60459990

CONTROL DATA CORPORATION

REMOTE HOST FACILITY ACCESS METHOD REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 2
NOS/BE 1

PRELIMINARY EDITION

REVISION RECORD

REVISION	DESCRIPTION
01 (05-09-83)	Preliminary manual released at NOS 2.1, PSR level 580 and NOS/BE 1.5, PSR level 577.
A (03-29-85)	Manual released at NOS 2.4.1, PSR level 630 and NOS/BE 1.5, PSR level 627. This manual has been revised to support NAD code conversion and miscellaneous technical and editorial changes. This edition obsoletes all previous editions.

Publication No.
60459990

REVISION LETTERS I, O, Q, S, X AND Z ARE NOT USED.

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV
Front Cover	-
Title Page	-
2	A
3/4	A
5	A
6	A
7	A
8	A
9	A
1-1	A
1-2	A
1-3	A
1-4	A
1-5	A
1-6	A
1-7	A
1-8	A
2-1	A
2-2	A
2-3	A
2-4	A
2-5	A
2-6	A
2-7	A
2-8	A
2-9	A
2-10	A
2-11	A
2-12	A
2-13	A
3-1	A
3-2	A
3-3	A
3-4	A
3-5	A
3-6	A
3-7	A
3-8	A
3-9	A
3-10	A
3-11	A
3-12	A
3-13	A
3-14	A
3-15	A
3-16	A
3-17	A
3-18	A
3-19	A
3-20	A
3-21	A
3-22	A
3-23	A
3-24	A
3-25	A
3-26	A
3-27	A
3-28	A
3-29	A
3-30	A
3-31	A
3-32	A
3-33	A
3-34	A
3-35	A
3-36	A
3-37	A
3-38	A
3-39	A
3-40	A
3-41	A
3-42	A
3-43	A
3-44	A
3-45	A
3-46	A
3-47	A
4-1	A
4-2	A
4-3	A
4-4	A
4-5	A
4-6	A
4-7	A
4-8	A
4-9	A
4-10	A
4-11	A
4-12	A
4-13	A
4-14	A
4-15	A
4-16	A
4-17	A
4-18	A
4-19	A
4-20	A
4-21	A
4-22	A
4-23	A
4-24	A
4-25	A
4-26	A
5-1	A
5-2	A
5-3	A
5-4	A
5-5	A
5-6	A
5-7	A
5-8	A
A-1	A
A-2	A
A-3	A
A-4	A
A-5	A
A-6	A
A-7	A
A-8	A
A-9	A
A-10	A
A-11	A
B-1	A
B-2	A
B-3	A
B-4	A
C-1	A
C-2	A
C-3	A
C-4	A
C-5	A
D-1	A
D-2	A
E-1	A
E-2	A
F-1	A
Index-1	A
Index-2	A
Index-3	A
Comment Sheet	A
Back Cover	-

PREFACE

This manual describes the Remote Host Facility (RHF) access method, which controls network communications between two or more application programs. The RHF access method runs on either the CONTROL DATA® Network Operating System (NOS) or the CDC® Network Operating System/Batch Environment (NOS/BE).

AUDIENCE

This manual is intended for application programmers. It assumes the user is familiar with NOS or NOS/BE operations and RHF network configurating as documented in the NOS 2 Analysis Handbook or NOS/BE Installation Handbook.

RELATED PUBLICATIONS

Additional information is provided in the following manuals.

Control Data Publication	Publication Number
NOS Version 2 Analysis Handbook	60459300
NOS Version 2 Installation Handbook	60459320
NOS Version 2 Operations Handbook	60459310
NOS Version 2 Reference Set, Volume 4 Program Interface	60459690
NOS/BE Version 1 Diagnostic Handbook	60494400
NOS/BE Version 1 Installation Handbook	60494300
NOS/BE Version 1 Operator's Guide	60493900
NOS/BE Version 1 Reference Manual	60493800
NOS/BE Version 1 System Programmer's Reference Manual, Volume 1	60494100
NOS/BE Version 1 System Programmer's Reference Manual, Volume 2	60457370
Remote Host Facility Usage	60460620
380-170 Network Access Device Hardware Reference Manual	60458500

SUBMITTING COMMENTS

The last page of this manual is a comment sheet. Please use it to give your opinion on the manual's usability, to suggest specific improvements, and to report any errors. If the comment sheet has already been used, you can mail your comments to:

Control Data Corporation
Publications and Graphics Division ARH219
4201 Lexington Avenue North
St. Paul, MN 55112

Additionally, if you have access to SOLVER, an online facility for reporting problems, you can use it to submit comments about the manual. Declare your problem type as DOC and use NS2 as the product identifier.

DISCLAIMER

The RHF access method is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

GENERAL DESCRIPTION 1

This section outlines the software and hardware involved in processing network applications. It describes the interactions among these elements in a general manner. In the context of this manual, network is a blanket term for all operations involving a Loosely Coupled Network (LCN) and Remote Host Facility (RHF).

A Control Data communication network is a shared-access data message and file transfer system that is transparent to the application programs using it. LCN consists of linked CDC® 380 Network Access Devices (NADs). Because the connections among NADs can become extremely complex, RHF acts as an interface between local application programs in a host computer at one end of the network and remote application programs in other hosts at other ends of the network.

The host computer executes CDC-written or site-written service programs called application programs that are connected to the network. An application program can communicate with other application programs connected to the network. All connections to the network processing are controlled by RHF.

RHF is both flexible and efficient so that application programs can exchange information with other application programs in a shared-access, generalized manner.

FEATURES AND SERVICES

RHF incorporates the following features:

- RHF is equally suitable for application programs written in COMPASS or high-level languages, such as FORTRAN.
- RHF imposes no data structures on an application program.
- RHF provides a simple way to handle unpredictable events, such as operator interrupts.
- RHF provides complete isolation of network communications from the operating system.

Basic services provided by RHF include the following:

- RHF establishes message paths (logical connections) between two application programs provided both programs have the correct network access permissions.
- RHF breaks logical connections when asked to by an application program or when network conditions make it necessary; for example, when a network shutdown occurs.
- After logical connections have been established, RHF passes incoming messages to the application program, and accepts and forwards outgoing messages from the application program.
- RHF can simultaneously connect a local application program to many remote application programs that may reside in different hosts.

- RHF can simultaneously connect many local application programs to one remote application program.

- RHF queues incoming messages until the application program requests them. This allows the local application program to service its connections with remote application programs in any order desired.

An installation option is available to log message traffic for application program debugging.

LOGICAL CONNECTIONS

In the network, when RHF passes data between application programs, a message path or logical connection exists between the two programs. Conceptually, this is equivalent to the connection between two telephones used in a conversation.

An application program, however, can be connected simultaneously to many other application programs. It is connected to each program by a separate and distinct logical connection. The application program identifies a particular program by specifying the logical connection between itself and the other program. This is possible because a one-to-one association exists between the connection and the program. From the application programmer's point of view, it is convenient to talk of connection x (literally, message path x) when it would be more precise to say the application program at the other end of connection x. Figure 1-1 shows typical logical connections in the network.

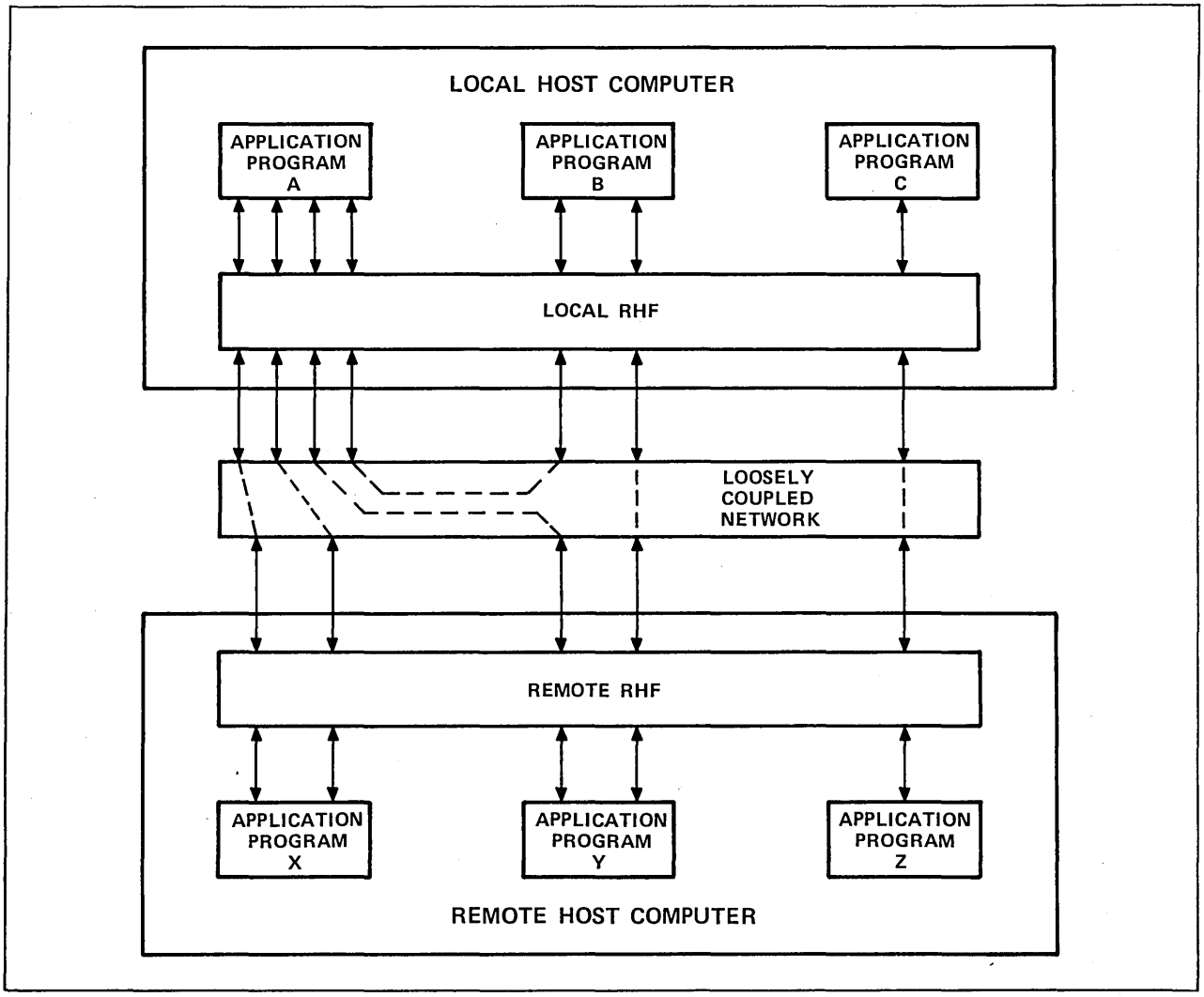


Figure 1-1. Typical Logical Connections in the Network

REMOTE HOST FACILITY COMPONENTS

Figure 1-2 shows the components of RHF used by the software discussed in this manual. The area enclosed by the dotted lines makes up the RHF access method, which includes:

- The RHF system control point program (SCP) - This central processor program executes at a system control point. It provides all connection management and is the intermediary between a facility interface program and the network driver.

- The network driver program (NDR) - This program executes in one or more peripheral processor units. NDR communicates with NADs using a host computer data channel, and is the interface between the RHF system control point program and LCN.

- The network load/dump driver program (NLD) - This program executes in a peripheral processor unit. NLD is the program used to load or recover a NAD.

- The facility interface program (FIP) - FIP consists of routines and buffers that are loaded into each application program's field length. FIP is the interface between the application program and the RHF system control point program. This manual is primarily concerned with the use of FIP.

Application programs use calls to FIP procedures to receive and transmit data. Data being received or transmitted is buffered in the NAD.

Inbound data from LCN is saved, unmodified, by the NAD. Data is removed from the NAD when application program FIP statements request input on a logical connection. The data can be translated by FIP or the NAD if the application program has requested it. FIP transfers the data to a local file or to a data buffer established and maintained by the application program.

Outbound data is handled in the reverse manner. The application program calls a FIP procedure to send data on a logical connection. The data is transferred from a local file or the program's field length to the NAD. Code conversion and translation, if necessary, are done by FIP or the NAD.

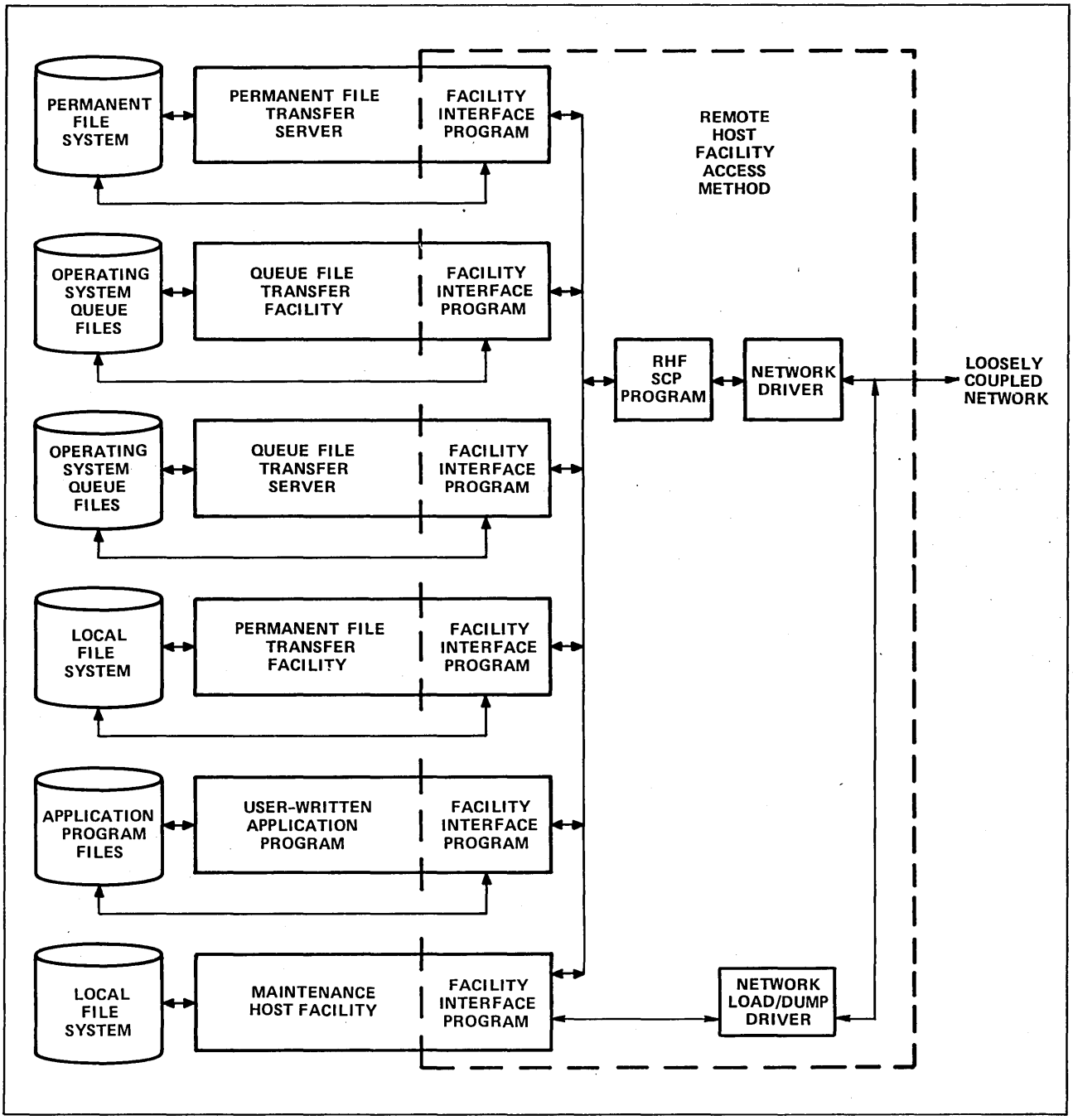


Figure 1-2. Remote Host Facility Components

APPLICATION PROGRAM REQUIREMENTS

RHF requires an application program to reside at a separate operating system control point. This program must contain some of the calls to FIP listed in appendix D and described in section 4. An application program begins accessing the network by calling NETON. It transmits data through the network by calling NETPUT and it receives data through the network by calling NETGET or NETGETL. Files can be transferred in either direction through the network by calling NETXFR or NETUXFR. The application program ends network access by calling NETOFF. An application program also:

- Must contain buffers for transmitting or receiving data. One buffer can be used for all logical connections, one buffer can be used for each logical connection, or many buffers can be used for each logical connection.
- Sends instructions to RHF and receives operational information from RHF through supervisory messages, as described in section 3. The application program must contain procedures to formulate or process these messages.
- Can contain procedures that optimize its use of the central processor. The FIP routine NETWAIT makes the program available for rollout or swapout when the program has no data to process.
- Can cause trace dumps of its network traffic by calling NETDBG.
- Must contain a call to NETOFF for proper termination of its access to the network.

OPERATION

The following topics are important when describing the operation of an application program in terms of other operating system features:

- System control point feature usage.
- Network access by privileged application programs.
- Effects of network failure on an application program.

SYSTEM CONTROL POINT FEATURE

The system control point feature of the operating system permits a program at a control point to exchange data with a program at another control point. Such programs are called system control point jobs when they are formally defined as subsystems of the operating system. They are called user control point jobs when they exchange data with a system control point job. System control point jobs (subsystems) can make privileged requests to the operating system and usually execute with an extremely high priority. System control point jobs, such as the RHF system control point program, usually reside in the operating system library.

Application programs accessing the network execute as user control point jobs using the system control point feature. The code implementing this feature is embedded in FIP and is therefore transparent to the application program. Certain aspects of system control point jobs and user control point jobs do affect application program operation, however.

Under NOS, an application program job cannot successfully execute unless the CUCP bit is set in the access word associated with its user name or it has an SSJ= special entry point. If this access word bit is not set and the program attempts to access the network, the program is aborted along with the dayfile messages, INCORRECT USER ACCESS and SYSTEM ABORT, and no error exit processing is made. Access word bits are set through the MODVAL utility described in the NOS 2 Administration Handbook.

The primary reason for requiring that an application program be privileged is to help ensure network security. When an application program calls NETON to access RHF, the NETON statement aname parameter must specify an RHF application program that is defined in the local RHF configuration file (refer to the NOS 2 Analysis Handbook or NOS/BE Installation Handbook for a definition of the RHF configuration file). RHF uses the configuration file to determine which privileges an application program must have before it can access RHF. For each application program entry in the RHF configuration file there is a set of flags that determines the privileges for that program. Table 1-1 gives the correlation between application program flags in the RHF configuration file and the privileges that RHF requires for NOS and NOS/BE.

Table 1-1. Correlation Between Application Program Flags and Privileges Required by RHF

<table>
<tr><th rowspan="2">Application Program Flag</th><th colspan="2">RHF Requirements</th></tr>
<tr><th>NOS</th><th>NOS/BE</th></tr>
<tr><td>Auto-Start Application</td><td rowspan="2">Job must be started by RHF. (Job has system origin privileges; that is, system origin or user has CSOJ validation and system is in DEBUG mode.)</td><td rowspan="2">Job must be started by RHF. (Source identificaton is $RH.)</td></tr>
<tr><td>Application Server</td></tr>
<tr><td>System Origin</td><td>Job has SSJ= special entry point.</td><td>Job is loaded from system library.</td></tr>
</table>

If none of the application program flags is set in a requested application program entry, RHF does not require any privileges. However, an individual application program may have its own scheme for privilege checking.

The current implementation of RHF requires that an auto-start application program and an application server program be installed in the system library. Each should be installed either as a program with an entry point name of aname or as a procedure file with a procedure name of aname. As used here, aname is the application program name in the RHF configuration file that defines a NETON statement aname parameter for an RHF application program.

If an RHF application program requires only system origin privileges, the application program must still be installed in the system library. In this case, NOS requires an SSJ= special entry point, but does not require a specific entry point name or procedure name.

RHF FAILURE

The present release of RHF makes no provision for data recovery if RHF failure occurs. If RHF fails, all application programs that are not system control point jobs are aborted under NOS, or rerun under NOS/BE (if possible), and a message is issued to the dayfile of each job. An aborted application program can reprieve itself under certain conditions without being reloaded. A reprieved application program must issue a NETOFF call before it can issue a new NETON call. A new NETON call can be successfully completed as soon as RHF is restarted.

BLOCKS AND MESSAGES 2

This section describes the interface between an application program and the Remote Host Facility (RHF). Topics discussed include:

- Data block formats of data transferred between application programs using RHF.
- Control messages exchanged between an application program and RHF and how these control messages are used.
- The method of exchanging these data blocks and control messages using the facility interface program (FIP).

The data blocks and control messages described in this section and used between an application program and RHF have the following advantages:

- They are simpler and more concise than would be required if the application interfaced directly with the network hardware.
- The application program interface can be independent of the network configuration.
- The integrity of the operating system, remote RHF software, and local RHF tables and software is protected because application programs do not directly access the operating system, RHF tables, or network hardware.
- This interface is compatible with the NAM AIP (network access method application interface program) interface.

MESSAGE TYPES

A message is a logical unit of information, comprising one or more physical units called blocks. A message can be a service request to RHF or data being sent to another application program.

There are two kinds of messages, data and supervisory. Data messages convey information to other application programs and have no significance to RHF. Supervisory messages convey information of significance only to RHF. Supervisory messages are used by an application program to control data messages between that application program and its logical connections. Data messages can consist of more than one block. Supervisory messages consist of only one block.

BLOCKS

All messages are sets of blocks. The block is the basic unit of information exchange for an application program. Block types are:

- Data message blocks that terminate messages. These can include physically empty blocks when such blocks convey logical information.

- Data message blocks that contain only parts of messages.
- Blocks constituting supervisory messages.

All blocks are exchanged between the application program and RHF on logical connections using two kinds of input and output buffers:

- The block header area.
- The block text area.

The block header area contains a 60-bit word describing the contents of the text area. This block header word accompanies the block during the exchange between the application program and RHF. For output blocks, the application program creates the block header and RHF interprets it. For input blocks, RHF creates the block header and the application program interprets it.

The block text area is addressed separately from the block header area and need not be contiguous to it. The text area contains the single block described by the header word in the header area. The text area can be of varying length to accommodate the various possible block lengths. The text area has a maximum length expressed as a whole number of central memory words. Blocks can occupy text areas from 0 to 409 central memory words long.

Block length is distinct from text area length. The length of a block depends on the type and use of the block. Null blocks have no length. Other block types have lengths expressed in characters. Blocks are always a whole number of characters long, but do not have to be a whole number of central memory words long. Not all words in the text area used for a given block need to be filled with meaningful information.

Supervisory message blocks are from 1 to 63 characters long. Data message blocks have lengths of zero up to the maximum number of characters that can fit in the maximum text area of 409 words, or 2043 characters, whichever occurs first. Messages containing more characters than this limit are divided into several blocks; each of these blocks meets the size restriction and is transmitted separately.

CONNECTION IDENTIFIERS

Two parameters identify and control the routing of messages, the application connection number, and the application list number. These parameters are used in the FIP calls that fetch incoming blocks and in the block header words of outgoing blocks.

APPLICATION CONNECTION NUMBER

The application connection number is a 12-bit integer used to address a particular logical connection. Connection numbers are assigned by RHF serially within each application program. Numbers that become available because of disconnections are reassigned to subsequent connections. A connection number of 0 indicates the control connection along which supervisory messages are sent and received (refer to section 3, Supervisory Messages). The connection number can be used as an index into a control structure (for example, the number of a connection could be the ordinal of a corresponding application table), or used in any other manner the application programmer chooses.

