL_IPADDR[2]			
+112 +116	Fixed to 0			AL_VER (1)
+128				

(*1) SA

31　24	23　16	15　0
DMN	DFN	LNN

DMN: Domain number
(Currently fixed to 0)
DFN: Data field number
LNN: Sender node number

(*2) DA

31　24	23　16	15　0
DMN	DFN	MGN

DMN: Domain number
(Currently fixed to 0)
DFN: Data field number
MGN: Multicast group number

(*3) M_CTL: Fixed to 0x80000000 (Multicast communication)

Symbol name	Size (in bytes)	Description
H_TYPE	4	Header type (set to “NUXM” in ASCII)
ML	4	Length of the message including the NX header (Fixed to 128)
SA	4	Message sender address. See (*1).
DA	4	Message destination address. See (*2).
V_SEQ	4	Version of sequence number (set to the time when the sequence number is initialized)
SEQ	4	Message transmission sequence number (0x00000001 to 0x7FFFFFFF)
M_CTL	4	Message transmission control information (Fixed to 0x80000000)
TCD	2	Transaction code (Fixed to 60003)
MODE	2	Message mode (0: Online mode, 1: Test mode)
PVER	1	NX protocol version (Fixed to 1)
PRI	1	The priority level of messages (Fixed to 1)
CBN	1	Current block number (Fixed to 1)
TBN	1	Total block count (Fixed to 1)
BSIZE	2	User data size in a split packet (Fixed to 128)
AL_ND_NAME	10	Node name (ASCII string that ends with NULL. Up to 9 characters.)
AL_OS_NAME	10	Vendor device name (HI_S10VE)
AL_TM_OUT	4	Alive signal timeout monitoring time (in seconds)
AL_MSGSERNO	2	Alive report message sequence number
AL_MODE	1	Alive report mode. See (*4).
AL_PROTOCOL	1	Protocol type (Fixed to 4)
AL_CHG_TIME	4	Timestamp of node status change. See (*5).
AL_IPADDR	8	IP address (AL_IPADDR[0] for LAN1 and AL_IPADDR[1] for LAN2)
AL_VER	1	Alive signal message version number (Fixed to 1)

(*4) AL_MODE = 1: Normal mode (Alive status)
= 2: Scheduled shutdown mode

(*5) AL_CHG_TIME: Greenwich time of startup. Denoted in the Year, Month, Day, Hour, Minute, and Second format. When stopped, 0 is set.