APPLICATION LIST NUMBER

RHF permits an application program to group connections with similar processing requirements into numbered lists. This is an efficiency consideration, relieving the application of specifing individual connections each time an input block is requested. Instead, when a request is made for a block from a connection list, connections with empty input queues are automatically skipped and a block from the first nonempty queue is returned. A single null block is returned when none of the connections on the list have any input queued.

The parameter used for this kind of processing is called the application list number. The application list number is an integer specified by the application program when it accepts a connection. Message input queues of all connections that have been assigned the same application list number are linked together by RHF. An application program can request blocks from these linked queues in rotation (without specifying individual connections) by including the appropriate application list number in a NETGETL statement (refer to section 4, Facility Interface Program).

An application list number identifies a connection list. A connection list can be viewed as a table of connection numbers. These connection numbers are entered in the table when the connections are assigned to the list by the application program. When the list is scanned for input from a connection, the scanning begins with the connection immediately following the connection from which input was previously transferred. The first connection on the list is examined after the last connection on the list. Scanning ends when a logical connection with queued input is found. If no logical connection has queued input, scanning ends with the connection preceding the one at which scanning started.

The application program explicitly assigns an application list number to each logical connection when the connection is established. Because the logical connection corresponding to application connection number 0 already exists when the application is connected to the network, application connection number 0 is automatically assigned to application list number 0 without program intervention.

The application program does not have to maintain any tables associating application connection numbers and application list numbers, and is not required to use list processing at all. RHF makes all necessary associations.

DATA MESSAGE BLOCKS

Blocks always contain characters. These characters can be of several lengths and can be packed within bytes of several sizes. Each permitted combination of character length and byte size is called an application character type. There are four application character types:

- One 60-bit character per 60-bit byte.
- One 8-bit character per 8-bit byte.
- One 8-bit character per 12-bit byte.
- One 6-bit character per 6-bit byte.

For an application program output block, RHF:

- Performs no mapping or character conversion on 60-bit characters.
- Performs no mapping or character conversion on 8-bit characters in 8-bit bytes.
- Performs no character conversion on 12-bit byte characters, but maps the 12-bit character to an 8-bit byte by discarding the leftmost 4 bits of the 12-bit byte.
- Maps 6-bit characters to 8-bit characters by translating the former as 6-bit display code and substituting the corresponding hexadecimal code from the ASCII 128-character set.

For an application program input block, RHF:

- Performs no mapping or character conversion on 60-bit characters.
- Performs no mapping or character conversion on 8-bit characters in 8-bit bytes.
- Performs character mapping but no conversion by right-justifying 8-bit characters in 12-bit bytes with zero fill.
- Maps and converts 8-bit characters to 6-bit characters by translating all ASCII control characters to 6-bit display coded blanks, and translating all 7-bit ASCII codes between 60_{16} and $7F_{16}$ to the 6-bit display code equivalents of the 7-bit ASCII codes between 40_{16} and $5F_{16}$. All other 7-bit ASCII codes are translated to the 6-bit display codes equivalent to the CDC subset of the ASCII 128-character set.

Conversion and mapping between 6-bit and 8-bit characters is done in the NAD and involves a time-consuming character-by-character replacement of the block's data. For efficiency, 8-bit byte characters are recommended for blocks exchanged with other application programs.

The application character type of an input block is determined by the character type associated with the logical connection. This association is first made by RHF when the connection is established but can be changed as necessary while the connection exists. The application character type of a specific input block is always indicated by a field in its associated block header word.

The application character type of an output block is determined solely by a field in its associated block header word. Input and output blocks transmitted over the same logical connection can, therefore, have different application character types.

The character conversion to or from 6-bit characters and the mapping between 8-bit characters and 8/12-bit characters does not take zero-byte line terminators into account. For example, a 12-bit zero-byte would be converted to two 7-bit ASCII code colons. Thus, these character types are primarily limited to use in exchanges where the string length is known or specified in the exchange.

Application programs typically use the file transfer capability for transferring files or large amounts of data. The file transfer capability processes zero bytes for both 6-bit characters and 8/12-bit characters. Refer to File Transfer Statements in section 4 for additional information.

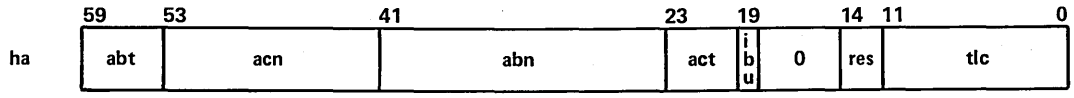
RHF provides a mechanism for the application program to determine when data blocks are queued on a logical connection. When a call to a FIP statement is completed, a status word at a location defined by the application program is updated to indicate whether any data blocks are queued. The application program can test the status word periodically, instead of attempting to fetch null blocks from all application connection numbers.

DATA BLOCK HEADER FORMATS

The format of the block header word associated with a data block depends on whether the application program is receiving an input data block or sending an output data block.

Input Data Blocks

The application block header format for input data blocks is:



Field	Description
ha	Symbolic header area address specified as the location receiving the application block header in a call to NETGET or NETGETL.
abt	Application block type of the associated data block. This field can have one of the following values:

Value	Description
0	Indicates a null block. No block is queued from the logical connection polled.
1	Indicates the associated block is one of several blocks comprising a single message, but is not the last such block.
2	Indicates the associated block is either the last block in a series or that the message being sent requires only one block.

Values of 3 through 63 are not valid for data blocks on input.

acn	Application connection number of the logical connection from which the associated data block was received. This field can have the values $1 \leq \text{minacn} \leq \text{acn} \leq \text{maxacn} \leq 4095$, where the values minacn and maxacn are parameters in the NETON statement.
abn	Application block number assigned to the associated data block. If the application block header is associated with a data block from another application program, this field contains the 18-bit integer used by that program to identify the block when the block was sent.

Field	Description

act — Application character type used to encode the accompanying data block. This field can have one of the following values:

Value	Description
1	60-bit transparent characters, packed one character per central memory word.
2	8-bit characters, packed 7.5 characters per central memory word.
3	8-bit characters, right-justified in 12-bit bytes with zero fill, packed five characters per central memory word.
4	6-bit display code characters, packed 10 characters per central memory word.

Values of 0 and 5 through 15 are reserved for expansion and currently are not valid. The value contained in the act field is the value assigned to the connection by the application program for input in the connection-accepted supervisory message.

ibu — Input-block-undeliverable bit. Normally when this bit is set (has a value of 1), the data block associated with this block header has not been delivered to the application program because the block is larger than the maximum text length (tlmax parameter) declared by the application program in its NETGET call. The block remains queued by FIP until a call is issued that specifies an adequate text area length. The block header for such a block contains the actual length of the queued block in its tlc field, given in character units of the type specified in the act field.

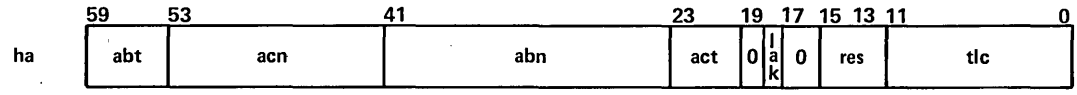
res — Reserved for CDC.

tlc — Text length of the associated data block, given in character units of the type specified in the act field. The equivalent length in central memory words that the block can occupy can be computed as follows:

act	Central Memory Words
1	tlc is the number of central memory words the block can occupy.
2	Divide the value of tlc by 7.5 and round upward to an integer.
3	Divide the value of tlc by 5 and round upward to an integer.
4	Divide the value of tlc by 10 and round upward to an integer.

Output Data Blocks

The application block header format for output data blocks is:



Field	Description
ha	Symbolic header area address specified as the application block header's location in a call to NETPUT.
abt	Application block type of the accompanying data block. This field can have one of the following values:

Value	Description
1	Indicates the accompanying block is one of several blocks comprising a single message, but is not the last such block.
2	Indicates the accompanying block is either the last block in a series or that the message requires only one block.
6	Indicates the accompanying block is one of several blocks comprising a single qualified data message, but is not the last such block. RHF converts the block type value of 6 to a value of 1 before the block is sent.
7	Indicates the accompanying block is either the last block in a series or that the qualified data message requires only one block. RHF converts the block type value of 7 to a value of 2 before the block is sent.

Values of 0, 3 through 5, and 8 through 63 are not valid for data blocks on output.

acn — Application connection number of the logical connection to which the accompanying data block should be sent. This field can have the values $1 \leq minacn \leq acn \leq maxacn \leq 4095$, where the values minacn and maxacn are parameters in the NETON statement.

Field	Description
abn	Application block number assigned to the data block being sent. This field is an 18-bit integer that identifies the block when RHF returns certain supervisory messages. For maximum flexibility, definition of the block number is left to the programmer; it can be: • A sequencing number. • The block's central memory address. • The block's mass storage address (physical record unit). • An index value for a block control array or table. • An external label.
act	Application character type used to encode the accompanying data block. This field can have one of the following values: 1 — 60-bit transparent characters, packed one character per central memory word. 2 — 8-bit characters, packed 7.5 characters per central memory word. 3 — 8-bit characters, right-justified in 12-bit bytes, packed five characters per central memory word. 4 — 6-bit display code characters, packed 10 characters per central memory word. Values of 0 and 5 through 15 are reserved and currently are not valid.
lak	Local acknowledge flag. This field can have one of the following values: 0 — Indicates the accompanying block is an acknowledged data block. No bits are set in the data block clarifier field of the network block header (refer to appendix E). 1 — Indicates the accompanying block is an unacknowledged data block. The PRU block flag and end of record flag are set in the data block clarifier field of the network block header (refer to appendix E).
res	Reserved for CDC.

Field	Description
tlc	Text length of the associated data block, given in character units of the type specified in the act field. The equivalent length in central memory words that the block can occupy can be computed as follows:

act	Central Memory Words
1	tlc is the number of central memory words the block can occupy.
2	Divide the value of tlc by 7.5 and round upward to an integer.
3	Divide the value of tlc by 5 and round upward to an integer.
4	Divide the value of tlc by 10 and round upward to an integer.

SUPERVISORY MESSAGE BLOCKS

Supervisory message blocks consist of from 1 to 63 60-bit words. The fields within these words convey information and instructions between RHF and the application program's text area buffer. Supervisory messages are sent and received through the same FIP routines used for data message blocks. Supervisory messages have associated block header words that convey information between RHF and the header area buffer of the application program.

All supervisory messages have the same general format (refer to Supervisory Message Formats in section 3). The text area of a specific message contains a fixed combination of four fields and can include additional fields to clarify processing of the message. Together, the four common fields define one supervisory message. The individual messages supported by RHF are described in section 3.

The first of the fields common to all supervisory messages is the primary function code. The primary function code is used to group supervisory messages into related functions and determine their routing within the network. The following are the symbols used to identify primary function codes (pfc) and their meanings:

pfc	Meaning
CM	Convert mode control.
CON	Connection control.
CTRL	Application control.
DC	Connection characteristic definition.
ERR	Error reporting.
FC	Connection data flow control.
LST	Connection list management.
SHUT	Network shutdown.

The second common field is the error bit. When the error bit is set to 1, it indicates an error or abnormal response; 0 indicates a normal response.

The third common field is the response bit. When the response bit is set to 1, it indicates a normal response; 0 indicates an error or abnormal response.

The fourth common field is the secondary function code. The precise function of a message within a primary function code grouping is indicated by its secondary function code. The following are the symbols used to identify secondary function codes (sfc) and their meanings:

sfc	Meaning
ACK	Acknowledgment.
ACRQ	Application-to-application connection request.
BRK	Break.
CB	Connection broken.
CICT	Change input character type.
END	End of connection.
EXIT	Exit convert mode.
INFO	Information exchange with RHF.
INIT	Initialization.
INSD	Internal shutdown.
LGL	Logical problem.
NAK	No acknowledgment.
OFF	Turn connection list processing off.
ON	Turn connection list processing on.
REQ	Request for logical connection.
RST	Reset connection.
SEL	Select convert mode.
SWH	Switch connection between lists.

When more than one possible action can be taken for a function, the error bit and response bit fields are used to indicate the action chosen. The error bit is usually set to indicate the message recipient's refusal to complete the function, while the response bit is set to indicate the recipient's normal completion of the function.

Supervisory messages are sent or received separately from the stream of data message blocks between an application program and a logical connection. Whether these messages are used alone or as part of a sequence, their receipt or the need to send them cannot be predicted from the generalized logic required for data block processing. Such messages are said to be asynchronous to the data block stream.

All supervisory messages are sent or received on a special logical connection that has a preassigned application connection number of 0. This application connection number is preassigned by RHF to connection list 0.

All supervisory messages are sent to or received from software resident in the host computer, although they may be reformatted by this software for communication with software outside of the host. For ease of communication with this resident software, supervisory messages use an application character type of 1.

Supervisory messages are processed with the same FIP routines used by an application program to process data message blocks on logical connections other than application connection number 0. Supervisory messages are queued on their special connection until fetched by the application program. The application program fetches them one message at a time until the connection queue is empty and a null block with an application block type of 0 is returned. Because the application program must test for a zero block type to determine when no additional supervisory messages are available, supervisory messages are assigned the nonzero application block type of 3.

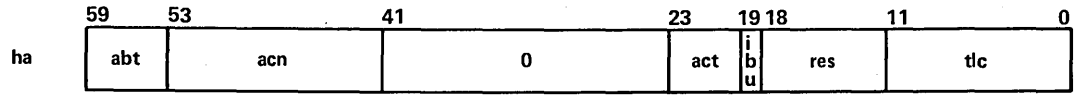
RHF provides a mechanism for the application program to determine when supervisory messages are queued on application connection number 0. When a call to a FIP routine is completed, a supervisory status word at a location defined by the application program is updated to indicate whether any supervisory messages are queued. As long as the application program continues to make calls to FIP routines it can test the supervisory status word periodically, instead of attempting to fetch null blocks from application connection number 0.

SUPERVISORY BLOCK HEADER FORMATS

The format of the block header word associated with a supervisory message depends on whether it is an input supervisory message or an output supervisory message.

Input Supervisory Messages

The application block header format for input supervisory messages is:



Field	Description
ha	Symbolic header area address specified as the location receiving the application block header in a call to NETGET or NETGEL.
abt	Application block type of the associated message block. This field can have one of the following values:

Value	Description
0	Indicates a null block. No supervisory message is queued from the logical connection polled.
3	Indicates the accompanying block is a supervisory message block.

Values of 1, 2, and 4 through 63 are not valid for supervisory messages on input.

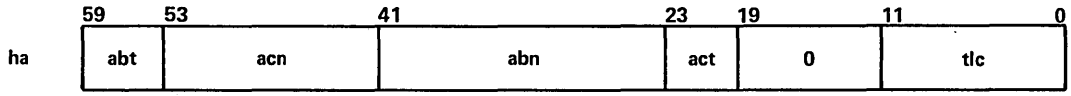
acn — Application connection number of the logical connection from which the message block was received. This field can have one of the following values:

Value	Description
0	Supervisory messages from the host portion of RHF.
Nonzero	Reserved for CDC.

Field	Description
act	Application character type used to encode the accompanying message block. A value of 1 specifies a supervisory message in 60-bit transparent characters, packed one character per central memory word.
ibu	Input-block-undeliverable bit. Normally when this bit is set (has a value of 1), the message block associated with this block header has not been delivered to the application program because the block is larger than the maximum text length (tlmax parameter) declared by the application program in its NETGET or NETGETL call. The message block remains queued by FIP until a call is issued that specifies an adequate text length. A block header containing a set ibu bit contains the actual length of the queued block in its tlc field, given in character units of the type specified in the act field.
res	Reserved for CDC.
tlc	Text length of the associated message block, given in character units of the type specified in the act field. tlc is the number of cental memory words the message block can occupy.

Output Supervisory Messages

The application block header format for output supervisory messages is:



Field	Description
ha	Symbolic header area address specified as the application block header's location in a call to NETPUT.
abt	Application block type; abt is 3 for all supervisory messages.
acn	Application connection number of the logical connection to which the message block should be sent. This field can have one of the following values:

Value	Description
0	Supervisory messages addressed to the host portion of RHF.
Nonzero	Reserved for CDC.

Field	Description
abn	Application block number assigned to the message block being sent. This field is an 18-bit integer that identifies the block. For maximum flexibility, definition of the block number is left to the programmer; it can be: • A sequencing number. • The block's central memory address. • The block's mass storage address (physical record unit). • An index value for a block control array or table. • An external label.
act	Application character type used to encode the accompanying message block. A value of 1 specifies a supervisory message in 60-bit transparent characters, packed one character per central memory word.
tlc	Text length of the associated message block, given in character units of the type specified in the act field. tlc is the number of central memory words the message block can occupy.

UNACKNOWLEDGED DATA MESSAGE BLOCKS

To more efficiently transmit large amounts of data between two application programs, RHF provides special data message blocks known as unacknowledged data blocks or physical record unit (PRU) data blocks. These blocks differ from acknowledged (non-PRU) data blocks in the following ways:

- No receiver-to-sender acknowledgment of the data block delivery occurs.
- The application block number is used to verify the correct sequencing of blocks. The application block number is incremented by one for each block sent; starting with zero.
- A data block clarifier field in the network block header can be used to define higher level file structures. Refer to appendix E for additional information about the data block clarifier field.
- Data blocks contain data in one of two basic formats: either character data or binary data.

Unacknowledged data blocks are described further under FET I/O Interface (NETUXFR) in section 4.

SUPERVISORY MESSAGES 3

This section describes supervisory messages that are valid for application program communication with RHF. These messages are described in the context of their use. Program processing of all supervisory messages must include a block header word, as described in section 2.

A supervisory message is identified by the contents of its primary function code field, error bit, response bit, and secondary function code field. This structure allows a supervisory message to be described by the following format:

pfc/sfc/sm

Field	Description
pfc	The reserved symbolic mnemonic for the contents of the primary function code field.
sfc	The reserved symbolic mnemonic for the contents of the secondary function code field.
sm	A letter indicating the combined settings of the error bit and response bit. The letter, bit settings, and meaning are:

Letter	Error Bit	Response Bit	Meaning
R	0	0	Initial request.
N	0	1	Normal response.
A	1	0	Abnormal response.

Although many combinations of valid field values are possible, only certain combinations are permitted. The permissable combinations are listed alphabetically in table 3-1. The table also shows the message initiator (RHF or application), its octal code (bits 59 through 42 of the first text area word), and its meaning.

Table 3-1. Supervisory Messages (Sheet 1 of 2)

Supervisory Message	Initiator	Octal Code	Meaning
CM/EXIT/N	RHF	606404	Convert mode terminated.
CM/EXIT/R	Application	606004	Exit convert mode.
CM/SEL/A	RHF	607000	Request to select convert mode rejected.
CM/SEL/N	RHF	606400	Request to select convert mode accepted.
CM/SEL/R	Application	606000	Request to select convert mode.
CON/ACRQ/A	RHF	307010	Application-to-application connection reject.
CON/ACRQ/R	Application	306010	Application-to-application connection request.
CON/CB/R	RHF	306024	Connection broken.
CON/END/N	RHF	306430	Connection processing ended.
CON/END/R	Application	306030	End connection processing.
CON/REQ/A	Application	307000	Connection rejected.
CON/REQ/N	Application	306400	Connection accepted.
CON/REQ/R	RHF	306000	Connection requested.
CTRL/INFO/A	RHF	603050	Request for RHF control information rejected.
CTRL/INFO/N	RHF	602450	Request for RHF control information accepted.
CTRL/INFO/R	Application	602050	Request for RHF control information.
DC/CICT/R	Application	604000	Change application character type of connection input.
ERR/LGL/R	RHF	410004	Logical error.
FC/ACK/R	Remote RHF	406010	Output block delivered.
FC/BRK/R	Application	406000	Report application-to-application error to remote RHF.

Table 3-1. Supervisory Messages (Sheet 2 of 2)

Supervisory Message	Initiator	Octal Code	Meaning
FC/BRK/R	RHF	406000	Application-to-application error reported by remote application.
FC/RST/R	Application	406004	Request to reset connection.
FC/RST/R	RHF	406004	Request from remote application to reset connection.
FC/INIT/N	Application	406434	Application accepts connection initialization.
FC/INIT/R	RHF	406034	RHF requests connection initialization.
FC/NAK/R	RHF	406014	Output block not delivered.
LST/OFF/R	Application	600000	Turn off list processing for connection.
LST/ON/R	Application	600004	Turn on list processing for connection.
LST/SWH/R	Application	600010	Switch application list number for connection.
SHUT/INSD/R	RHF	204030	Network shutdown in progress.

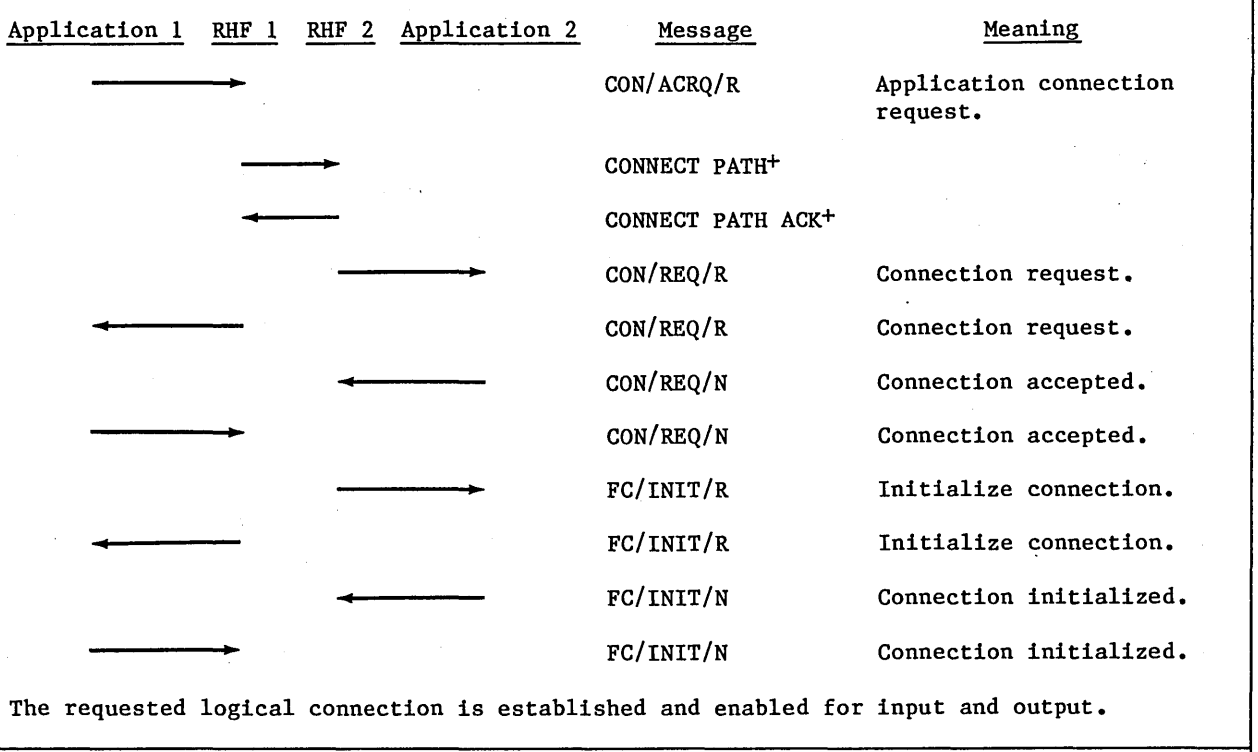
SUPERVISORY MESSAGE SEQUENCES

Supervisory messages are used in fixed sequences of one or more messages. Related messages (messages distinguished by the use of the error and response bits) are always part of multimessage sequences. Each sequence is illustrated with a figure that shows the sender and recipient of the messages in the sequence, and the direction of transmission of each message (arrows).

Application programs should be prepared to send or receive supervisory messages according to one of the fixed sequences shown on the following pages. However, some variation of these sequences may occur. For instance, an application program may receive a connection-broken (CON/CB/R) message from RHF at any point in its processing, even in the middle of another message sequence.

APPLICATION-TO-APPLICATION CONNECTION MESSAGE SEQUENCE

Figure 3-1 shows the three possible message sequences that can occur in response to an application-connection-request (CON/ACRQ/R) message. The message sequences are: 1) the requested logical connection is made, 2) the requested logical connection is not made, and 3) the requested logical connection is rejected.



+ CONNECT PATH and CONNECT PATH ACK are messages defined within a lower level protocol. They are shown in this manual only to indicate how the various messages of a sequence fit together.

Figure 3-1. Application-To-Application Connection Message Sequences (Sheet 1 of 2)

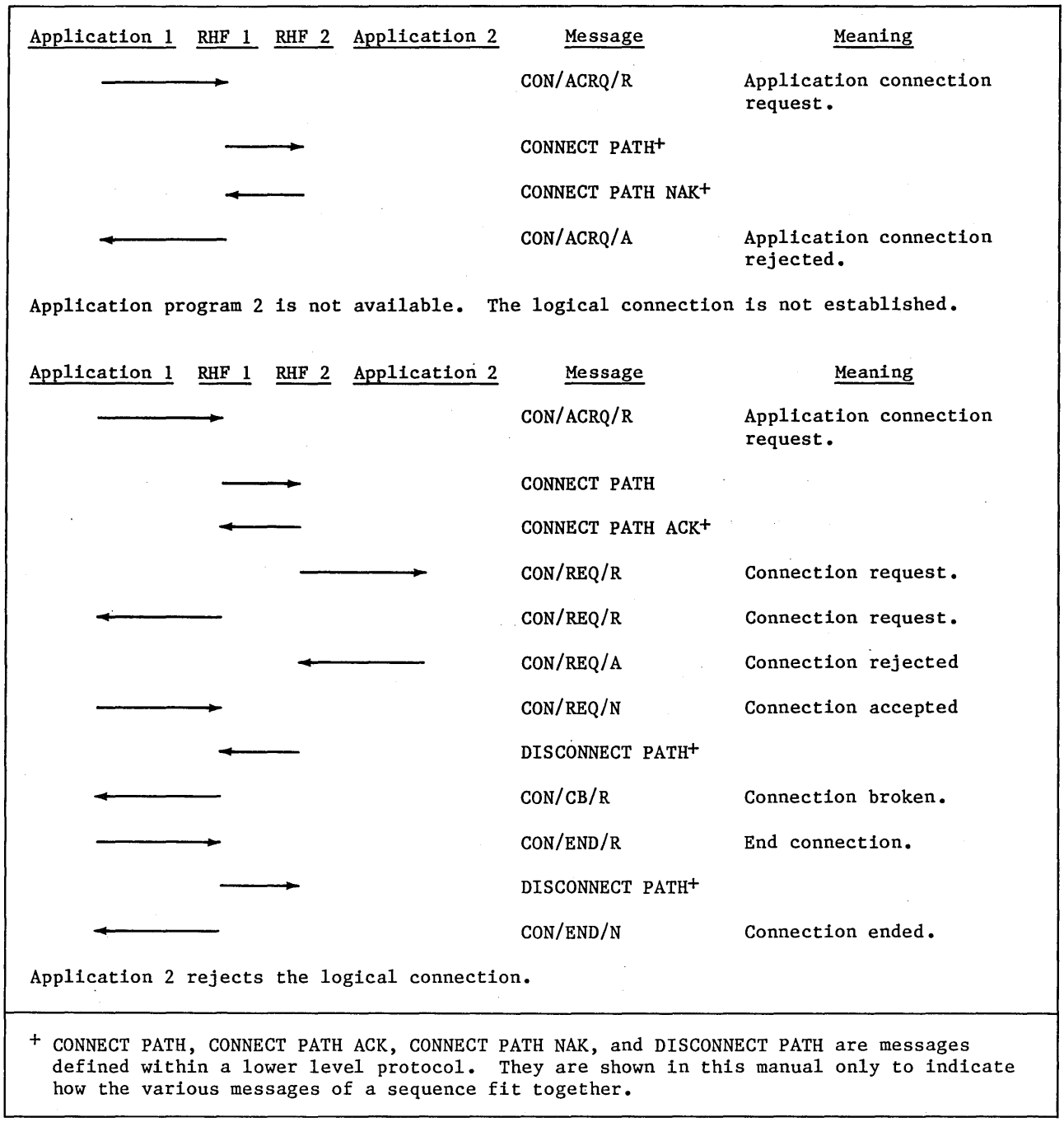


Figure 3-1. Application-To-Application Connection Message Sequences (Sheet 2 of 2)

In the connection rejection sequence, Application 2 rejects the logical connection after RHF 2 has accepted it. RHF 2 responds by sending a DISCONNECT PATH message to RHF 1. RHF 1, in turn, generates either an application-connection-rejection (CON/ACRQ/A) message or a connection-broken (CON/CB/R) message, depending on when RHF 1 receives the DISCONNECT PATH message in the connection establishment sequence. If RHF 1 receives the DISCONNECT PATH message before it sends a connection-request (CON/REQ/R) message to Application 1, RHF 1 generates an application-connection-rejected (CON/ACRQ/A) message instead of a connection-request (CON/REQ/R) message. However, if RHF 1 receives the DISCONNECT PATH message anytime after sending a connection-request (CON/REQ/R) message to Application 1, RHF 1 generates a connection-broken (CON/CB/R) message.

CONNECTION TERMINATION MESSAGE SEQUENCE

A logical connection can be terminated any time after it is established. This termination can be voluntary or involuntary from an application program's viewpoint. The application connection number of the terminated logical connection then becomes available for use with another logical connection.

Voluntary termination of a logical connection occurs when the application program has finished exchanging blocks over the logical connection. The voluntary message sequence begins when the application program issues an end-connection (CON/END/R) message, as shown in figure 3-2.

Involuntary termination of a logical connection occurs whenever a condition arises that is not caused by an application program. The involuntary message sequence begins when RHF issues a connection-broken (CON/CB/R) message to the application program (figure 3-2). When the application program receives this message, it can still fetch any input blocks queued from the logical connection. As soon as it has fetched all outstanding blocks, the application program must issue an end-connection (CON/END/R) message. RHF responds by disconnecting the path and issuing a connection-ended (CON/END/N) message.

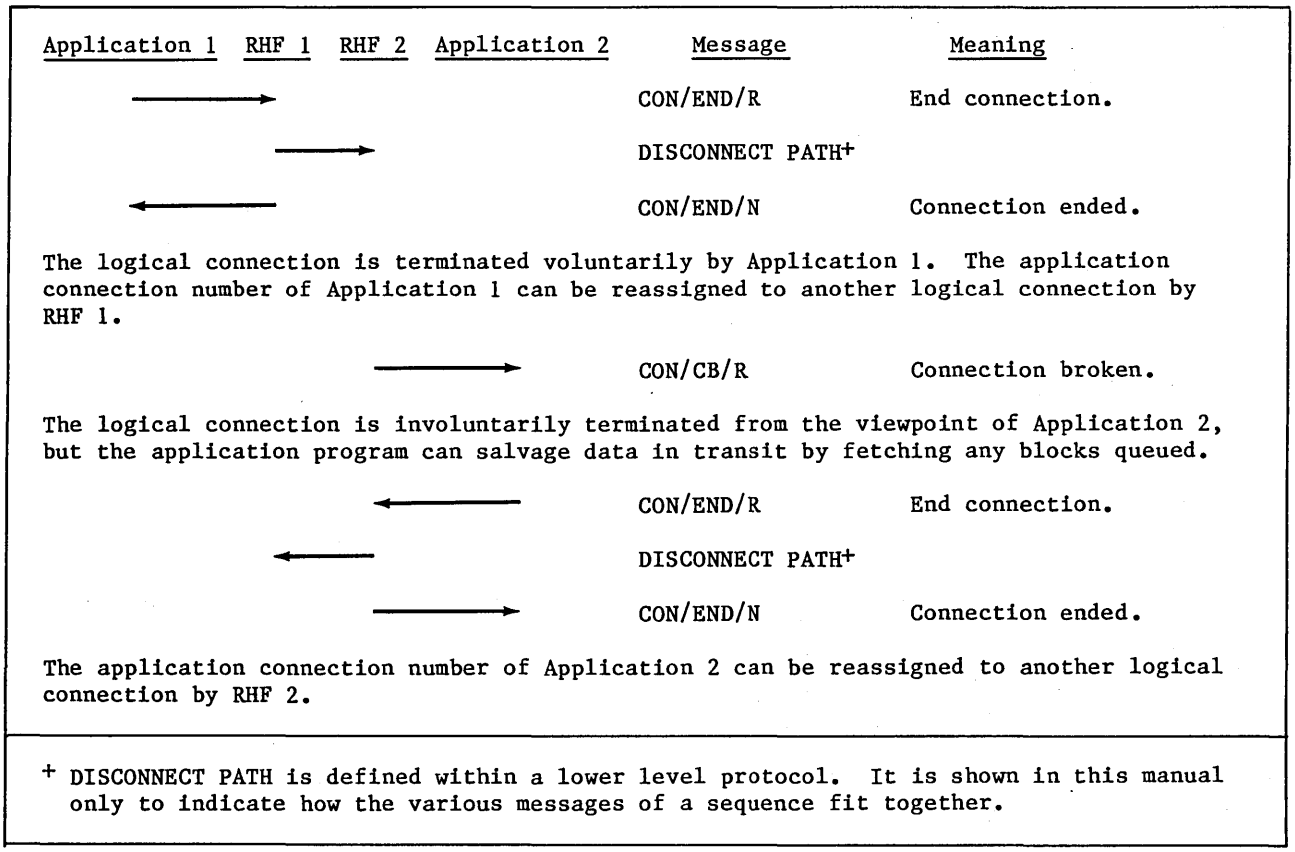


Figure 3-2. Connection Termination Message Sequences

NOTE

CON/END/R, CON/END/N is the sequence used to disconnect an established logical connection. The CON/CB/R message can be generated by the other application rejecting a connection, the other application voluntarily terminating the path, or by a network interruption. The CON/CB/R message is not generated if the two application programs simultaneously voluntarily terminate the path.

CONTROL INFORMATION MESSAGE SEQUENCE

The control-information-request (CTRL/INFO/R) message function is determined by selecting one of three function codes.

The CTRL/INFO/R message with function code 0 selected allows an application program to query RHF to determine what host it will likely be connected to when it issues an application-connection-request (CON/ACRQ/R) message. RHF responds to this function by issuing a control-information-accepted (CTRL/INFO/N) message or a control-information-rejected (CTRL/INFO/A) message, as shown in figure 3-3.

The CTRL/INFO/R message with function code 1 selected (NOS/BE only) allows an application program to set the end-of-job connection to RHF. RHF does not reply to this function. If the application program does not have the proper privileges to issue this function, it is aborted.

The CTRL/INFO/R message with function code 2 selected allows an application program to obtain a copy of the network description table being used by RHF. RHF does not reply to this function. If the application program does not have the proper privileges to issue this function, it is aborted.

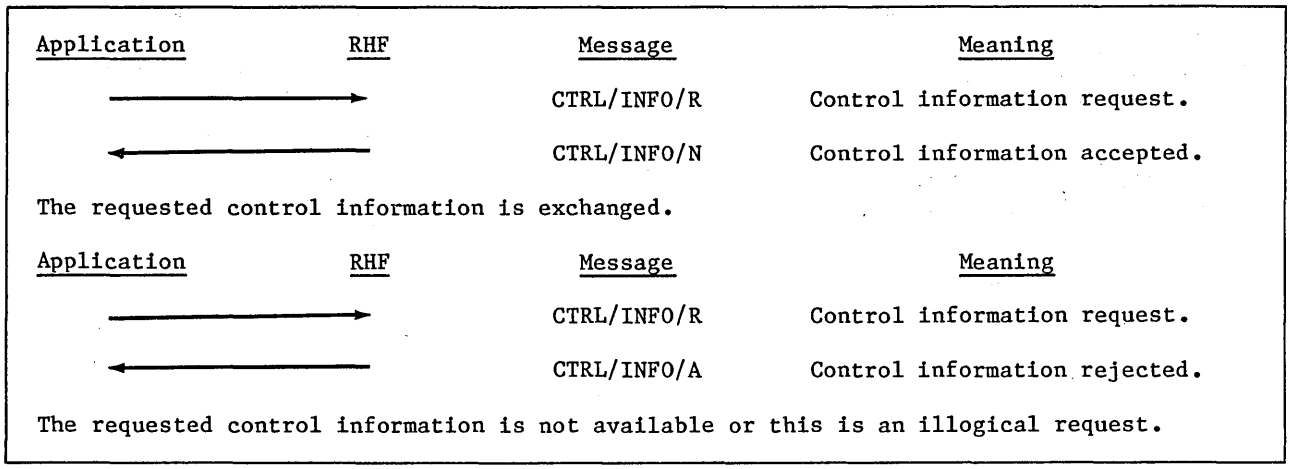


Figure 3-3. Control Information Message Sequences

CONNECTION LIST POLLING CONTROL MESSAGE SEQUENCE

Figure 3-4 shows the messages that turn connection list polling off or on and the message that changes the list number associated with a specific connection.

Application	RHF	Message	Meaning
——————→		LST/OFF/R	Turn off list processing.

When the list number associated with the affected logical connection is next polled by the application program, no blocks will be returned from the connection.

Application	RHF	Message	Meaning
——————→		LST/ON/R	Turn on list processing.

When the list number associated with the affected logical connection is next polled by the application program, blocks may be returned from the connection.

Application	RHF	Message	Meaning
——————→		LST/SWH/R	Change connection list number.

When the new list number associated with the affected logical connection is next polled by the application program, blocks may be returned from the connection.

Figure 3-4. Connection List Polling Control Message Sequences

DATA BLOCK MONITORING MESSAGE SEQUENCE

Because an application program cannot control output conditions at the other end of its network connections, data flow control is implemented through block delivery monitoring mechanisms. These mechanisms involve exchanges of supervisory messages and the application program's adherence to data block transmission conventions. Figure 3-5 shows the data block monitoring message sequences.

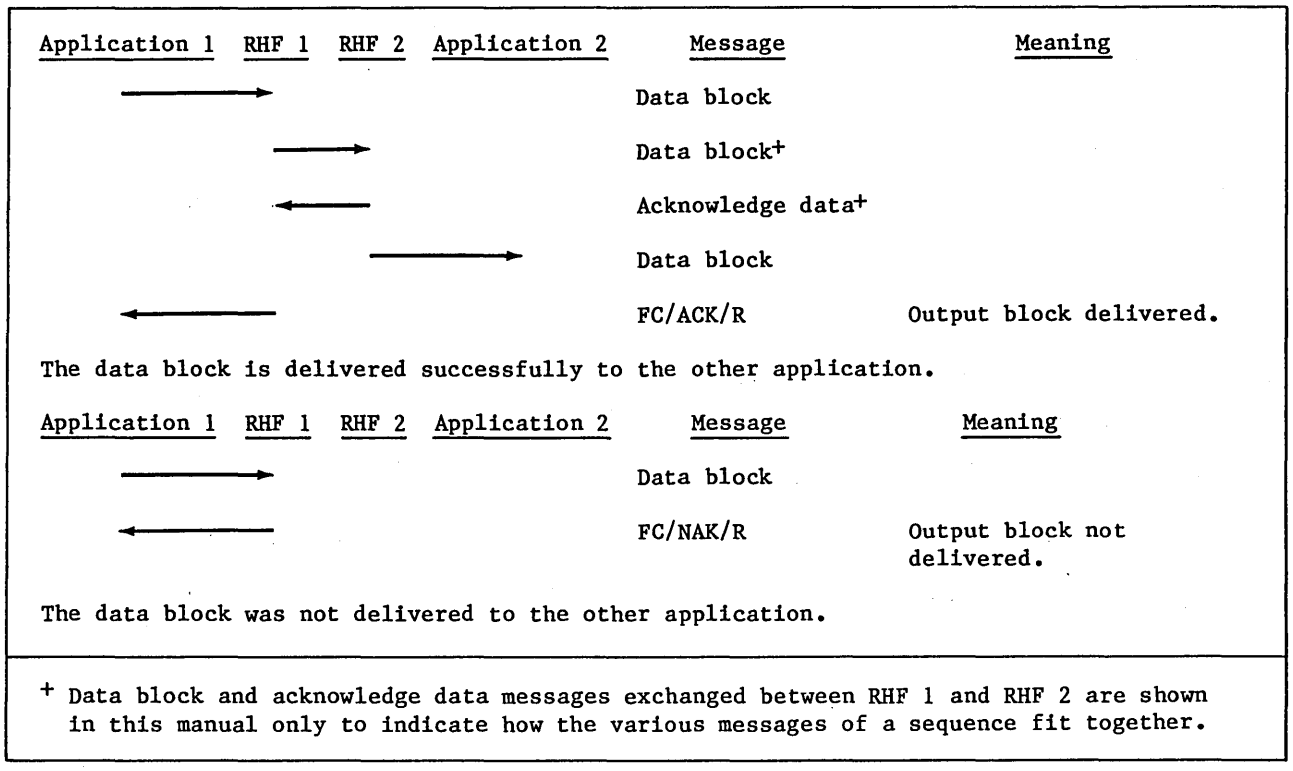


Figure 3-5. Data Block Monitoring Message Sequences

CHANGE INPUT CHARACTER TYPE MESSAGE SEQUENCE

An application program can change the data conversion on a given logical connection by changing the act field value associated with the logical connection. The program changes the act field value by issuing a change-input-character-type (DC/CICT/R) message, as shown in figure 3-6.

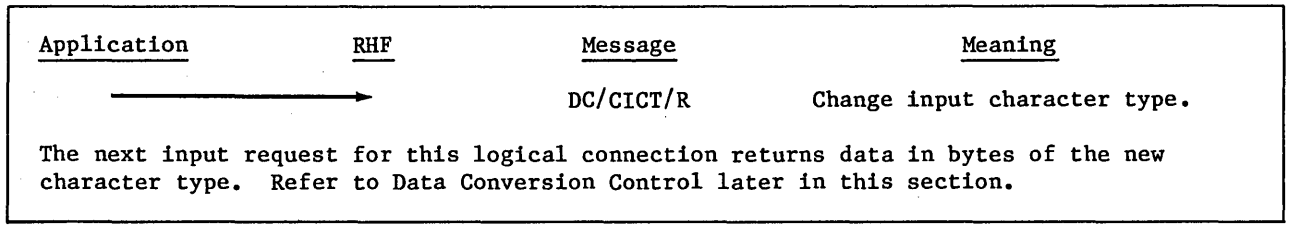


Figure 3-6. Change Input Character Type Message Sequence

HOST SHUTDOWN MESSAGE SEQUENCE

The host shutdown message sequences are shown in figure 3-7. The first sequence begins when the RHF operator requests an idle-down option. The application program receives an advisory shutdown message from RHF, shuts down its connections gracefully, and terminates RHF access. The second sequence begins when RHF is terminating involuntarily. The application program receives a mandatory shutdown message and should not attempt to terminate connections gracefully. The application program must terminate its operations immediately.

Application	RHF	Message	Meaning
←		SHUT/INSD/R	Network shutdown in progress (idle-down option set in supervisory message).

The application program fetches all queued blocks from other application programs, then ends all connections in an orderly manner prior to a network shutdown.

→		CON/END/R	End connection.
←		CON/END/N	Connection ended.

The application program disconnects from the network with a NETOFF call to FIP.

Application	RHF	Message	Meaning
←		SHUT/INSD/R	Network shutdown in progress (disable option set in supervisory message).

The application program must perform an immediate call to NETOFF to avoid being aborted by the system. The idle-down option may be followed at any time by the disable option.

Figure 3-7. Host Shutdown Message Sequences

ERROR MESSAGE SEQUENCE

Figure 3-8 shows the error message sequence. The application program does not send a response to this supervisory message.

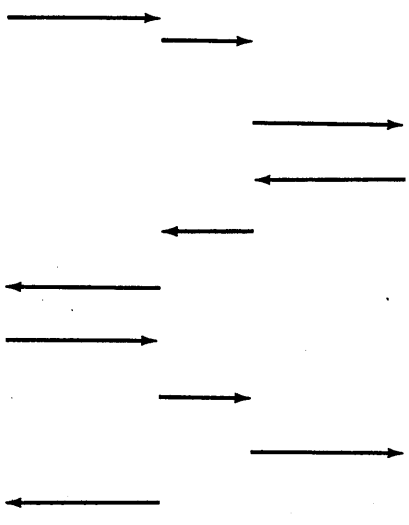
Application	RHF	Message	Meaning
←		ERR/LGL/R	Logical error.

A logical error was detected by RHF and is being reported to the application program.

Figure 3-8. Error Message Sequence

APPLICATION BREAK/RESET MESSAGE SEQUENCE

The application-break and application-reset message sequences are shown in figure 3-9. The first sequence begins when Application 1 issues an application-break message to RHF 1 to inform RHF 2 and Application 2 of a problem in the communication between applications. Application 2 responds by initiating a connection termination sequence. The second sequence begins the same as the first sequence with Application 1 issuing an application-break message to RHF. In this case, Application 2 responds by resetting the connection to a known state so processing can continue on that connection.



Application 1	RHF 1	RHF 2	Application 2	Message	Meaning
→	→			FC/BRK/R FLUSH CONTROL MESSAGE+	Application break.
		→		FC/BRK/R	Application break.
		←		CON/END/R	End connection.
	←			DISCONNECT PATH+	
←				CON/CB/R	Connection broken.
→				CON/END/R	End connection.
	→			DISCONNECT PATH+	
		→		CON/END/N	Connection ended.
←				CON/END/N	Connection ended.

A serious error was detected by Application 1 and reported to RHF 1. RHF 2 informs Application 2 of the break. Application 2 decides to end the connection.

+ FLUSH CONTROL MESSAGE and DISCONNECT PATH are messages defined within a lower level protocol. They are shown in this manual only to indicate how the various messages of a sequence fit together.

Figure 3-9. Application Break/Reset Message Sequences (Sheet 1 of 2)

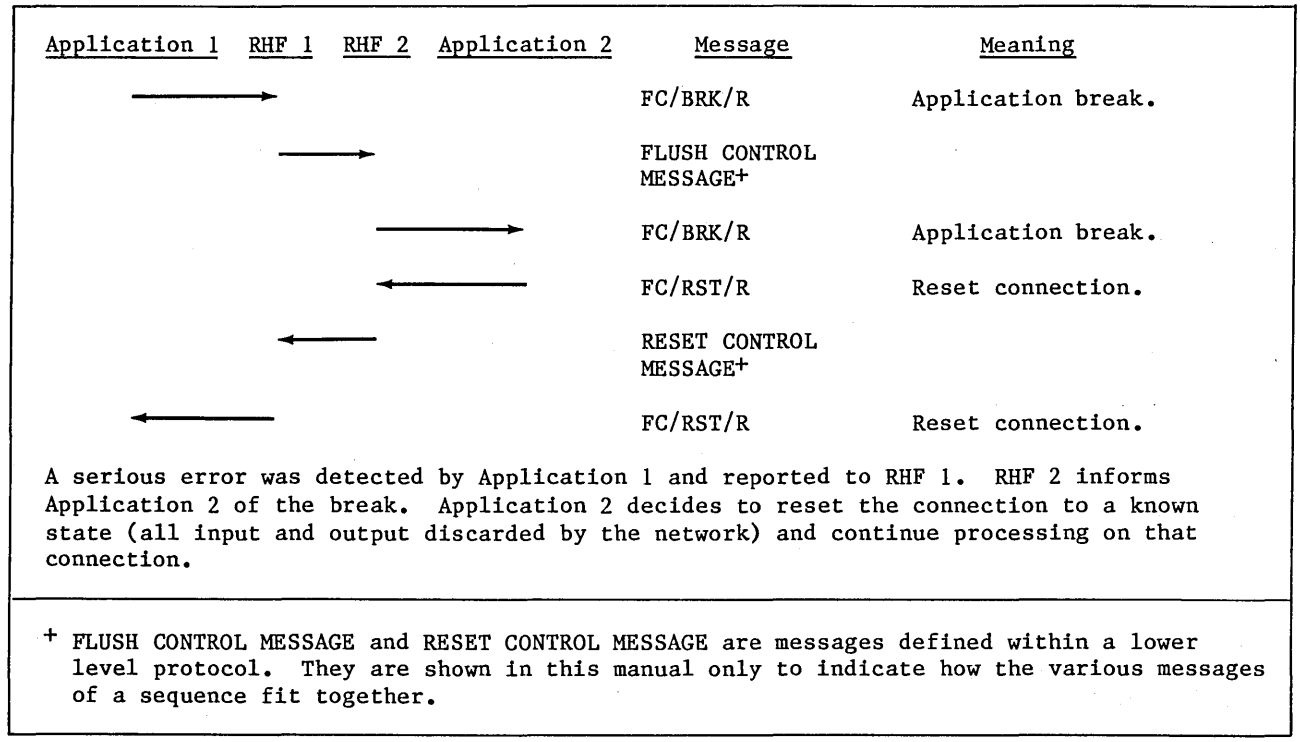


Figure 3-9. Application Break/Reset Message Sequences (Sheet 2 of 2)

CONVERT MODE SELECTION MESSAGE SEQUENCE

When sending or receiving unacknowledged data blocks in character format, an application program may request RHF to present application data blocks in either 6-bit display code format or 7-bit ASCII (zero-byte terminated) format instead of the standard network ASCII (unit-separator terminated) format. Data conversion enabling is dependent on the availability of NAD code conversion resources and may be disallowed as an installation option. Depending on the reason code returned in the CM/SEL/A supervisory message, an application program may elect to wait and retry to select code conversion, perform its own code conversion, or abandon the data transfer. Refer to FET I/O Interface in section 4 and the 380-170 Network Access Device Hardware Reference Manual for more information on code conversion and convert modes.

Figure 3-10 shows a successful convert mode selection message sequence followed by an unsuccessful convert mode selection message sequence.

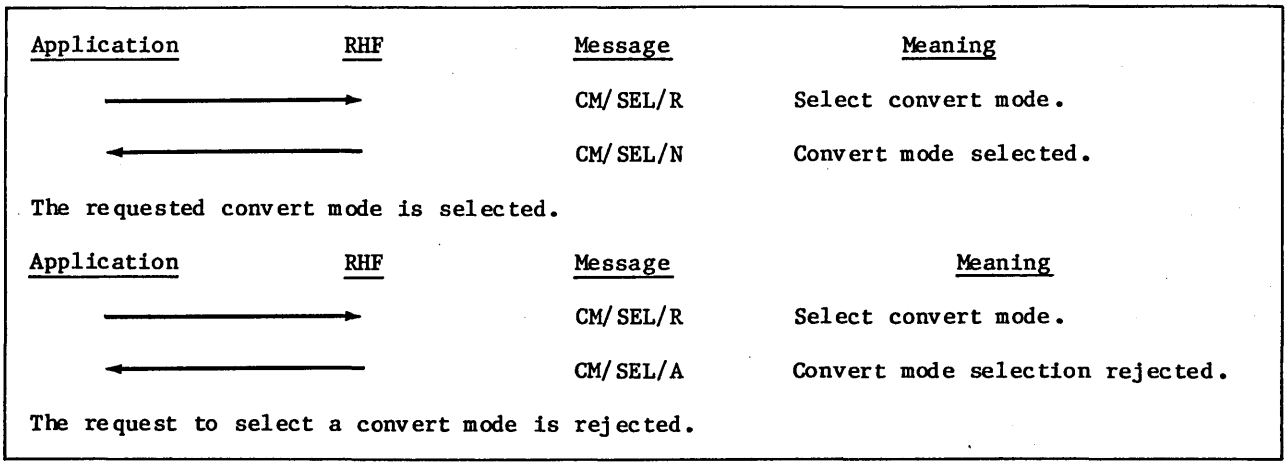


Figure 3-10. Convert Mode Selection Message Sequences

CONVERT MODE EXIT MESSAGE SEQUENCE

After completing a data transfer sequence of unacknowledged data blocks in character format with a data conversion mode selected, an application program must terminate the convert mode selection to return the connection to normal data presentation mode. Figure 3-11 shows the convert mode exit message sequence.

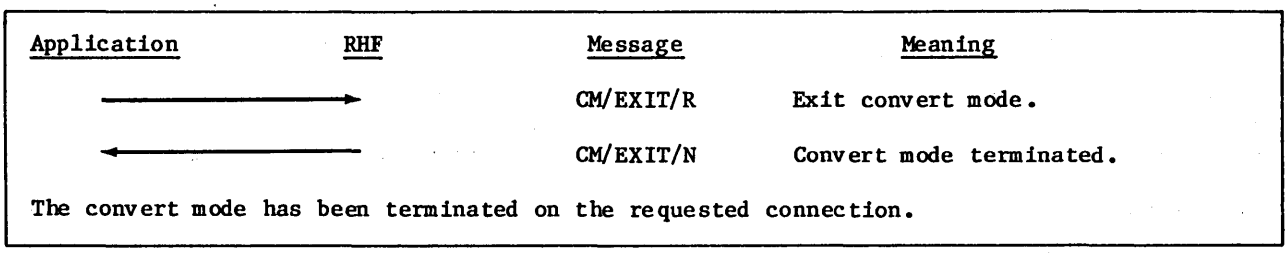
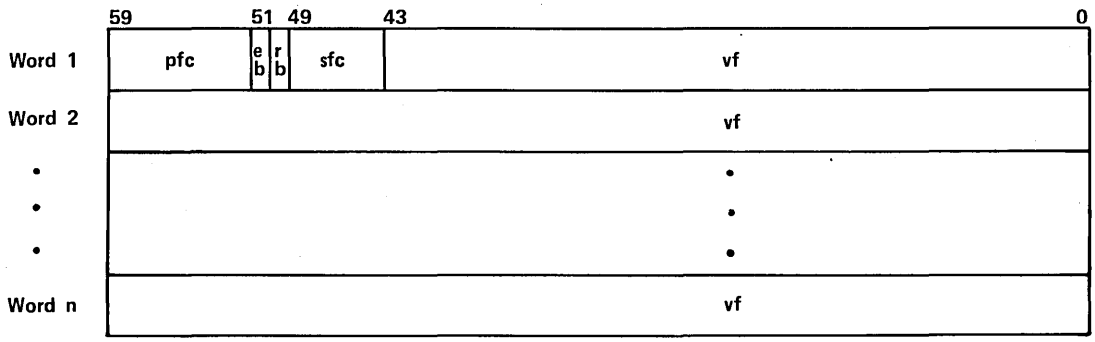


Figure 3-11. Convert Mode Exit Message Sequence

SUPERVISORY MESSAGE FORMATS

The general format of a supervisory message is:



Field	Bit(s)	Description
pfc	59-53	Primary function code. In the following supervisory message descriptions, reserved symbolic mnemonics are given for pfc. These mnemonics can be any of those listed in section 2 under Supervisory Message Blocks.
eb	51	Error bit; set to 1 to indicate an error or abnormal response.
rb	50	Response bit; set to 1 to indicate a normal response.
sfc	49-44	Secondary function code. In the following supervisory message descriptions, reserved symbolic mnemonics are given for sfc. These mnemonics can be any of those listed in section 2 under Supervisory Message Blocks, provided the secondary function code is valid for the primary function code used.
vf	43-0	Variable fields. These fields are defined in the field descriptions following each message format. Variable fields may be extended into words 2 through n as necessary; n has the value $2 \leq n \leq 63$.

If a supervisory message response is required (refer to individual message descriptions), the recipient fills in any fields that are returned, sets the error bit or response bit (as appropriate) and sends the message back.

APPLICATION CONNECTION REQUEST

When one application program needs to be connected to another, the first application program sends a supervisory message request to RHF asking for establishment of a logical connection. RHF can either accept or reject the logical connection request. If it rejects the request, one message sequence ends. If it does not reject the request, one of two message sequences can occur, depending on whether the logical connection is successfully established. These three possible message sequences are shown in figure 3-1.

RHF permits more than one logical connection to exist between two application programs. However, neither application program can send any data messages to a connection until connection initialization processing has been completed.

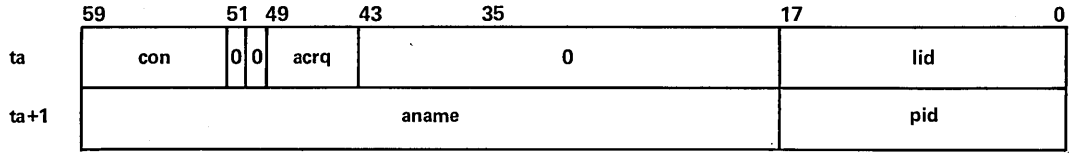
If either program cannot complete or service the logical connection, it can reject the connection request by issuing the application-connection-rejected message. When this occurs, the other application program must exchange the connection-broken, end-connection, and connection-ended supervisory messages with RHF. No further action is required by the rejecting application program.

If either application program does not follow these message sequences, a logical-error supervisory message is issued. Refer to Error Reporting later in this section.

A logical connection established between two application programs does not necessarily have the same application connection number for both applications, because RHF assigns the application connection number to each end of the logical connection independently.

Application-Connection-Request Message Format

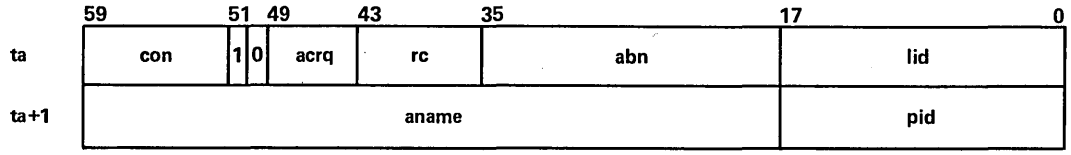
The application-connection-request (CON/ACRQ/R) message is sent to RHF from an application program when the program wants to establish a logical connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
con	Primary function code 63_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CON.
acrq	Secondary function code 2. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol ACRQ.
lid	A logical identifier of the host where the requested application program resides. The lid field is three alphanumeric 6-bit display code characters. If this field is 0, the pid field is used to determine the host where the requested application program resides. Either the lid field or pid field must be nonzero. If both fields (lid and pid) are nonzero, the specified lid must be associated with the specified pid in the network definition.
aname	Name of the application program with which this program wishes a logical connection to be established. This name can be one to seven 6-bit display coded letters and digits, left-justified with blank fill; the first character is always alphabetic. The name placed in this field must be the element name used to define the requested application program in the local configuration file. This field can be accessed with the reserved symbol CONANM, as described in section 5.
pid	The physical identifier of the host where the requested application program resides. The pid field is three alphanumeric, 6-bit display code characters. If this field is 0, the lid field is used to determine the host where the requested application program resides. Either the lid field or pid field must be nonzero. If both fields (lid and pid) are nonzero, the specified lid must be associated with the specified pid in the network definition.

Application-Connection-Rejected Message Format

The application-connection-rejected (CON/ACRQ/A) message is sent to an application program from RHF to reject the request for a logical connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
con	Primary function code 63_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CON.
acrq	Secondary function code 2. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol ACRQ.
rc	Reason code specifying the cause for rejecting the connection request. This field is made up of two 4-bit subfields rc1 (bits 43-40) and rc2 (bits 39-36). In a future release, the rc1 field will be used in combination with the rc2 field to qualify the reason for the connection rejection. Currently, the rc1 field is always zero.

The application program uses the rc2 field to determine what action to take when it receives a CON/ACRQ/A message. This field can have one of the following values.

Value	Description
1	The source host detected an error in the connection request.
2	The destination (remote) host detected an error in the connection request.
3	The source network or subsystem temporarily cannot make the connection.
4	The destination network or subsystem temporarily cannot make the connection.
5	The source network or subsystem indefinitely cannot make the connection.
6	The destination network or subsystem indefinitely cannot make the connection.

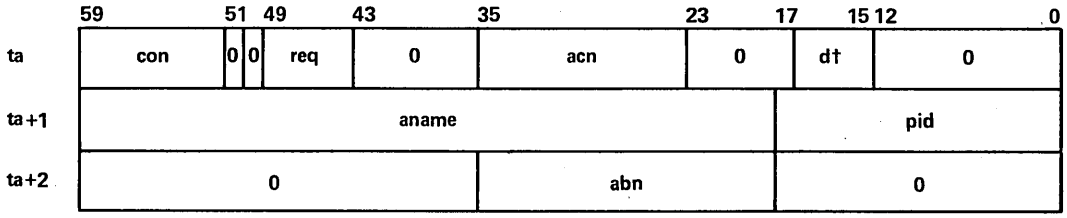
Field	Description
	If rc2 has a value 1 or 2, the application program should notify the user and/or operator that the connection is not possible rather than trying to reestablish the connection. If rc2 has a value of 3 or 4, the application program can retry the CON/ACRQ/R message after waiting a short time. If rc2 has a value of 5 or 6, the application program can retry the CON/ACRQ/R message after waiting a longer time. This field can be accessed with the reserved symbol RC, as described in section 5.
abn	The application block number of the initial application-connection-request message to which this message is a reply.
lid	A logical identifier of the host where the requested application program resides. This field is a copy of the lid field in the CON/ACRQ/R message to which this message is a response.
aname	Name of the application program with which connection was requested. This field will always be the same as the aname field in the CON/ACRQ/R message to which this message is a response. This field can be accessed with the reserved symbol CONANM, as described in section 5.
pid	The physical identifier of the host where the requested application program resides. This field is a copy of the pid field in the CON/ACRQ/R message to which this message is a response.

RHF CONNECTION REQUEST

This request is sent to an application program from RHF when a remote application program wants to be connected to a local application program or when a remote RHF has accepted the local application program's request to connect to one of the remote application programs. RHF has previously validated the connection. The receiving application program may either accept or reject the request.

Connection-Request Message Format

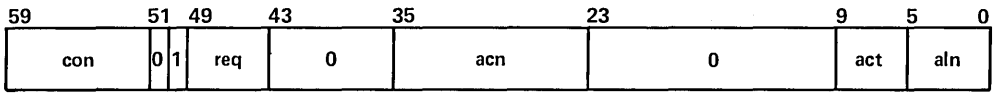
The connection-request (CON/REQ/R) message is sent to an application program from RHF to request a logical connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
con	Primary function code 63_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of reserved symbol CON.
req	Secondary function code 0. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol REQ.
acn	Application connection number assigned to this logical connection, if the connection is accepted; $1 \leq$ minacn $\leq$ acn $\leq$ maxacn ≤ 4095, where minacn and maxacn are minimum and maximum values established by the application program in its NETON call. This field can be accessed with the reserved symbol CONACN, as described in section 5.
dt	Device type associated with this connection. For application-to-application connections, the dt field always contains the value 5.
aname	Name of the application program requesting that the connection be initiated. This name is one to seven 6-bit display coded letters and digits, left-justified with blank fill; the first character is always alphabetic. The name in this field is the element name used to identify the requesting program in the local configuration file. This field can be accessed with the reserved symbol CONANM, as described in section 5.
pid	The physical identifier of the host where the application program resides that issued the application-connection-request message.
abn	The application block number of the initial application-connection-request message.

Connection-Accepted Message Format

The connection-accepted (CON/REQ/N) message is sent to RHF from an application program to accept a connection request. The message format is:



Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
con	Primary function code 63_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CON.
req	Secondary function code 0. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol REQ.
acn	Application connection number assigned by RHF to the program end of the logical connection being established. The value placed in this field must be the value used in the CON/REQ/R message to which this message is a response. This field can be accessed with the reserved symbol CONACN, as described in section 5.
act	Application character type, specifying the form of character byte packing that the application program requires for input data blocks from the logical connection. This field can have one of the following values:

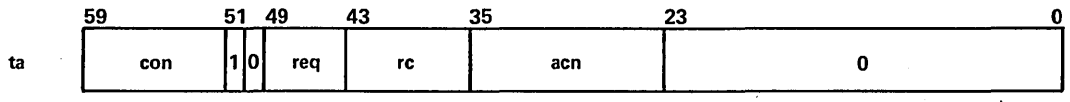
Value	Description
1	60-bit transparent characters, packed one character per central memory word.
2	8-bit characters in 8-bit bytes, packed 7.5 characters per central memory word.
3	8-bit characters in 12-bit bytes, packed five characters per central memory word, right-justified with zero fill within each byte.
4	6-bit display coded characters in 6-bit bytes, packed 10 characters per central memory word.

The act value declared applies only to input on the connection and can be changed by a DC/CICT/R supervisory message at any time during the existence of this logical connection. This field can be accessed with the reserved symbol CONACT, as described in section 5.

aln	Application list number assigned by the application program to the logical connection; $0 \leq aln \leq maxcon$, where maxcon is the maximum number of connections allowed to the application program. maxcon = maxacn - minacn + 1, where maxacn and minacn are the values used by the application program when it issued a NETON request. This field can be accessed with the reserved symbol CONALN, as described in section 5.

Connection-Rejected Message Format

The connection-rejected (CON/REQ/A) message is sent to RHF from an application program to reject a connection request. The message format is:



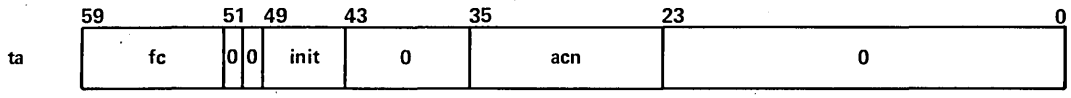
Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
con	Primary function code 63_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CON.
req	Secondary function code 0. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol REQ.
rc	Reason code for rejecting the connection request. This field may be used by an application program, but is ignored by RHF. This field can be accessed with the reserved symbol RC, as described in section 5.
acn	Application connection number assigned by RHF to the program end of the logical connection being rejected. The value placed in this field must be the value used in the CON/REQ/R message to which this message is a response. Upon receipt of this message, RHF can reuse this application connection number for a different logical connection with the same program. This field can be accessed with the reserved symbol CONACN, as described in section 5.

INITIALIZING CONNECTION

A logical connection is established when RHF sends an application program a connection-request message and the application program responds with a connection-accepted message. RHF then sends an initialize-connection message to the application program and the application program acknowledges this message with a connection-initialized message. The logical connection is now fully initialized and can be used for input and output. Any input received on the logical connection is queued and available to the application program.

Initialize-Connection Message Format

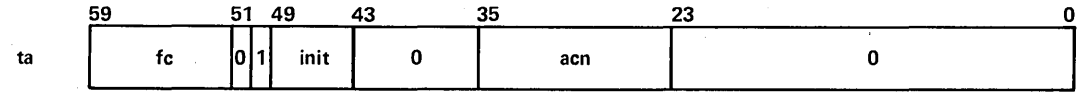
The initalize-connection (FC/INIT/R) message is sent to an application program from RHF to request connection initialization processing. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
fc	Primary function code 83_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol FC.
init	Secondary function code 7. This field can be accessed with the reserved symbol SFC, as defined in section 5. Its value is defined as the value of the reserved symbol INIT.
acn	Application connection number assigned by RHF to the program end of the logical connection that has been initialized. This value is the same as that used in previous CON/REQ/R and CON/REQ/N messages. This field can be accessed with the reserved symbol FCACN, as described in section 5.

Connection-Initialized Message Format

The connection-initialized (FC/INIT/N) message is sent to RHF from an application program to acknowledge a connection initialization processing request. The message format is:



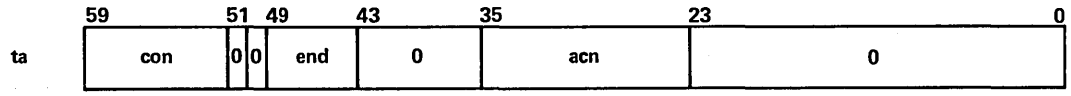
Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
fc	Primary function code 83_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol FC.
init	Secondary function code 7. This field can be accessed with the reserved symbol SFC, as defined in section 5. Its value is defined as the value of the reserved symbol INIT.
acn	Application connection number assigned by RHF to the program end of the logical connection that has been initialized. The value placed in this field must be the value used in the FC/INIT/R message to which this message is a response. This field can be accessed with the reserved symbol FCACN, as described in section 5.

TERMINATING CONNECTION

A local application program or remote application program may terminate a logical connection at any time. The disconnection sequence can be voluntary or involuntary from an application program's viewpoint.

End-Connection Message Format

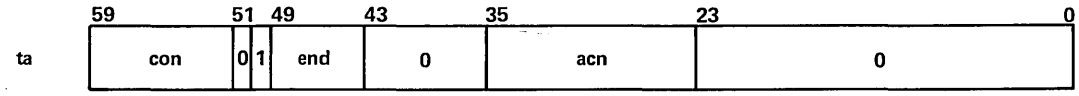
The end-connection (CON/END/R) message is sent to RHF from an application program when all processing on a particular logical connection has been completed. The message format is:



Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
con	Primary function code 63_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CON.
end	Secondary function code 6. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol ENDD.
acn	Application connection number assigned by RHF to the program end of the logical connection being terminated. The value placed in this field must be the value used in the CON/REQ/R message beginning this message sequence. Upon receipt of this message, RHF issues a response message and can reuse this application connection number for a different logical connection with the same program. This field can be accessed with the reserved symbol CONACN, as described in section 5.

Connection-Ended Message Format

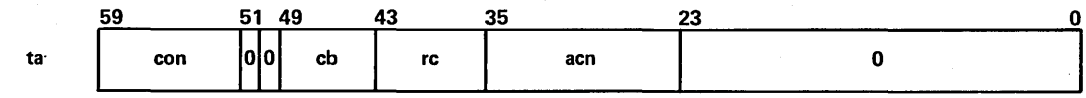
The connection-ended (CON/END/N) message is sent to an application program from RHF to acknowledge an end logical connection request. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
con	Primary function code 63_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CON.
end	Secondary function code 6. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol ENDD.
acn	Application connection number assigned by RHF to the program end of the logical connection that has been terminated by the CON/END/R message to which this message is a response. After issuing this message, RHF can reassign this application connection number to another logical connection with the same program. This field can be accessed with the reserved symbol CONACN, as described in section 5.

Connection-Broken Message Format

The connection-broken (CON/CB/R) message is sent to an application program from RHF when the logical connection is broken. The logical connection could have been broken because the other application program ended the connection, the program failed, or the program temporarily does not have the resources to establish the logical connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
con	Primary function code 63_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CON.
cb	Secondary function code 5. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol CB.
rc	Reason code, specifying the cause of the broken connection. This field can have one of the following values:

Value	Description
1	Communication has been lost with the application program at the other end of the logical connection.
2	RHF broke the connection. This can occur if this message is a response to a CON/REQ/N message containing an invalid field or if RHF temporarily does not have the resources to complete the connection.

This field can be accessed with the reserved symbol RC, as described in section 5.

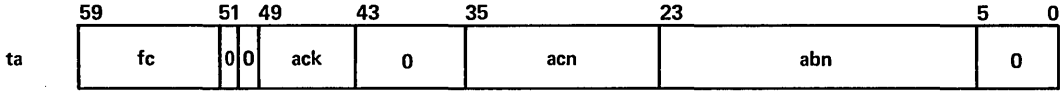
acn	Application connection number assigned by RHF to the program end of the logical connection being broken. This number is always one for which the application program has previously received a CON/REQ/R message. This field can be accessed with the reserved symbol CONACN, as described in section 5.

BLOCK DELIVERED

RHF sends a block-delivered message to an application program when a block has been delivered to its destination. After receipt of this supervisory message, the application program may output another block on the logical connection, since the number of blocks outstanding on the connection has been reduced by one.

Block-Delivered Message Format

The block-delivered (FC/ACK/R) message is sent to an application program from RHF to inform the application program that its data block has been delivered successfully. The message format is:



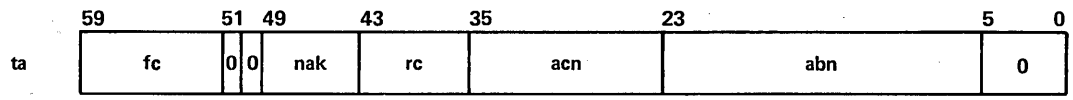
Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
fc	Primary function code 83_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol FC.
ack	Secondary function code 2. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol ACK.
acn	Application connection number assigned by RHF to the program end of the logical connection on which the block was delivered. This value is always nonzero and is the acn value used by the program in the application block header sent with the delivered block. This field can be accessed with the reserved symbol FCACN, as described in section 5.
abn	Application block number assigned by the application program to the delivered block. This value is the abn value used by the program in the application block header sent with the delivered block. This field can be accessed with the reserved symbol FCABN, as described in section 5.

BLOCK NOT DELIVERED

RHF sends a block-not-delivered message to an application program when for some reason a block is not delivered to its destination. The application program may attempt to recover by sending the same block again, but the block will not be in its original sequential position if other blocks have been sent following the undelivered block.

Block-Not-Delivered Message Format

The block-not-delivered (FC/NAK/R) message is sent to an application program from RHF to inform the application program that its data block has not been delivered. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
fc	Primary function code 83_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol FC.
nak	Secondary function code 3. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol NAK.

Field	Description
rc	Reason code, explaining why the block was not delivered. This field can have the following values:

Value	Description
1	Network software error caused loss of the block in transit; the block can be retransmitted but might be delivered out of sequence with subsequently transmitted blocks.
2	The resource limit has been reached. Output has been attempted several times, but the NAD does not accept the output.
3	The NAD is not available. Either the connection is in the process of being terminated or the NAD is disabled.

This field can be accessed with the reserved symbol RC, as described in section 5.

acn	Application connection number assigned by RHF to the program end of the logical connection on which the block was not delivered. This value is always nonzero and is the acn value used by the program in the application block header sent with the block that was not delivered. This field can be accessed with the reserved symbol FCACN, as described in section 5.
abn	Application block number assigned by the application program to the block that was not delivered. This value is the abn value used by the program in the application block header sent with the block that was not delivered. This field can be accessed with the reserved symbol FCABN, as described in section 5.

DATA CONVERSION CONTROL

Data exchanged on a logical connection is converted to and from 6-bit display code or 7-bit ASCII code at the discretion of the application program. RHF converts data in a given block according to the application character type associated with that block.

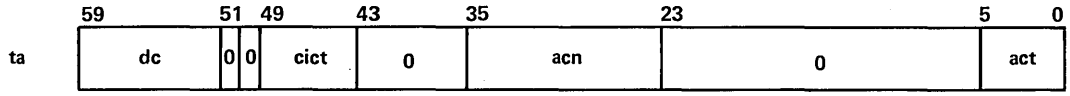
Output data sent by an application program has an application character type associated with it on a block-by-block basis. When the application program needs to change the conversion performed for output data on a given connection, it simply changes the act field value used in the block header of each data block.

Input data received by an application program has an application character type associated with the logical connection on which the data blocks are received. The application character type associated with a given block of input data is assigned by the application program when the logical connection is established; this assignment is part of the connection-accepted (CON/REQ/N) message. When the application program needs to change the convertion performed for input data on a given connection, it changes the act field value associated with the logical connection by issuing a change-input-character-type (DC/CICT/R) message.

This message can be issued at any time the logical connection exists, after the application program has issued a connection-initialized (FC/INIT/N) message for the connection. No response is necessary to the change-input-character-type message, but the change takes effect immediately.

Change-Input-Character-Type Message Format

The change-input-character-type (DC/CICT/R) message is sent to RHF from an application program to change the data conversion performed for input data on a given logical connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
dc	Primary function code $C2_{16}$. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol DC.
cict	Secondary function code 0. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol CICT.
acn	Application connection number assigned by RHF to the program end of the logical connection when it was established. The value placed in this field must be the value associated with an existing connection and used in the FC/INIT/N supervisory message that completed initialization of the connection. This field can be accessed with the reserved symbol DCACN, as described in section 5.
act	Application character type specifying the form of character byte packing that the application program requires for all future input data blocks from the logical connection. The value declared replaces the value previously declared by the application program for this connection in a CON/REQ/N or DC/CICT/R message. This field can have one of the following values:

Value	Description
1	60-bit transparent characters, packed one character per central memory word.
2	8-bit characters in 8-bit bytes, packed 7.5 characters per central memory word.
3	8-bit characters in 12-bit bytes, packed five characters per central memory word, right-justified with zero fill within each byte.
4	6-bit display coded characters in 6-bit bytes, packed 10 characters per central memory word.

The act value declared applies only to input on the connection and can be changed by another DC/CICT/R message at any time during the existence of this logical connection.

HOST SHUTDOWN

Conditions sometimes require the local operator to terminate network operations including executing application programs.

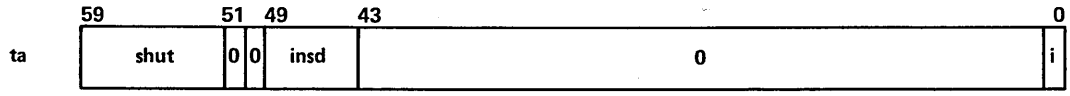
The operator has two shutdown options available. An idle-down option can be selected that permits gradual termination of operations, usually as a normal part of network service. The idle-down option is selected by entering the IDLE command for RHF (refer to the RHF K display in the NOS 2 Analysis Handbook or the NOS/BE Operator's Guide). A disable option also can be selected; this option requests immediate termination of application program operations, and can either follow selection of the idle-down option or be independently selected to reflect an abnormal operational condition. The disable option is selected by entering the STOP command for RHF (refer to the NOS 2 Analysis Handbook) or the DROP command or KILL command for RHF (refer to the NOS/BE Operator's Guide).

The source of a shutdown decision is not meaningful to an application program. The type of shutdown, however, determines the shutdown processing that should be performed by the application program.

RHF does not attempt to force the termination of applications that do not terminate in response to an idle-down request. Normal idle-down termination of network operations is dependent upon correct application behavior. Applications that do not eventually terminate after receiving an idle-down request must be dropped by the local operator. This then permits normal termination of network operations.

Host-Shutdown Message Format

The host-shutdown (SHUT/INSD/R) message is sent to an application program from RHF to inform the application program that the local operator has requested termination of network operations. The two forms of the host-shutdown message are determined by the i field value. An i field value of 0 indicates the message was caused by the operator selecting the idle-down option (IDLE command). An i field value of 1 indicates the message was caused by the operator selecting the disable option (STOP, DROP, or KILL command). The application program does not issue a response to this message. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supevisory message.
shut	Primary function code 42_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol SHUT.
insd	Secondary function code 6. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol INSD.

Field	Description
i	Indicator for type of shutdown option. This field can have one of the following values:

Value	Description
0	Idle-down option: a warning message of a pending network shutdown. RHF will not permit any more logical connections to be established, but the application program can inform existing connections of the shutdown, fetch queued input data from all connections, and voluntarily end all connections before issuing a NETOFF call.
1	Disable option: network shutdown is beginning. The application program cannot send or receive blocks on any existing connection and no more logical connections can be established. The application program must issue a NETOFF call immediately without ending any existing connections.

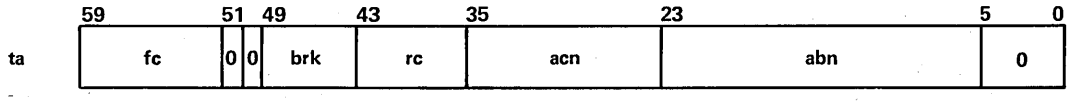
This field can be accessed with the reserved symbol SHUTYP, as described in section 5.

APPLICATION BREAK/RESET

A local or remote application program sends an application-break message through RHF to inform the other application program of a communications problem between the two applications. The application-break message is initiated by the application program detecting the problem. The application program receiving the message can either terminate the logical connection or reset the connection to allow further processing from a known state of the connection.

Application-Break Message Format

The application-break (FC/BRK/R) message is sent from one application program to another application program through RHF when either program detects a communications problem. The message format is:

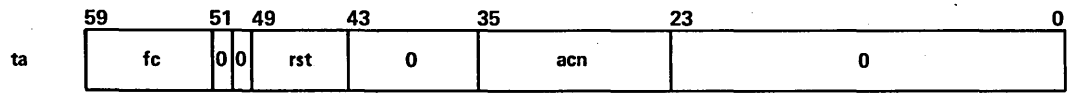


Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent or received.
fc	Primary function code 83_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol FC.
brk	Secondary function code 0. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol BRK.

Field	Description
rc	Reason code that explains the cause of the break condition. The reason code is generated by the application program sending the application-break (FC/BRK/R) message.
acn	Application connection number assigned by RHF to the program end of the logical connection on which the break occurred. This field always contains a nonzero value previously used by the application program in a CON/REQ/R message and must be used by the application program in a subsequent FC/RST/R message before data transmission on the connection is again possible. This field can be accessed with the reserved symbol FCACN, as described in section 5.
abn	Application block number assigned by the application program to the last block for which a FC/ACK/R message was generated before the break condition occurred. This field contains a 0 only when no FC/ACK/R messages were generated before the break occurred. This field is not used for FC/BRK/R messages sent from an application program to RHF. This field can be accessed with the reserved symbol FCABN, as described in section 5.

Application-Reset Message Format

The application-reset (FC/RST/R) message is sent from one application program to another application program through RHF in response to an application-break message. The message format is:



Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent or received.
fc	Primary function code 83_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol FC.
rst	Secondary function code 1. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol RST.
acn	Application connection number assigned by RHF to the program end of the logical connection to be reset. This value is always nonzero and must be the acn value received by the application program in a previous FC/BRK/R message. This field can be accessed with the reserved symbol FCACN, as described in section 5.

LIST POLLING CONTROL

Connection list polling control consists of enabling or disabling the fetching of input blocks from a single logical connection when the list that connection is assigned to is polled for input. All connections are initially enabled for list processing without application program action. Each time the application program polls the list number it has associated with a specific logical connection, blocks queued from that connection can be returned to the porgram. If the application program requires the list to be polled without returning any blocks queued from the connection, the program issues a turn-list-processing-off message that causes the next poll of the list to exclude the connection. This turn-list-processing-off message effectively disables list processing for the connection. RHF does not acknowledge this messge and it remains in effect until cancelled by a turn-list-processing-on message.

The turn-list-processing-on message is issued by the application program to enable list processing for a specific logical connection. This message causes the next poll of the list number associated with the indicated connection to include the connection's data block queue. RHF does not acknowledge this message. If the message is issued when list processing already has been enabled for the connection, no error occurs. The message remains in effect until cancelled by a turn-list-processing-off message.

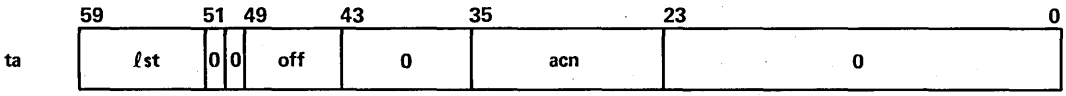
Enabling list processing for a logical connection does not necessarily cause a queued block to be returned from that connection the next time the connection's list is polled. The next poll of the list causes a scan of application connection numbers starting from the application connection number following the block returned by the last poll. The next application connection number that has input available will be the connection from which input is returned. Disabled connections are skipped during the polling process; enabled connections are included.

The list number associated with a specific logical connection is determined by the application program when it accepts the logical connection. This list number can be changed while the connection exists by issuing a change-connection-list message. RHF does not acknowledge this message, but the change is effective at the time of the next poll of the new list number. After the application program issues a change-connection-list message, polls of the old list number cannot return blocks queued from the affected connection.

Connection list polling is performed through application calls to the FIP routine NETGETL [refer to Fetch List Input (NETGETL) in section 4].

Turn-List-Processing-Off Message Format

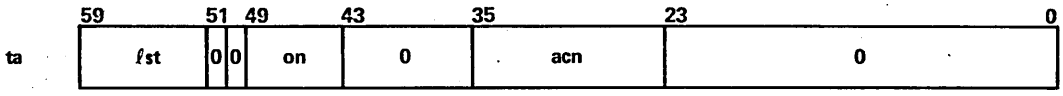
The turn-list-processing-off (LST/OFF/R) message is sent to RHF from an application program to disable list processing for a specific logical connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
lst	Primary function code CO_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol LST.
off	Secondary function code 0. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol OFF.
acn	Application connection number assigned by RHF to the program end of the logical connection for which list processing is being disabled. The value used in this field must be the value used in a CON/REQ/R message processed by the application program. This field can be accessed with the reserved symbol LSTACN, as described in section 5.

Turn-List-Processing-On Message Format

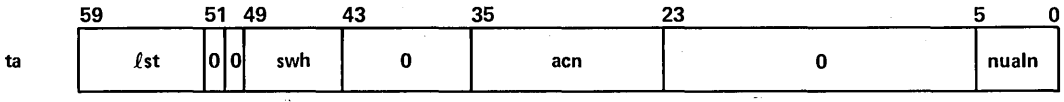
The turn-list-processing-on (LST/ON/R) message is sent to RHF from an application program to enable list processing for a specific logical connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
lst	Primary function code CO_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol LST.
on	Secondary function code 1. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol ON.
acn	Application connection number assigned by RHF to the program end of the logical connection for which list processing is being enabled. The value used in this field must be the value used in a CON/REQ/R message processed by the application program. This field can be accessed with the reserved symbol LSTACN, as described in section 5.

Change-Connection-List Message Format

The change-connection-list (LST/SWH/R) message is sent to RHF from an application program to change the list number associated with a specific logical connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
lst	Primary function code $C0_{16}$. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol LST.
swh	Secondary function code 2. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol SWH.
acn	Application connection number assigned by RHF to the program end of the logical connection being switched to a new connection list. The value used in this field must be the value used in a CON/REQ/R message processed by the application program. This field can be accessed with the reserved symbol LSTACN, as described in section 5.
nualn	New application list number to which the logical connection is reassigned; $0 \leq$ nualn $\leq$ maxcon. Where maxcon is the maximum number of connections allowed to the application program. Maxcon = maxacn - minacn + 1, where maxacn and minacn are the values used by the application program when it issued a NETON request. This field can be accessed with the reserved symbol LSTALN, as described in section 5.

ERROR REPORTING

The primary mechanism used by RHF to indicate logic errors to an application program is the logical-error supervisory message. The application program does not send a response to this supervisory message.

As indicated by the reason codes included in the message field descriptions, many conditions are considered to be logical errors by RHF. The simpler conditions are completely defined within the field descriptions; more complex conditons are described in the following paragraphs.

The rc field value of 1 is received when an application character type other than 1 is used in an output block header for a supervisory message.

The rc field value of 4 is received when:

- The application connection number involved is out-of-range for the application program and, therefore, nonexistent. Connection numbers not yet assigned to the application program, or numbers with negative values, are out of range.
- Application connection number 0 is specified in a change-connection-list or turn-list-processing-off supervisory message. Such messages are ignored.
- The application connection number involved was used during the period when the application-to-application connection message sequence for it was still incomplete. An application connection number cannot be used other than in that supervisory message sequence, until the application program has issued the connection-initialized response message.

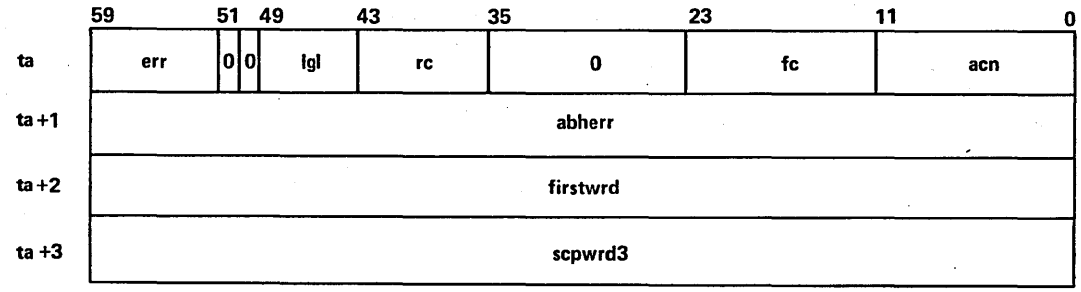
The rc field value of 6 is received when RHF attempts to protect itself from infinite loops by an application program or from application program flaws in supervisory message processing logic. To prevent the application program from deadlocking the network in such cases, a partial limit is imposed on the number of logical errors permitted for an application program. This limit applies only to logical-error messages queued for the application program; the limit keeps the program from committing large numbers of errors in output transmissions without periodically fetching queued supervisory messages to identify the errors. The limit is implemented as follows:

- Each time RHF sends a logical-error message to the application program, a limit counter for the program is incremented by one.
- Each time the application program fetches all queued supervisory messages it has outstanding, the limit counter for the program is reset to zero.
- When the limit counter for the program reaches 50, a logical-error message with the rc field value of 6 is queued for the program. Until the application program fetches all queued supervisory messages it has outstanding, any output transmission by the program that causes a logical-error message condition is discarded by RHF without being processed.

RHF imposes an additional limit on supervisory messages that may be queued for an application program. If more than 200 supervisory messages are queued for an application program, RHF aborts that program.

Logical-Error Message Format

The logical-error (ERR/LGL/R) message is sent to an application program from RHF to indicate an error condition. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
err	Primary function code 84_{16}. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol ERR.
lgl	Secondary function code 1. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol LGL.
rc	Reason code, identifying the cause of the message. This field can contain one of the following values:

Value	Description
1	An invalid act value was specified in the block header of an output data block or in a DC/CICT/R message.
2	An invalid tlc was encountered; either the value in the block header of an output block was greater than 2043, or the length of the block exceeded 409 central memory words, or the length of an output supervisory message exceeded 63 central memory words.
3	An invalid abt value was specified in the block header of an output block; either the value was 0 or greater than 3.
4	An invalid acn value was encountered in the block header of an output data block or in a supervisory message.
5	Not used.
6	More than 50 ERR/LGL/R messages have been issued to the application program, and the program still has supervisory messages queued for it. Until the application program fetches all queued supervisory messages, all output transmission causing ERR/LGL/R messages is ignored.
7	An incorrect or illogical supervisory message was encountered; either the combined primary and secondary function codes of the message are not a valid value, or the message is not permitted as part of supervisory message sequences currently in progress with the application program, or the application program sent a supervisory message on a connection other than 0.
8-12	Reserved for NAM AIP compatibility.
13	An invalid FET I/O function was encountered.
14	An invalid nualn value was specified in a change-connection-list (LST/SWH/R) message.

This field can be accessed with the reserved symbol RC, as described in section 5.

Field	Description
fc	Function code associated with the application request that caused the ERR/LGL/R message. This field can contain one of the following values:

Value	Application Request
1	NETON
2	NETOFF
3	NETWAIT
4	NETGET
5	NETGETL
6	NETPUT
7	FET I/O
other	Invalid application request.

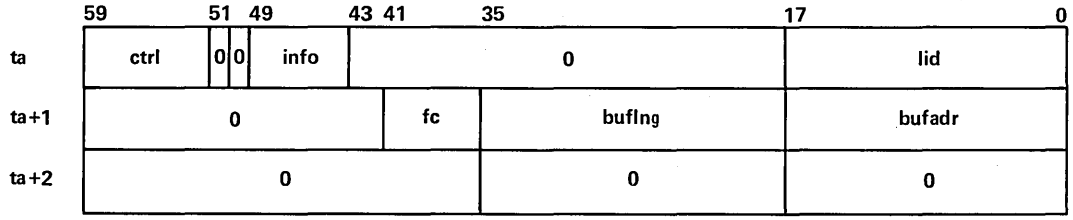
acn	Application connection number associated with the application request that caused the ERR/LGL/R message.
abherr	Application block header word associated with the block or supervisory message that caused the ERR/LGL/R message. This field contains a zero word if RHF cannot provide a copy of the block causing the problem. This field can be accessed with the reserved symbol ERRABH, as described in section 5.
firstwrd	The first 60 bits of the block or supervisory message causing the ERR/LGL/R message are placed in this field if RHF can supply the information. This field contains a zero word unless the abherr field is nonzero and the tlc field within that block header is nonzero. This field can be accessed with the reserved symbol ERRMSG, as described in section 5.
scpwrd3	Third word of the CALLSS parameter block of the application request that caused the ERR/LGL/R message. This field depends on the function code (fc) requested. The CALLSS parameter block fields are generated by the FIP modules described in section 4.

RHF CONTROL INFORMATION REQUEST

An application program can request specific control information from RHF by selecting one of three function codes when issuing a control-information-request message.

Control-Information-Request Message Format

The control-information-request (CTRL/INFO/R) message is sent to RHF from an application program when the application program needs information from RHF to control its processing. The information requested is defined by the function code field in the control-information-request message. The message format is:



Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
ctrl	Primary function code $C1_{16}$. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CTRL.
info	Secondary function code A_{16}. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol INFO.
lid	Used with function code 0. A logical identifier of the host computer to be used in an application-connection-request (CON/ACRQ/R) message. The lid field is three alphanumeric, 6-bit display code characters.
fc	Function code, specifying the type of control information request that is being sent to RHF. This field can have one of the following values:

Value	Description
0	Determine physical identifier (pid) from logical identifier (lid). This function allows an application program to query RHF to determine what host computer (pid) the program will likely be connected to when it issues an application-connection-request (CON/ACRQ/R) message. In other words, this function allows the application program to determine the mapping that RHF may use for lid to pid. RHF responds to this function by issuing either a control-information-accepted message or a control-information-rejected message.

Field	Description

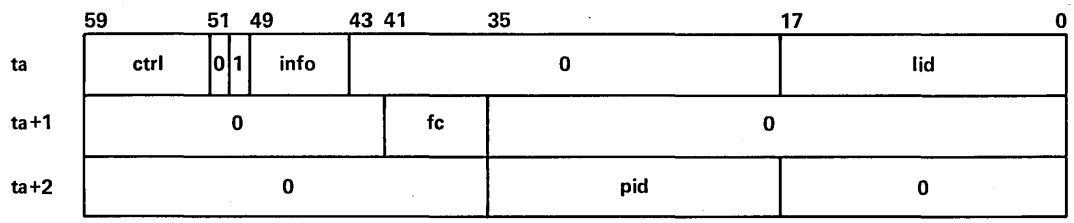
Value	Description
	This function may be required by any application program that allows multiple user interfaces using different lids to use only one host-to-host (pid-to-pid) connection path through RHF and LCN. For example, ITF allows transmission of interactive data for many users on one host-to-host connection. For each new lid specified by a user wishing access to ITF, the program issues this function to determine what mapping (lid-to-pid) RHF will likely use. If RHF returns a pid to which ITF has already established a connection, then ITF uses that connection for the new user. If RHF returns a pid to which ITF has not established a connection, then ITF issues an application-connection-request (CON/ACRQ/R) message to RHF for connection to the new pid.
1	Set end-of-job connection (NOS/BE only). This function allows an application program to set the end-of-job connection to RHF. This feature may be used by a privileged application program to retain a connection to RHF across job steps. The next request to RHF after issuing this function must be a NETON request. The application program is aborted if it attempts an RHF request other than NETON after the end-of-job connection is established. RHF automatically releases the end-of-job connection when it receives the NETON request. RHF does not reply to this function. Upon return from the FIP call to a NETPUT statement to issue this function, the end-of-job connection has been set. If the application program does not have the proper privileges to issue this function, the program is aborted.
2	Return network description table to buffer. This function allows an application program to obtain a copy of the network description table being used by RHF. The network description table is reserved for system use only. RHF does not reply to this function. Upon return from the FIP call to a NETPUT statement to issue this function, the network description table has been written to the specified buffer. If the application program does not have the proper privileges to issue this function, the program is aborted. The application program also aborts if either the specified buffer address (bufadr) exceeds the application field length or the buffer address plus the buffer length (buflng) exceeds the application field length. If the buffer length is great enough, the entire network description table is written to the buffer. If the buffer length is too small, only as many words of the network description table as fit are written to the buffer.

buflng Used with function code 2. 18-bit length of the buffer to which the network description table is to be written.

bufadr Used with function code 2. 18-bit address of the buffer in the application program to which the network description table is to be written.

Control-Information-Accepted Message Format

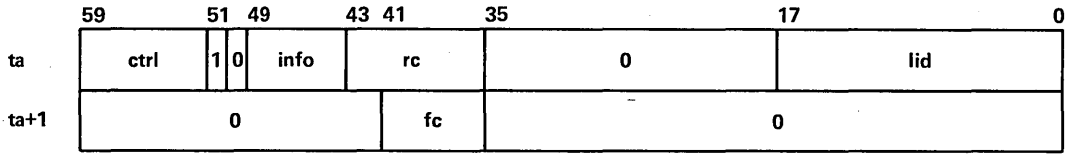
The control-information-accepted (CTRL/INFO/N) message is sent to an application program from RHF to indicate that the requested control information has been exchanged. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
ctrl	Primary function code $C1_{16}$. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CTRL.
info	Secondary function code A_{16}. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol INFO.
lid	Logical identifier of the host computer. The value used in this field must be the value used in the CTRL/INFO/R message.
fc	This field contains 0 and is a copy of the function code field from the CTRL/INFO/R message to which this message is a response.
pid	Physical identifier of the host computer to which the application program is likely to be connected if it issues an application-connection-request (CON/ACRQ/R) message using the lid field. The pid field is three alphanumeric, 6-bit display code characters.

Control-Information-Rejected Message Format

The control-information-rejected (CTRL/INFO/A) message is sent to an application program from RHF to indicate inability to process the control information request. The message format is:



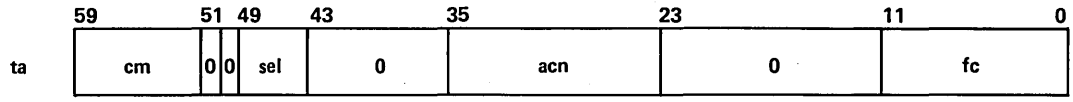
Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
ctrl	Primary function code $C1_{16}$. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CTRL.
info	Secondary function code A_{16}. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol INFO.
rc	Reason code 1, indicating that the lid field is invalid or no paths are defined to the specified lid.
lid	Logical identifier of the host computer. The value used in this field must be the value used in the CTRL/INFO/R message.
fc	This field contains 0 and is a copy of the function code field from the CTRL/INFO/R message to which this message is a response.

CONVERT MODE SELECTION

An application program sends a convert mode selection request to RHF to select a conversion mode for unacknowledged data blocks in character format that are sent or received using the FET I/O interface (NETUXFR) on a connection. The conversion mode affects only one direction of the full-duplex connection. An application program must not select a conversion mode for both directions of a given connection simultaneously. RHF may either accept or reject the request.

Convert-Mode-Select-Request Message Format

The convert-mode-select-request (CM/SEL/R) message is sent to RHF from an application program to select a conversion mode for a given connection. The message format is:

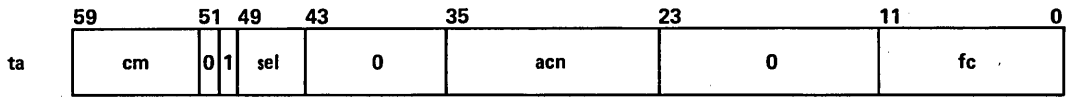


Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
cm	Primary function code $C3_{16}$. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CM.
sel	Secondary function code 0. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol SEL.
acn	Application connection number assigned by RHF to the program end of the logical connection requesting a convert mode selection. This value is the same as that used in a CON/REQ/R message processed by the application program.
fc	Function code, specifying the desired code conversion mode. This field can have one of the following values:

Value	Direction	Description
1	Send	Convert 7-bit ASCII in 12-bit bytes to network ASCII.
2	Send	Convert 6-bit display code to network ASCII.
3	Receive	Convert network ASCII to 7-bit ASCII in 12-bit bytes.
4	Receive	Convert network ASCII to 6-bit display code.

Convert-Mode-Select-Accepted Message Format

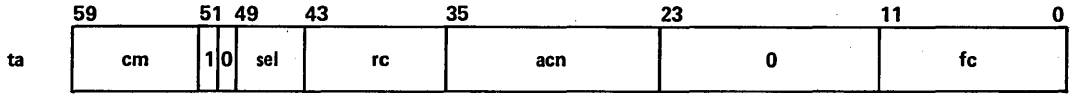
The convert-mode-select-accepted (CM/SEL/N) message is sent to an application program from RHF to acknowledge a successful convert mode selection. From the time conversion mode is selected until conversion mode is terminated, all data transfers in the selected direction must be made through the FET I/O interface (NETUXFR); NETGET and NETPUT cannot be used. Also, no other convert mode selections can be requested for the connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
cm	Primary function code $C3_{16}$. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CM.
sel	Secondary function code 0. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol SEL.
acn	Application connection number assigned by RHF to the program end of the logical connection that selected the conversion mode. The value placed in this field must be the value used in the CM/SEL/R message to which this message is a response.
fc	Function code. This field contains a copy of the function code field from the CM/SEL/R message to which this message is a response.

Convert-Mode-Select-Rejected Message Format

The convert-mode-select-rejected (CM/SEL/A) message is sent to an application program from RHF to notify the application program of an unsuccessful convert mode selection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
cm	Primary function code $C3_{16}$. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CM.
sel	Secondary function code 0. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol SEL.
rc	Reason code, identifying the reason that the convert mode selection request was rejected. This field can contain one of the following values:

Value	Description
0	Not used.
1	The requested conversion mode is not available.
2	The conversion mode resources temporarily are not available.

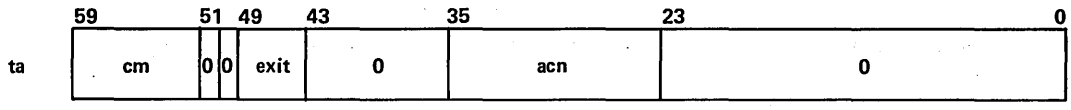
acn	Application connection number assigned by RHF to the program end of the logical connection that requested the conversion mode. The value placed in this field must be the value used in the CM/SEL/R message to which this message is a response.
fc	Function code. This field contains a copy of the function code field from the CM/SEL/R message to which this message is a response.

CONVERT MODE TERMINATION

An application program sends a conversion mode exit request to RHF to terminate a conversion mode for a connection.

Convert-Mode-Exit-Request Message Format

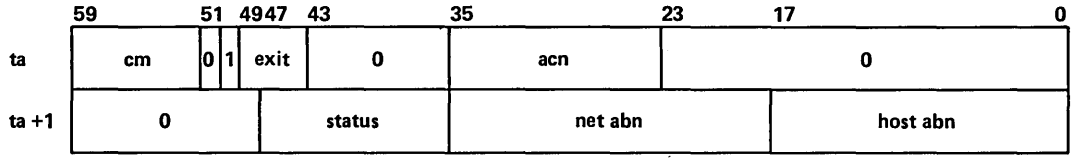
The convert-mode-exit-request (CM/EXIT/R) message is sent to RHF from an application program to initiate a conversion mode termination message sequence for a given connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area from which this supervisory message is sent.
cm	Primary function code $C3_{16}$. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CM.
exit	Secondary function code 1. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol EXIT.
acn	Application connection number assigned by RHF to the program end of the logical connection requesting a conversion mode termination. The value placed in this field must be the value used in the CM/SEL/R message used to select the conversion mode.

Convert-Mode-Exit-Accepted Message Format

The convert-mode-exit-accepted (CM/EXIT/N) message is sent to an application program from RHF to notify the application program that the conversion mode has been terminated for the requested connection. The message format is:



Field	Description
ta	Symbolic address of the application program's text area receiving this supervisory message.
cm	Primary function code $C3_{16}$. This field can be accessed with the reserved symbol PFC, as described in section 5. Its value is defined as the value of the reserved symbol CM.
exit	Secondary function code 1. This field can be accessed with the reserved symbol SFC, as described in section 5. Its value is defined as the value of the reserved symbol EXIT.
acn	Application connection number assigned by RHF to the program end of the logical connection that terminated the conversion mode. The value placed in this field must be the value used in the CM/EXIT/R message to which this message is a response.
status	Status code, identifying the status of the conversion mode termination. This field can contain one of the following values:

Value	Description
0	The conversion mode termination is successful.
1	The conversion mode termination is successful, but partial blocks were discarded.

net abn	The application block number associated with the last network data block sent or received in the selected conversion mode direction.
host abn	The application block number associated with the last application data block sent or received in the selected conversion mode direction. This value should match the final host abn value in the NETUXFR FET.

FACILITY INTERFACE PROGRAM 4

This section describes the facility interface program (FIP) statements used by an application program to access RHF, to control network processing, and to transmit and receive the supervisory messages described in section 3.

FIP is a set of relocatable CPU modules that supports a subset of the entry points defined by the network host products application interface program (AIP).

FIP MODULES

The appropriate FIP module is brought into an application program's field length when the application program is loaded, provided the program refers to one of the entry points of that FIP module. In order for the appropriate FIP module to be loaded, the system library LCNLIB containing FIP modules must be specified in the load step. If an application program uses an overlay structure, the overlay that contains NETON must remain in memory until NETOFF is called.

The FIP entry points, equivalent entry points, and associated functions are:

Entry Points	Function
NETON/RHFON	Establish access to the network.
NETOFF/RHFOFF	End access to the network.
NETWAIT/RHFWAIT	Suspend an application program temporarily.
NETDBG/RHFDBG	Turn debugging on or off.
NETGET/RHFGET	Get data from a specified logical connection.
NETGETL/RHFGETL	Get data from one of a list of logical connections.
NETPUT/RHFPUT	Send data to a specified logical connection.
NETUXFR/RHFUXFR	Get or send data using the FET I/O interface.

The FIPXFR module is also brought into an application program's field length when the application program is loaded, provided that the program overlay refers to one of the entry points of the FIPXFR module. The FIPXFR module must remain in the user application program's field length as long as there is an outstanding file transfer in progress. FIPXFR entry points, equivalent entry points, and associated functions are:

Entry Points	Function
NETXFR/RHFXFR	Initiate a file transfer to or from a specified logical connection.
NETXFRC/RHFXFRC	Update (continue) outstanding file transfers in progress.

If an application program uses an overlay structure, the overlay that contains NETXFR must remain in memory until all file transfers are completed. Because NETXFR uses the common memory manager (CMM), the CMM parameter must be used for any overlay loading. Any memory management done by the application program must be done through calls to CMM.

FIP FUNCTION PROCESSING

All FIP functions, except NETXFR, take the parameters passed to them by an application program and format a corresponding call to RHF using the CALLSS macro. The operating system passes the call to RHF, if possible. If not, it aborts the application program's control point (NETON is an exception discussed later in this section). If the call is passed to RHF, it processes the function. If a fatal error is detected, the application program's control point is aborted. Otherwise, return parameters, if any, pass back to the application program and control returns to FIP. Then, FIP completes its processing and returns control to the application program.

Besides performing the basic processing described above, some FIP functions must perform the following additional processing.

- If there is outstanding NETXFR file-transfer activity, NETGET calls INTAMSG, a subroutine in the FIPXFR module, to intercept supervisory messages destined for application connection numbers with outstanding file transfers in progress.
- Four of the functions (NETPUT, NETWAIT, NETGET, and NETGETL) call NETXFRC to continue the processing of each active file-transfer one step further, if possible.
- If the application program sets a debug flag, NETPUT, NETGET, NETGETL, and NETOFF write debug information to the dayfile.
- When control returns to NETON after the call to RHF, NETON must check for error conditions and set an error status for the application program.

The FIPXFR module has two entry points: NETXFR and NETXFRC. An application program calls NETXFR to initiate a file transfer. NETXFR performs the following functions to initiate the file transfer.

- Ensures that no more than the installation-defined maximum number of file transfers are active.
- Turns list processing off for this application connection number.
- Calls common memory manager (CMM) to obtain more field length.
- Builds the file environment table (FET) for the RHF file.
- Builds the FET for the local file.
- Initiates a combined input/output (CIO) call to open the local file.
- Selects NAD code conversion if necessary and available.

NOTE

NETXFR assumes an application character type (act) value of 2. Applications using different act values must change to an act value of 2 by issuing a change-input-character-type (DC/CICT/R) message prior to calling NETXFR.

If immediate return is requested, NETXFR returns control to the application program. Otherwise, NETXFR continues calling NETXFRC, places itself in RECALL, and checks for supervisory messages until the file transfer is complete.

NETXFRC can also be called directly by an application program, or by NETGET, NETGETL, NETPUT, NETWAIT, or NETXFR. NETXFRC performs the processing necessary to complete a file transfer. Each time NETXFRC is called, it attempts to process each active file transfer in progress until it receives a wait response.

If the application program calls NETXFR and requests that file transfers be completed before returning control, NETXFR also intercepts supervisory messages destined for application connection numbers with active file transfers. If the application program calls NETXFR with immediate return specified, the application program must make periodic calls to NETGET or NETGETL to intercept supervisory messages destined for application connection numbers with active file transfers.

SYNTAX OF FIP STATEMENTS

FIP statements are used in COMPASS programs or in programs written in high-level languages such as FORTRAN. In most high-level languages, only positional parameters can be used; FIP statements conform to this syntactical requirement and, therefore, do not permit the use of keywords. The interpretation attached to a given parameter is determined solely by its location within the string of parameters of each FIP statement. All input parameters must be supplied; there are no defaults.

Mnemonic variables identifying each parameter are defined in the statement descriptions, along with any coding constraints imposed on them. Commas delimit parameters in all languages; the significance of blanks depends on the language used. Unless otherwise specified, all values supplied for parameters should be decimal integers.

COMPASS programs can use the FIP macros, symbolic names, and symbolic values defined in systems text FIPTEXT.

NETWORK ACCESS STATEMENTS

An application program uses two FIP statements to begin and end access to the network's resources. The NETON statement must be used before the program can use any other FIP statement except NETOFF. The NETOFF statement must be used after all FIP functions are completed to cause the FIP portion of the application program to perform vital housekeeping tasks.

CONNECT TO NETWORK (NETON)

The NETON statement performs the following functions:

- Identifies the application program to RHF for validation.
- Causes FIP to establish communication with RHF.
- Identifies a word to be used for communication from FIP to the application program, outside of the supervisory message mechanism.
- Informs RHF of limitations on the number of logical connections the application program can handle.

An application program must successfully complete a NETON call before it can use any FIP statement other than NETOFF. If another FIP statement is used before a NETON call is successfully completed, FIP aborts the job and issues a message to the job's dayfile; the incorrectly placed call has no effect.

If the application program is successfully validated by RHF, the program has access to the network as long as a NETOFF statement is not issued and communication with RHF continues. If the application program validation is not successful, the program is aborted.

NETON Calling Sequence

The format of the NETON calling sequence is:

```
CALL NETON (aname,nsup,status,minacn,maxacn)
```

Parameter	Description
aname	An input parameter specifying in 6-bit display code the name of the application program that wishes to establish communications with RHF. This name can be one to seven 6-bit, alphabetic and numeric characters, but the first character must be alphabetic. This parameter must be left-justified, with blank fill. Refer to appendix C for application program names reserved by CDC.
nsup	A return parameter; nsup is the symbolic address of the supervisory status word for communication from FIP to the application program. Only the upper 5 bits of this status word are used. The upper 3 bits of this word are reserved. The next 2 bits are set to report the status of the data message and supervisory message queuing performed by FIP. This word need not contain zeros at the time of the NETON call and should not be changed at any time by the application program.

Parameter	Description

Bit	Description
59	Reserved for CDC.
58	Reserved for CDC.
57	Reserved for CDC.
56	Input in queue bit. This bit is set to 1 if FIP has data messages queued for the application program. The bit is valid after the application program issues a NETWAIT, NETGET, NETPUT, NETON, or NETGETL call. This bit is set to 0 on return from any of these calls when no data messages remain queued for the program.
55	Supervisory message in queue bit. This bit is set to 1 on return from FIP routines NETGET, NETGETL, NETPUT, NETWAIT, or NETON if supervisory messages are queued on application connection number 0 for this program. A value of 1 is advisory only; no program action is required. This bit is set to 0 on return from any of these calls when no supervisory messages remain queued for the program.

status — A return parameter; status is the symbolic address of the NETON call status word. On return from the NETON call, the content of this word indicates RHF's disposition of the application program's NETON attempt. The values of status can be:

Value	Description
0	NETON was successful and network access is established.
1	NETON was unsuccessful because RHF was not at a control point or did not have enough resources to service this application program (too many application programs running at the same time).
2	NETON was rejected because too many application programs are presently accessing the network with the same name specified for the aname parameter.
3	NETON was rejected because the application program has a status of disabled. The program must be rerun after the local operator has enabled it.
4	NETON was rejected because the aname parameter is invalid. RHF aborts the calling application program.
5	NETON was rejected because the application program does not have the proper privileges to issue a NETON call. RHF aborts the calling application program.
6	NETON was rejected because the minacn or maxacn values are outside the range of those allowed for the calling application program. RHF aborts the calling application program.
7	NETON was rejected because the application program has already established network access. RHF aborts the calling application program.

Parameter	Description
minacn	An input parameter specifying the smallest application connection number the application program can process; $0 < minacn \leq maxacn \leq 4095$. RHF does not attempt to complete any more connections to the program after all connections from minacn through maxacn (inclusive) are in use.
maxacn	An input parameter specifying the largest application connection number the application program can process; $0 < minacn \leq maxacn \leq 4095$. RHF does not attempt to complete any more connections to the program after all connections from minacn through maxacn (inclusive) are in use.

NOTE

The range between minacn and maxacn cannot exceed the maximum number of connections defined in the application table for that application program. Refer to the NOS 2 Analysis Handbook or NOS/BE Installation Handbook for details about application table definition.

DISCONNECT FROM NETWORK (NETOFF)

The NETOFF statement performs the following functions:

- Breaks FIP communication with RHF.
- Clears internal tables so that the application program can issue another NETON call, if necessary.

Normally the NETOFF statement is used only after all processing of logical connection activities is finished and the program is prepared to end connection with the network. After the NETOFF call is completed, no FIP statement other than NETON can be used. The NETOFF call breaks any logical connection still existing between the application program and another application program and prevents RHF from attempting to establish any new connection. After the NETOFF statement is processed, the application program continues to execute under control of the operating system.

An application program should always issue a NETOFF call before terminating. Otherwise, RHF informs other application programs with which connections exist that the program has failed.

NETOFF Calling Sequence

The format of the NETOFF calling sequence is:

```
CALL NETOFF
```

The NETOFF call has no parameters.

MESSAGE BLOCK INPUT/OUTPUT STATEMENTS

Input can be obtained from a logical connection either by its individual connection number, or according to its membership in a list of connections. FIP statements permit an application program two calls for input (NETGET and NETGETL), and one call for output (NETPUT).

FETCH INPUT (NETGET)

Each NETGET call transfers one data block or supervisory message block for the logical connection specified in the call. The block header is placed in the application program's block header area, and the data or message body is placed in the application program's text area.

If no data or message block is available from the indicated connection, FIP returns a null block; that is, a header word with an application block type of 0 is placed in the header area, and the text area is unchanged from what it contained after any previous transfer.

The application program indicates the size of its buffer in each NETGET call. If a data or message block larger than this buffer size is queued from the specified connection, the block remains queued. FIP copies the header word of the block into the application program's block header area, sets the ibu bit of the header to 1 to indicate the condition, and places the actual length of the queued block in the tlc field of the header. The application program's text area is unchanged from what it contained after any previous transfer. To obtain the still-queued block, the program must issue another NETGET call indicating a buffer size sufficient to accommodate the queued block.

If the application program's text area is larger than the data or message block transferred by the NETGET call, the portion of the text area after the last word used for the block remains unchanged from what it contained after any previous transfer. If the transferred block does not completely fill the last word used for it, all character positions in the last word used are altered by the transfer. Only the leftmost character positions of the last word included in the block header word tlc field value contain meaningful data.

NETGET can be used to obtain a supervisory message from application connection number 0. NETGET can also be used to obtain data blocks from application connection numbers other than 0. Data blocks are never queued on application connection number 0.

NETGET Calling Sequence

The format of the NETGET calling sequence is:

```
CALL NETGET (acn,ha,ta,tlmax)
```

Parameter — **Description**

acn — An input parameter specifying the application connection number of the logical connection from which a data or message block is requested. This parameter can have the values:

Value	Description
0	Transfer one supervisory message.
acn	Transfer one data block from the logical connection with the indicated acn. Where acn is $0 < \text{minacn} \leq \text{acn} \leq \text{maxacn}$.

ha — A return parameter specifying the symbolic address of the application program's header area. The header area always contains an updated application block header after return from the call.

ta — A return parameter specifying the symbolic address of the first word of the buffer array constituting the text area for the application program. On return from the call, the text area contains the requested block if a block was available and the text area was large enough. The text area identified by ta should be at least tlmax words long.

tlmax — An input parameter specifying the maximum block length in central memory words that the application program can accept. The value declared for tlmax should be less than or equal to the length of the text area identified in the same call; if tlmax is greater than the length of the text area, the block transfer resulting from the NETGET call might overwrite a portion of the application program. The maximum value needed for tlmax is a function of the block size used by the logical connection for input to the program and of the application character type the program has specified for input from the logical connection. The following ranges are valid:

act	Range	Block Size
1	$1 \leq \text{tlmax} \leq 384$	60-bit transparent characters, packed one character per central memory word.
2	$1 \leq \text{tlmax} \leq 273$	8-bit characters, packed 7.5 characters per central memory word.
3	$1 \leq \text{tlmax} \leq 409$	8-bit characters, right-justified in 12-bit bytes with zero fill, packed five characters per central memory word.
4	$1 \leq \text{tlmax} \leq 205$	6-bit display code characters, packed 10 characters per central memory word.

A tlmax value of 0 can be declared but always results in an input-block-undeliverable condition.

FETCH LIST INPUT (NETGETL)

Each NETGETL call causes RHF to select, on a rotating basis, one of the logical connections from a specified list. RHF only chooses a connection that has data or message blocks queued and has not been turned off by a supervisory message. One data or message block is transferred from the selected connection for each call to NETGETL; the block header is placed in the application program's block header area and the data or message body is placed in the application program's text area.

Each NETGETL statement causes the logical connection list to be scanned only once. Scanning begins with the connection immediately following the connection from which a block was previously transferred. The first connection on the list is examined after the last connection on the list. Scanning ends when a logical connection with a queued input block is found. If no logical connection has a queued input block, scanning ends with the connection preceding the one at which scanning started.

If data or supervisory message blocks are not available from any logical connection on the list, FIP returns a null block. A header word with an application block type of 0 is placed in the header area, and the text area is unchanged from its content before the NETGETL call. Null blocks are not returned from each logical connection.

The application program indicates the size of its buffer in each NETGETL call. If a data or message block larger than this buffer size is available for transfer, the block remains queued. FIP copies the header word of the block into the application program's block header area, sets the ibu bit of the header to 1 to indicate the condition, and places the actual length of the queued block in the tlc field of the header. The application program's text area is unchanged from what it contained after any previous transfer. To obtain the still-queued block, the program must issue a separate NETGETL call indicating a buffer size sufficient to accommodate the queued block. The connection pointer within the list is incremented regardless of whether a transfer occurs, so the same connection is not involved in a second NETGETL call.

If the application program's text area is larger than the data or message block transferred by the NETGETL call, the portion of the text area after the last word used for the block remains unchanged from what it contained after any previous transfer. If the transferred block does not completely fill the last word used for it, all character positions in the last word used are altered by the transfer. Only the leftmost character positions of the last word included in the block header word tlc field value contain meaningful data.

NETGETL can be used to obtain a supervisory message from application connection number 0. Application connection number 0 is always part of application list number 0. When a NETGETL call specifying input from list 0 is issued, any supervisory messages queued for the application program are returned before list scanning continues to other connection numbers on list 0. Data blocks on connection numbers other than 0 can also be obtained when their connection numbers have been assigned to list 0.

NETGETL Calling Sequence

The format of the NETGETL calling sequence is:

```
CALL NETGETL (aln,ha,ta,tlmax)
```

Parameter — **Description**

aln — An input parameter specifying the application list number to be scanned for a queued block. This parameter can have the values:

Value	Description
0	First obtain all supervisory messages queued on application connection number 0, then obtain any data blocks queued on application connection numbers other than 0.
aln	Obtain one data block from one connection on the indicated list. Where aln is $1 \leq aln \leq$ maximum number of connections allowed by the application program.

ha — A return parameter specifying the symbolic address of the application program's header area. The header area always contains an updated application block header after return from the call.

ta — A return parameter specifying the symbolic address of the first word of the buffer array constituting the text area for the application program. On return from the call, the text area contains the requested block if a block was available and the text area was large enough. The text area identified by ta should be at least tlmax words long.

tlmax — An input parameter specifying the maximum block length in central memory words that the application program can accept. The value declared for tlmax should be less than or equal to the length of the text area identified in the same call; if tlmax is greater than the length of the text area, the block transfer resulting from the NETGETL call might overwrite a portion of the application program. The maximum value needed for tlmax is a function of the block size used by the connection for input to the program and of the application character type the program has specified for input from the connection. The following ranges are valid:

act	Range	Block Size
1	$1 \leq tlmax \leq 384$	60-bit transparent characters, packed one character per central memory word.
2	$1 \leq tlmax \leq 273$	8-bit characters, packed 7.5 characters per central memory word.
3	$1 \leq tlmax \leq 409$	8-bit characters, right-justified in 12-bit bytes with zero fill, packed five characters per central memory word.
4	$1 \leq tlmax \leq 205$	6-bit display code characters, packed 10 characters per central memory word.

A tlmax value of 0 can be declared but always results in an input-block-undeliverable condition.

SEND OUTPUT (NETPUT)

Each NETPUT call requests FIP to form a data or message block from the information located in the application program's block header and text areas. The calling application program must construct a complete data or message block header. The text portion of the block can be either a data block or a supervisory message block. The block formed by FIP is sent to the logical connection specified in the block header.

NETPUT can be used to send supervisory messages to application connection number 0. NETPUT can also be used to send data blocks to application connection numbers other than 0. Data blocks are never sent to application connection number 0.

NETPUT Calling Sequence

The format of the NETPUT calling sequence is:

```
CALL NETPUT (ha,ta)
```

Parameter	Description
ha	An input parameter specifying the symbolic address of the application program's block header area. The block header area must contain a valid application block header word.
ta	An input parameter specifying the symbolic address of the application program's text area. The text area must contain a valid data block or supervisory message block correctly described by the contents of the block header area.

PROCESSING CONTROL STATEMENT

The processing control statement NETWAIT can cause or reduce processing delays to alter the application program's efficiency. The NETWAIT call is used in conjunction with the supervisory status word established by the application program in its NETON call (nsup parameter).

SUSPEND PROCESSING (NETWAIT)

The NETWAIT statement performs the following functions:

- Allows an application program to make itself a candidate for rollout/swapout by the operating system or otherwise suspend its processing.
- Allows the application program to declare a maximum time for processing suspension.
- Allows the application program to delay resumption of processing until input is available for it on any of its logical connections.
- Causes the supervisory status word (NETON nsup parameter) for the application program to be updated on return from the NETWAIT call.

Calls to NETWAIT with nonzero flag values always suspend processing when suspension is possible. Calls to NETWAIT with zero flag values suspend processing only when no input is available.

NETWAIT calls with a flag value of 0 should be made only after all outstanding supervisory messages have been fetched by the application program. A NETWAIT call made while any supervisory message remains queued always results in immediate return to the application program, regardless of whether any other input is available. Such calls require unnecessary additional processing by FIP and the application program.

An application program that repeatedly polls the network for input (using NETGET or NETGETL calls) should contain NETWAIT calls. If such programs omit frequent NETWAIT calls, severe performance degradation can result; if online debugging of such application programs is expected, small time limits should be used for the job while it is in the debugging phase.

NETWAIT Calling Sequence

The format of the NETWAIT calling sequence is:

```
CALL NETWAIT (time,flag)
```

Parameter	Description
time	An input parameter, $1 \leq time \leq 4095$, specifying the number of seconds for which the application program should be suspended. If a value of 0 is declared, a default value of 1 is used; if a value greater than 4095 is declared, a default value of 4095 is used.
flag	An input parameter specifying the conditions under which processing should be resumed. This parameter can have the values:

Value	Description
0	Return from NETWAIT call (resume processing) when input is available from any logical connection, or when the time period declared by the time parameter has elapsed. When a flag value of 0 is declared and input is available immediately, the value declared for the time parameter is ignored.
1	Return from NETWAIT call (resume processing) only when the time period declared by the time parameter has elapsed, regardless of whether input is available from any logical connection.

FILE TRANSFER STATEMENTS

The file transfer statements NETXFR and NETXFRC enable an application program to more efficiently transmit or receive large quantities of data from a remote application program. These statements control only the actual transfer portion of a file transfer operation; the startup processing and termination processing must be done by the application program intiating the transfer.

TRANSFER STORED FILE (NETXFR)

Each NETXFR call either transmits a file residing on a mass storage device or receives a file to be stored on a mass storage device. The application program specifies the application connection number, the name of the file to be transferred, whether the transfer is a read or write operation, the conditions under which processing should resume, and the format of the data to be transferred. The application program also monitors the status of the transfer operation.

NETXFR Calling Sequence

The format of the NETXFR calling sequence is:

```
CALL NETXFR (acn,fname,op,status,wait,dd,tmout,abl,facil,bsize,ackw,ickval)
```

Parameter	Description
acn	An input parameter specifying the application connection number of the logical connection on which the file is to be transferred. Where acn is minacn $\leq$ acn $\leq$ maxacn.
fname	An input parameter specifying in 6-bit display code the name of the file to be transferred.
op	An input parameter specifying whether the operation is a read or a write across the network. This parameter can have the values:

Value	Description
0	The file is to be read from the network and written to disk.
1	The file is to be read from disk and written to the network.

status	A return parameter; status is the symbolic address of the NETXFR call status word. When the application program regains control, it must monitor or check the status word to determine the status of the transfer. This parameter can have the values:

Value	Description
-2	The file transfer is terminating.
-1	The file transfer is initiating.

Parameter	Description

Value	Description
0	The file transfer is in progress.
1	The file transfer is complete (no error).
2	Reserved for CDC.
3	A connection-broken message was received from the network. The file transfer terminates.
4	A protocol error was detected. The file transfer terminates.
5	The transfer operation timed out while waiting for a response from the network. The file transfer terminates.
6	An output block was not delivered. The file transfer terminates.
7	An application program issued more than the installation-defined number of file transfers. The file transfer terminates.
8	The specified application connection number is not within the allowable range. The file transfer terminates.
9	A second file transfer was attempted on the specified application connection number. The file transfer terminates.
10	A CIO error was detected. The file transfer terminates.
11	A premature termination was received from the remote application program. The file transfer terminates.
12	The file transfer was completed with no errors, but RHF has requested an idle-down.
13	The file transfer was terminated by an error and RHF has requested an idle-down.
14	RHF has issued a disable warning.
15	RHF has detected a NAD I/O error. The file transfer terminates.
16	The file specified by the fname parameter does not reside on a mass storage device. The file transfer terminates.
17	NETXFR was unable to initiate the requested code file transfer (dd parameter value 3 or 4) because network code conversion is enabled and all network conversion resouces are currently in use (NETXFR waits for up to tmout seconds before terminating the file transfer). The file transfer terminates.
18-32	Reserved for NAM AIP compatibility.

Parameter	Description
wait	An input parameter specifying the condition under which the application program processing should resume. This parameter can have the values:

Value	Description
0	Return from NETXFR call (resume processing) when the file transfer is completed.
1	Return from NETXFR call (resume processing) immediately. The application program must monitor the status word in order to determine the progress of the file transfer.

dd — Format of the data that is to be transferred. This parameter can have the values:

Value	Description
0	The file contains binary data with embedded control words.
1	The file contains binary data in an unstructured format. (UU)+
2	The file contains binary data in a structured format. (US)+
3	The file contains 6-bit display code data with zero-byte records and physical boundaries. (C6)+
4	The file contains 7-bit ASCII code data in 12-bit bytes, packed five characters per central memory word. Zero-bytes terminate lines and physical boundaries may exist. (C8)+

NOTE

Coded files (values 3 and 4) are converted to a network file structure consisting of 7-bit ASCII code characters using unit-separator characters to terminate lines. This is the format of all coded files sent across the network. NAD code conversion is used, if available; otherwise NETXFR performs the code conversion using the host CPU.

tmout — An input parameter specifying the time in seconds to wait for a response before timing out. Where tmout is $1 \leq$ tmout ≤ 4095. The timer is reset after each response.

abl — Application block limit. This parameter is included for NAM AIP compatibility and is ignored by RHF.

+ Refer to the Remote Host Facility Usage manual for information concerning the DD parameter in the MFLINK or MFQUEUE commands.

Parameter	Description
facil	Facilities (data transfer options). This parameter is included for NAM AIP compatibility and is ignored by RHF.
bsize	Maximum size of the data block used during the data transfer phase. This parameter is included for NAM AIP compatibility and is ignored by RHF. Regardless of the value specified, RHF uses one of the following values based on the dd parameter:

Value	dd Parameter
2880 (binary mode)	0, 1, or 2
3840 (character mode)	3 or 4

ackw	Acknowledgment window. This parameter is included for NAM AIP compatibility and is ignored by RHF.
ickval	Initial checkmark value. This parameter is included for NAM AIP compatibility and is ignored by RHF.

CONTINUE FILE TRANSFER (NETXFRC)

If an application program has requested an immediate return from a NETXFR call, the application program must make periodic calls to FIP statements to allow the file transfer to continue. This can be done by calls to NETGET, NETGETL, NETPUT, NETWAIT, or NETXFRC. If the application program has no other need for calls to NETGET, NETGETL, NETPUT, or NETWAIT, it can call NETXFRC. Each NETXFRC call advances the file as much as possible and then immediately returns control to the calling application program. The application program should call NETXFRC only if at least one file transfer is in progress.

NETXFRC Calling Sequence

The format of the NETXFRC calling sequence is:

```
CALL NETXFRC
```

The NETXFRC call has no parameters.

FET I/O INTERFACE

The FIP statement NETUXFR initiates the FET I/O interface which enables an application program to transfer data to or from a remote application program. The FET I/O interface uses a file environment table (FET) and circular buffer similar to those used for operating system disk or tape input/output requests using CIO. This allows large amounts of data to be transferred very efficiently between two application programs and also allows standard techniques to be used for dealing with circular buffers and FET pointers.

This manual assumes you are familiar with CIO requests and FET I/O processing for sequential files. Refer to the NOS 2 Reference Set, Volume 4 and the NOS/BE Reference Manual for additional information concerning CIO and FET I/O operations.

DATA TRANSFER OPERATIONS

Although data transfer operations can be performed using either NETPUT/NETGET calls or NETUXFR calls, there are important operational differences. These differences are described in the following paragraphs.

- A NETPUT call is used to send data and a NETGET call is used to receive data. The NETUXFR call is used for both send and receive operations. The NETUXFR call includes a parameter that specifies a FET function code that indicates the operation to be performed and the direction of the data transfer. These FET function codes are a subset of the CIO function codes and include the following: READ, WRITE, WRITER, WRITEF, READCW, WRITECW, and CLOSE.

- A NETPUT call or NETGET call can transfer only one data block per call. A single NETUXFR call can transfer many data blocks. The data blocks are transferred to or from the circular buffer. If the application program monitors and updates the FET IN or OUT pointers while the operation is in progress, the amount of data transferred can exceed the length of the circular buffer. NETUXFR calls are always made without recall.

- Usually NETPUT or NETGET calls are used to send or receive acknowledged data blocks. However, they can transfer unacknowledged data blocks, if properly formatted (refer to appendix F for RHF restrictions that must be observed). The block size for acknowledged data blocks can vary up to the maximum size permitted by the application character type in effect.

  NETUXFR transfers only unacknowledged data blocks. These blocks have a fixed block size (except for blocks with an EOR, EOF, or EOI flag set). The fixed block size is 2880 8-bit bytes for binary mode and 3840 8-bit bytes for character mode. Refer to appendix E and appendix F for more information concerning network block formats.

- For NETPUT data transfer operations, the sending application program receives a block-delivered (FC/ACK/R) supervisory message when an acknowledged data block has been delivered. No significance is attached to the contents of the application block number field.

  For NETUXFR data transfer operations, RHF maintains a host abn field in the corresponding FET on both the sending and receiving sides of the logical connection. As the sending application's RHF removes a network data block from its circular buffer, RHF prefixes the data block with a network block header that includes the current host abn value from the FET. The sending RHF then increments the host abn field in the FET. On the receiving side, RHF compares the value of the abn in the network block header to the current host abn value in the receiving FET. If the abn values match, the receiving RHF transfers the data block (without the network block header) to the circular buffer and increments the host abn field in the FET. The sending and receiving application programs using NETUXFR must agree on the initial value for the abn field (usually zero) and also the transfer mode (binary or character). The transfer mode determines the maximum block size to be used.

- NETPUT/NETGET (act value of 1 or 2) and NETUXFR (binary mode) require packed or bit-string formatted data.

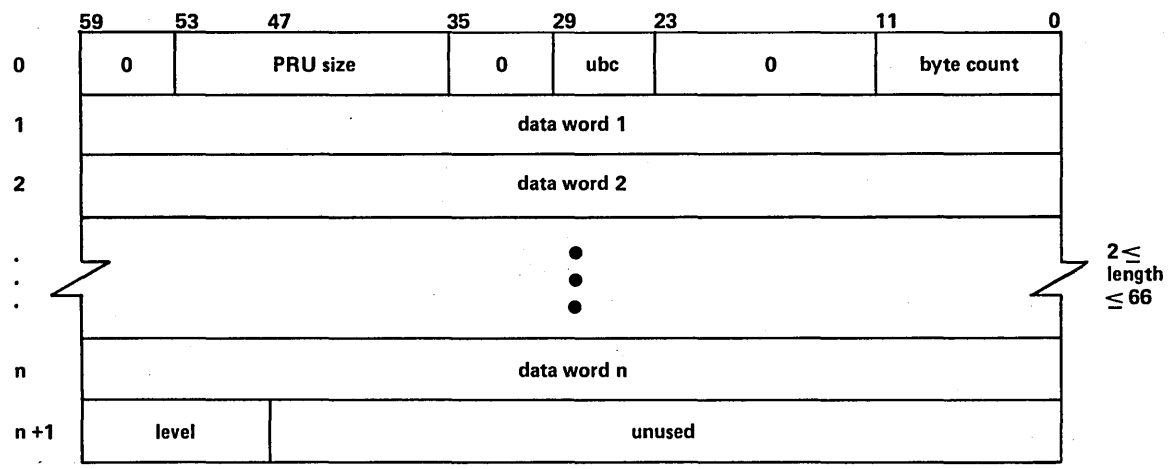
  NETPUT/NETGET (act value of 3) requires data presented as 8-bit characters, right-justified in 12-bit bytes. NETPUT/NETGET (act value of 4) requires data presented as 6-bit display code characters. The NAD performs character conversion to or from network ASCII characters by a one-to-one translation in the NAD hardware.

  NETUXFR (character mode) normally requires character (coded) data to be presented in 8/12 network ASCII format; that is, 8-bit characters, right-justified in 12-bit bytes with end-of-line boundaries denoted by ASCII unit-separator characters. As an installation option, either application program or both application programs can request RHF to enable NAD code conversion for a data transfer operation. However, an application program cannot enable code conversion for both directions of a given connection simultaneously. NAD code conversion translates coded data in either 6-bit display code format or 7-bit ASCII format with 12- to 66-bit end-of-line terminators to standard network ASCII format with unit-separator terminators.

NETUXFR DATA FORMATS

NETUXFR allows data to be represented in either NOS control word PRU format or NOS/BE control word PRU format. These formats are compatible with the corresponding operating system CIO disk control word PRU formats, except that RHF uses the ubc field to indicate the number of unused bits in a short PRU. The ubc field is defined but not supported for operating system disk files. Control word functions allow EORs and EOFs to be represented within the data in the circular buffer without terminating the read or write operation.

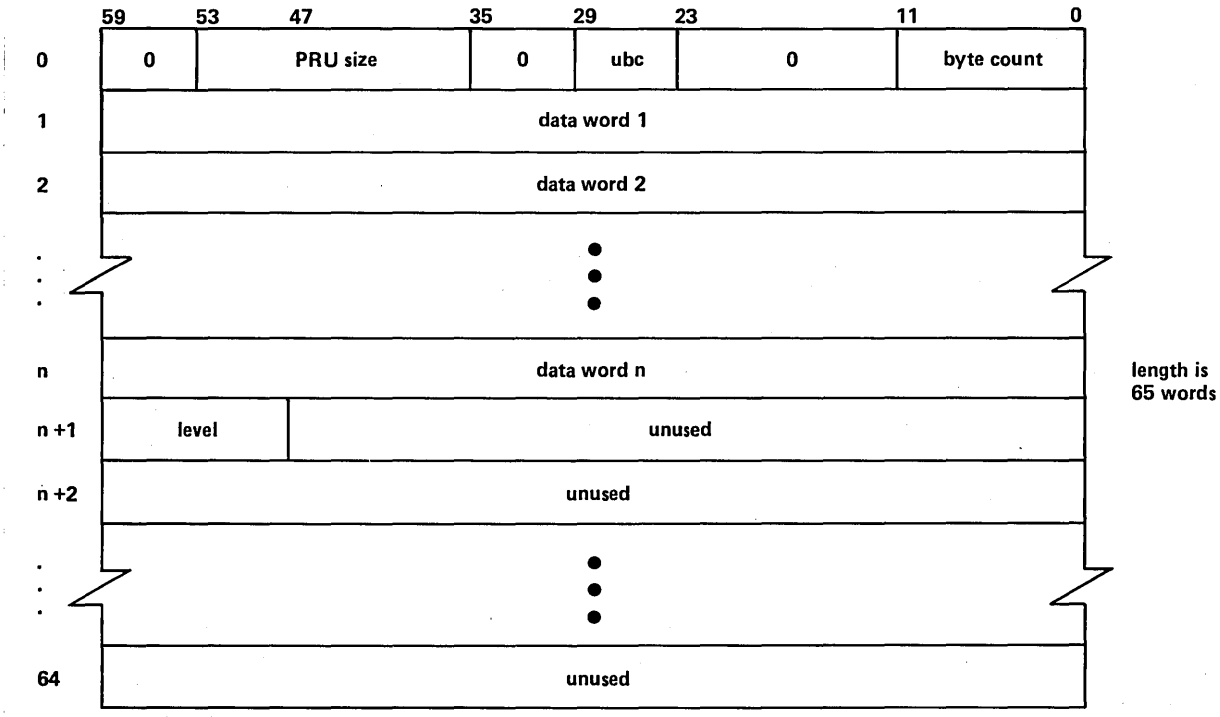
The format of the NOS control word PRU format is:



Field	Description
PRU size	100_8 central memory words.
ubc	Unused bit count ($0 \leq ubc \leq 11$).

Field	Description
byte count	Number of 12-bit data bytes in the PRU ($0 \leq$ byte count $\leq 500_8$).
level	Logical record level number (if short PRU).
	$0\text{-}16_8$ The PRU is an EOR. 17_8 The PRU is an EOF.

The format of the NOS/BE control word PRU format is:



Field	Description
PRU size	100_8 central memory words.
ubc	Unused bit count ($0 \leq$ ubc ≤ 59).
byte count	Number of 12-bit data bytes in the PRU ($0 \leq$ byte count $\leq 500_8$).
level	Logical record level number (if short PRU).
	$0\text{-}16_8$ The PRU is an EOR. 17_8 The PRU is an EOF.

NETUXFR FUNCTION REQUIREMENTS

When issuing NETUXFR functions, the application program must ensure that sufficient space is available in the buffer for read functions and sufficient data is available for write requests.

Table 4-1 shows the minimum buffer space requirements for the various receive functions and data formats.

Table 4-1. Receive Function Requirements

Function	Data Format	Minimum available buffer space in CM words	
		NOS	NOS/BE
READ	binary	384	384
	coded	768	768
	convert mode coded	384	384
READCW	binary	398	455
	coded	794	845
	convert mode coded	398	455

Table 4-2 shows the minimum data requirements for the various send functions and data formats.

Table 4-2. Send Function Requirements

Function	Data Format	Minimum available data in CM words	
		NOS	NOS/BE
WRITE	binary	384	384
	coded	768	768
	convert mode coded	384	384
WRITER	all	none	none
WRITEF	all	none	none
WRITECW	binary	396+	390+
	coded	792+	780+
	convert mode coded	396+	390+
CLOSE	all	N/A	N/A

+ These values assume all control words denote full PRUs. If the first PRU is short, the minimum available data in CM words is 2 for NOS; and 65 for NOS/BE.

NETUXFR Calling Sequence

The format of the NETUXFR calling sequence is:

```
CALL NETUXFR (acn,fetadr,reqarea,fetfc,conmode)
```

Parameter	Description
acn	An input parameter specifying the application connection number of the logical connection on which the data is to be transferred. Where acn is minacn $\leq$ acn $\leq$ maxacn.
fetadr	An input parameter specifying the address of an RHF FET describing where data is to be transferred to or from.
reqarea	An input parameter specifying the address of a three-word block that FIP uses as the RHF request block.
fetfc	An input parameter specifying the FET function code to be performed by this request. These octal codes are a subset of the CIO function codes.

Code (Octal)	Function	Direction	Description
010	READ	Receive	Read data until EOR, EOI, or buffer full.
014	WRITE	Send	Write data until less than full block remains.
024	WRITER	Send	Write data; terminate with EOR.
034	WRITEF	Send	Write data; terminate with EOF (level 17_8 EOR).
150	CLOSE	Send	Write zero-length EOI block.
200	READCW	Receive	Read data with host control words until EOI or buffer full.
204	WRITECW	Send	Write data with host control words.

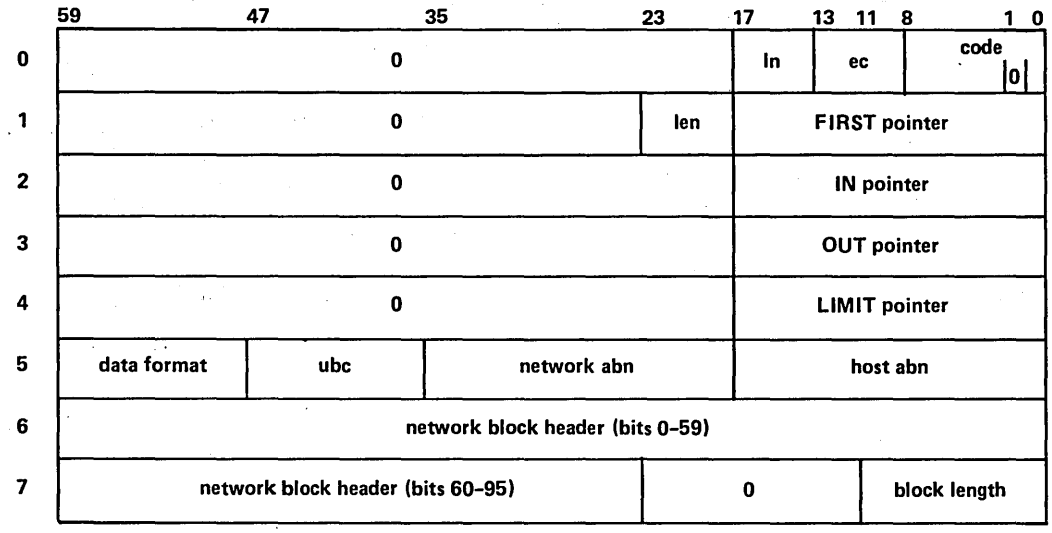
Parameter	Description

conmode — An input parameter specifying the code conversion mode in use. This value must match the convert mode selected with the CM/SEL/R supervisory message. This parameter can have the values:

Value	Direction	Description
0	Send/Receive	No conversion mode is in use.
1	Send	Convert 7-bit ASCII in 12-bit bytes to network ASCII.
2	Send	Convert 6-bit display code to network ASCII.
3	Receive	Convert network ASCII to 7-bit ASCII in 12-bit bytes.
4	Receive	Convert network ASCII to 6-bit display code.

RHF FET Format

The format of the RHF FET is:



Field	Description
Level number (ln)+	Level number for EOR and EOF.
Error code (ec)	Status information returned by RHF when an error or termination condition occurs. This field can have one of the following values:

Value	Description
0	Successful completion.
1	End of information.
2	FET parameter error.
3	Next input block is an acknowledged data block; use NETGET to receive the block.
4	Host abn does not match the abn for the next input block.
5	Last block exceeded the allowable block size (excess is discarded).
6	Control word PRU format error.
7	No buffers are available for data output; reissue the request.
8	Data length does not match the text length specified in the network block header (the smaller of the two data lengths is used).
9	Partial block without EOR or EOI.
10	Network abn miscompare on an input data block.
11	Data block is not a multiple of CM words.
12	Unknown block error.

code+	Code and status similar to CIO FET (bit 1 must be zero).
FET length (len)+	Specifies the number of extra words in the FET beyond the normal size of five words.
FIRST pointer+	First word address of circular buffer.
IN pointer+	Next available location to enter data into the circular buffer.

+ These fields are the same as those described for the CIO FETs in the NOS 2 Reference Set, Volume 4 and NOS/BE Reference Manual.

Field	Description
OUT pointer+	Next available location to remove data from the circular buffer.
LIMIT pointer+	Last word address plus 1 of the circular buffer.
data format	Specifies the data format selected by the application program for the network data block. This field can have one of the following values:

Value	Description
1	Binary format, 2880 8-bit bytes.
3	Character format, 3840 8-bit bytes.

ubc	Unused bit count. Specifies the number of unused bits in a short PRU. An application program sets this field for WRITER and WRITEF functions; RHF sets this field for READ functions.
network abn	RHF returns the network abn value when a block error occurs during a conversion mode data transfer operation.
host abn	The host abn is initialized by an application program and incremented by RHF as data is removed from or transferred to the circular buffer.
network block header	RHF returns the network block header when a block error occurs during a conversion mode data transfer operation (refer to appendix E for a description of the network block header fields).
block length	Number of 12-bit bytes occupied by the data block. RHF returns this field when a block error occurs.

DEBUGGING CONTROL STATEMENT

The FIP statement NETDBG enables an application program to record all messages exchanged between the program and the network. These messages are logged into the application program's dayfile so they can be used for debugging.

LOGGING MESSAGES (NETDBG)

The NETDBG statement allows an application program to control logging of supervisory messages, data messages, or file transfer messages into the application program's dayfile.

+ These fields are the same as those described for the CIO FETs in the NOS 2 Reference Set, Volume 4 and NOS/BE Reference Manual.

NETDBG Calling Sequence

The format of the NETDBG calling sequence is:

```
CALL NETDBG (dbugsup,dbugdat,dbugstat,dbugxfr)
```

Parameter	Description
dbugsup	An input parameter that turns the logging of supervisory messages on or off. This parameter can have the values:

Value	Description
0	Turn supervisory message logging on.
Nonzero	Turn supervisory message logging off.

When supervisory message logging is turned on, all supervisory messages (except block-delivered messages) exchanged on logical connection 0 between the application program and RHF are logged into the application program's dayfile. Logging occurs whenever a call to NETGET, NETGETL, or NETPUT causes a message transfer. Logging continues until a call with a nonzero dbugsup parameter is issued.

The format of the dayfile message for both supervisory messages and data messages is:

```
FIP  aaaaaa  bbbb  MHA  hhhhhhhhhhhhhhhhhhhh
FIP  aaaaaa  bbbb  MTA  ttttttttttttttttttt
tttt...                              ...tttt
                      .
                      .
                      .
tttt....                             ...tttt
```

Where:
- aaaaaa is NETPUT or NETGET.
- bbbb is SUPR or DATA.
- hhh...h is the header word in 6-bit display code.
- ttt...t is the first nine words of the text area of the message in 6-bit display code.

The total length of the dayfile message is six central memory words.

Parameter	Description

dbugdat — An input parameter that turns the logging of data messages on or off. This parameter can have the values:

Value	Description
0	Turn data message logging on.
Nonzero	Turn data message logging off.

When data message logging is turned on, the first 10 central memory words of all data messages exchanged on any logical connection between the application program and RHF are logged into the application program's dayfile. Block-delivered supervisory messages are also logged, regardless of the setting of the dbugsup parameter. Logging occurs whenever a call to NETGET, NETGETL, or NETPUT causes a message transfer. Logging continues until a call with a nonzero dbugdat parameter is issued. Note that when dbugdat is on, dbugxfr is also on.

dbugstat — A return parameter, included for AIP compatibility. This parameter always has a value of 0.

dbugxfr — An input parameter that turns the logging of file transfer messages on or off. This parameter can have the values:

Value	Description
0	Turn file transfer message logging on.
Nonzero	Turn file transfer message logging off.

When file transfer message logging is turned on, certain messages giving the status of the file transfer will be logged into the application program's dayfile. Logging continues, when file transfers are active, until a call with a nonzero dbugxfr parameter is issued.

INTERFACE DESCRIPTIONS 5

This section describes the language interface requirements of an application program and the interfacing utilities available to an application program.

Application program use of FIP is essentially the same, regardless of the language used to code the application program. Parameter list and calling sequence requirements are the same for COMPASS assembly language and compiler-level languages. However, the residence of the FIP routines, the form of the calling sequences, and the utilities available to the application program differ for COMPASS and compiler-level languages.

PARAMETER LIST AND CALLING SEQUENCE REQUIREMENTS

The FIP statements and interfacing utilities use standard FORTRAN-style calling sequences and parameter lists; that is, a parameter list contains one 60-bit word per parameter. The address of this parameter list is passed to the appropriate routine in register A1. Linkage with the statement within the application program is performed by executing a return jump instruction (RJ) to the entry point. To provide compact object code, traceback information is not generated, and the parameter list need not be followed by a word of zeros.

Because the statement parameters are passed by address (called by reference), the RHF application programmer should be careful about substituting values when defining the parameters. Those parameters identified as return parameters should not be specified as constants or expressions in the call statement. Such specifications can produce unpredictable errors in program code. This restriction is compatible with normal FORTRAN programming practices.

Return parameters are normally defined by variable names, array names, array element names, or similar symbolic addresses; although the terminology for such entities varies according to the programming language used, this manual uses the term symbolic address. Unless otherwise stated, numeric absolute or relative addresses are not used in call statements.

Those parameters identified as input parameters can be defined by constants, expressions that can be evaluated to produce constants, or symbolic addresses (as defined above). Input parameters are usually defined by constants or expressions for which this manual uses the term value.

All FIP statement parameters used by a COBOL program must be described in the Data Division as level 01 data entries, or data entries at other levels when the entries are left-justified to word boundaries. COBOL 5 programs that access fields within parameters must also describe the fields in the Data Division as COMP-4 numeric data entries in order to manipulate values within the fields as 6-bit entities. Direct field access and FIP use is difficult using COBOL; COMPASS or FORTRAN subroutines are sometimes necessary to set up parameters before FIP calls or unpack parameters after FIP calls.

All direct calls from a COBOL program to FIP must be coded as calls to FORTRAN subroutines.

The RHF calling sequence does not permit recursive calls.

PREDEFINED SYMBOLIC NAMES

The fields in RHF supervisory messages and supervisory and data message headers have been assigned symbolic names so that they can be identified to the macros and subroutines described later in this section. These names are display-coded characters and are listed and defined in appendix C.

PREDEFINED SYMBOL VALUES

Some of the fields with predefined symbolic names have predefined values that can be obtained through the macros and subroutines described later in this section. The predefined values for the predefined symbolic names are given in appendix C, where the names are listed alphabetically.

Care should be taken in using symbolic names with predefined values; in some instances, a symbolic name and its corresponding value have been assigned to a group of fields. Because choosing the wrong name can fill more fields than intended, the RHF application programmer should become familiar with all of the predefined symbolic names before using them.

COMPASS ASSEMBLY LANGUAGE

Application programs coded in COMPASS can use macros that call FIP procedures. These macros have the same names as the FIP statements directly referenced by a FORTRAN program, as described in section 4.

Packing and unpacking supervisory message blocks in a COMPASS program is easily accomplished using the field access macros NFETCH and NSTORE.

COMPASS macros as well as predefined symbolic names and symbol values may be obtained during assembly from system text overlay FIPTEXT.

FIP MACRO CALL FORMATS

For those FIP statement calls with parameters, three forms of the COMPASS macro call are possible:

Location	Operation	Variable
label+	macro-name	parameters

This is the standard call, producing the full calling sequence.

Location	Operation	Variable
label+	macro-name	LIST=label or register name

When this form is used, macro expansion assumes the proper calling parameter block is located at the specified address, loads this address into register A1, and performs the call to the FIP procedure.

+ Label is optional for this macro call.

Location	Operation	Variable
label	macro-name	parameters, LIST

When this form is used, macro expansion produces a parameter block in place but does not generate the call to the FIP procedure; the address (label) of the statement using this form is the address (LIST=label) used in the second form.

The first form should be used when making a straightforward call to the FIP procedures. The second form can be used once the parameter list has been created elsewhere with the third form, and might save space when procedures are used several times. For example,

```
          NETPUT    IHA,ITA
```

is a direct call to execute the NETPUT macro with the two symbolic address parameters shown;

```
PUT1      NETPUT    IHA,ITA,LIST
```

causes expansion of the NETPUT macro and creation of the indicated parameter list at symbolic address PUT1, but does not execute NETPUT;

```
          NETPUT    LIST=PUT1
```

actually executes the NETPUT macro with the parameters in the list expanded at location PUT1.

If a COMPASS macro call contains an error, COMPASS flags the error and provides an explanation.

A summary of the available macro call formats appears in appendix D.

FIELD ACCESS MACROS

Two additional macros, NFETCH and NSTORE, are provided to make message field definition and access easier. Application programmers are urged to use these macros as described below. Use of these macros and their related predefined symbolic names will simplify application program conversion under future versions of RHF.

NFETCH Macro

A call to the NFETCH macro returns the contents of a specific field within an array of one or more words that comprise all or part of a supervisory message block or supervisory or data message header. The octal integer value returned by the call is right-justified within the X or B register specified in the call.

The format of the NFETCH macro call is:

Location	Operation	Variable
label	NFETCH	array,field,Xj or Bj

Parameter	Description
label	Optional address label of the macro call.
array	The address of the first word of the array from which the field value should be obtained. This parameter can be an address label, the name of a register address, or 0. If 0 is declared, any predefined value for the indicated symbolic name is returned.
field	The predefined symbolic name of the field for which a value should be fetched from the array. The possible contents of field are listed alphabetically in appendix C.
j	The number of the X or B register that should receive the value fetched from the array. The value is right-justified in Xj or Bj on return from the call. When a B register is used, the field to be fetched must be less than 18 bits long.

Execution of NFETCH destroys the contents of registers A5, A6, X5, X6, X7, and the X or B register specified to receive the returned value.

As examples of NFETCH use, consider the following operations.

The statement

```
NFETCH      MYARRAY,PFC,X1
```

places the value of the primary function code field within MYARRAY into register X1. The primary function code field is identified by the symbolic name PFC.

The statements

```
SX2         BUFFER
NFETCH      X2,SFC,X3
```

place the value of the secondary function code field within BUFFER into register X3. The secondary function code field is identified by the symbolic name SFC, and the address label BUFFER is supplied through register X2.

The statements

```
NFETCH      ARRAY,EB,X3
NZ          X3,ERROR
```

place the value of the error bit (EB) field within ARRAY into register X3. If the value in X3 is nonzero (if EB has a value of 1), a jump to ERROR occurs.

The statement

```
NFETCH      0,CON,X1
```

places the predefined value 63_{16} into register X1. The value returned is that of the primary function code field of all connection-request supervisory messages, as identified by the predefined symbolic name CON.

If an NFETCH macro call contains an error, COMPASS flags the error and provides an explanation.

NSTORE Macro

A call to the NSTORE macro sets the contents of a specific field within an array of one or more words that comprise all or part of a supervisory message block or supervisory or data message header.

The format of the NSTORE macro call is:

Location	Operation	Variable
label	NSTORE	array,field=value

Parameter	Description
label	Optional address label of the macro call.
array	The address of the first word of the array into which the field value should be placed. This parameter can be declared as an address label or the name of an address register.
field	The predefined symbolic name of the field for which a value should be stored in the array. The possible contents of field are listed alphabetically in appendix C.
value	The value to be stored in the identified field within the arrary. This parameter can be: • A right-justified integer. • A right-justified, zero-filled character string. • A symbolic name with a predefined value (refer to table C-1). • Bj or Xj, where j is the number of an X or B register containing either a right-justified integer or a right-justified, zero-filled character string.

Execution of NSTORE destroys the contents of registers A5, A6, X5, X6, X7, and any X or B register specified in the call.

As examples of NSTORE use, consider the following operations.

The statements

```
SX2         MYARRAY
NSTORE      X2,PFC=CTRL
```

store the predefined value for CTRL in the primary function code field of MYARRAY. The primary function code field is identified by the symbolic name PFC, and the address label MYARRAY is obtained through register X2.

The statement

```
NSTORE      MYARRAY,PFC=CTRL
```

performs the same operation.

The statement

```
NSTORE  MYARRAY,CONANM=MYAPPL
```

stores the application program name MYAPPL in the requested application program name field of MYARRAY. The requested application program name field is identified by the predefined symbolic name CONANM.

If an NSTORE macro call contains an error, COMPASS flags the error and provides an explanation.

COMPILER-LEVEL LANGUAGES

Application programs coded in compiler-level languages such as FORTRAN use FIP statements that make relocatable subroutine calls. Entry point references are satisfied by the CYBER loader; the FIP routines are loaded from the system library LCNLIB, which must be declared in an LDSET or LIBRARY command.

Packing and unpacking supervisory message blocks in a compiler-level program is easily accomplished using the field access subroutines NFETCH and NSTORE. These subroutines reside in system library LCNLIB.

FIP SUBROUTINE CALL FORMAT

Only one form of the FIP subroutine call is possible in compiler-level language programs. This form is:

```
subroutine-name(parameters)
```

The syntax of this form is described in section 4 under Syntax of FIP Statements. A summary of the available call formats appears in appendix D. The FORTRAN form of the subroutine call format is used throughout this manual when discussing the FIP routines.

FIELD ACCESS SUBROUTINES

Two additional relocatable subroutines, NFETCH and NSTORE, are provided to make message field definition and access easier. Use of these routines and their related predefined symbolic names will simplify application program conversion under future versions of RHF. Because each call to one of these routines causes a table scan, use of the routines increases program execution time. This increase can be minimized by setting up all constants processed by calls to the routines with a single set of calls at the beginning of the program.

NFETCH Subroutine

A call to the NFETCH subroutine returns an integer value of the contents of a specific field within an array of one or more words that comprise all or part of a supervisory message block or supervisory or data message header. NFETCH can be used anywhere in a program expression that an operand can be used.

The format of the NFETCH subroutine call is:

```
ivalue=NFETCH(array,field)
```

Parameter	Description
ivalue	A return parameter; ivalue is an optional variable used to receive the integer value returned.
array	An input parameter specifying the predefined symbolic address of the first word of the array from which the field value can be obtained. This parameter can be the array name or 0. If 0 is declared, any predefined value for the indicated symbolic name is returned.
field	An input parameter specifying the predefined symbolic name of the field for which a value should be fetched from the array. This parameter must be left-justified with zero fill. The possible contents of field are listed alphabetically in appendix C.

The size of the field involved in the NFETCH subroutine call determines the sign of the integer value returned. If the field is 1 bit long and contains 0, the integer value is positive; if the field is 1 bit long and contains 1, the integer value is negative.

If either the array or field parameter is omitted from the NFETCH subroutine call, the application program is aborted and a dayfile message is issued.

As examples of NFETCH use, consider the following operations.

The statement

```
M=NFETCH(ARRAY,L"EB")
```

makes M positive if the error bit field EB within ARRAY is 0, and makes M negative if the error bit field is 1. The error bit is identified by the predefined symbolic name EB, left-justified with zero fill in the call.

The statement

```
M=NFETCH(0,L"CON")
```

makes M the integer value 63_{16}, equivalent to the predefined value for the primary function code field in connection-request supervisory messages. The primary function code field is identified by the predefined symbolic name CON, left-justified with zero fill in the call.

The statement

```
IF(NFETCH(ARRAY,L"EB").NE.-0) CALL ERROR
```

causes a jump to ERROR if the value of the error bit field EB within ARRAY is 1. Because EB is a 1-bit field, the sign of the returned integer must be tested, not the value of the integer. A field value of 0 produces a positive integer; a field value of 1 produces a negative integer.

NSTORE Subroutine

A call to the NSTORE subroutine stores the contents of a specific field within an array of one or more words that comprise all or part of a supervisory message block or supervisory or data message header.

The format of the NSTORE subroutine call is:

```
CALL NSTORE(array,field,value)
```

Parameter	Description
array	A return parameter; as input to the call, the symbolic address of the first word of the array into which the field value should be placed. This parameter is normally the array name.
field	An input parameter specifying the predefined symbolic name of the field for which a value should be stored in the array. This parameter must be left-justified with zero fill. The possible contents of field are listed alphabetically in appendix C.
value	An input parameter specifying the value to be stored in the identified field within the array. This parameter can be: • A right-justified integer value. • A right-justified, zero-filled Hollerith character string. • A left-justified, zero-filled symbolic name with a predefined value (refer to table C-1).

Integer values stored by the NSTORE subroutine call are stored as integers. Character strings are stored in 6-bit display code form, and symbolic names are converted to octal equivalents of their predefined values when stored. Only one field can be specified in each call. A value can be stored in a field anytime after the array is declared.

If either the array, field, or value parameters are not declared or are nonexistent, the application program is aborted and a dayfile message is issued.

As examples of NSTORE use, consider the following operations.

The statement

```
CALL NSTORE(ARRAY,L"PFC",L"CON")
```

stores the predefined value for the primary function code of all connection-request supervisory messages in the primary function code field of ARRAY. The primary function code value is identified by the predefined symbolic name CON, and the primary function code field is identified by the predefined symbolic name PFC; both symbolic names are left-justified with zero fill in the call.

The statement

```
CALL NSTORE(ARRAY,L"CONANM",R"MYAPPL")
```

stores the 6-bit display code application program name MYAPPL in the requested application program name field of ARRAY. The requested application program name field is identified by the predefined symbolic name CONANM, left-justified with zero fill in the call.

The statement

```
CALL NSTORE(ARRAY,L"RB",1)
```

stores a 1 in the response bit field RB of ARRAY. The response bit field is identified by the predefined symbolic name RB, left-justified with zero fill in the call.

DIAGNOSTIC MESSAGES A

This appendix contains diagnostic messages issued by routines documented in this manual. Messages are listed alphabetically. Lowercase letters in a message indicate a variable field; such fields are explained in the accompanying message description.

If you encounter a diagnostic or informative message that does not appear in this appendix, consult the NOS 2 or NOS/BE Diagnostic Index. These publications catalog all messages produced by NOS 2 or NOS/BE and their products and specify the manual or manuals in which each message is fully documented.

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
FIP - ABN MISCOMPARE ON filename.	A data transfer error has occurred. The system has halted the file transfer. filename The affected file.	Inform site analyst.	FIP
FIP - ACN acn NOT WITHIN RANGE.	A data transfer error has occurred. The system has halted the file transfer. acn Application connection number (octal).	Inform site analyst.	FIP
FIP - BLK NOT 60 BIT MULT ON filename.	Data blocks being transferred by RHF on convert mode paths must be multiples of 60 bits. filename The affected file.	Inform site analyst.	FIP
FIP - BLOCK LGTH MISMATCH ON filename.	RHF received a data block where the block length given in the block header was greater than the actual block length. filename The affected file.	Inform site analyst.	FIP
FIP - CIO ERROR code.	The system detected an error in an input/output request involving a local data file. The system has halted the file transfer. code CIO error code (octal). Refer to Volume 4 of the NOS 2 Reference Set, or the NOS/BE Reference Manual, for a description of CIO error codes.	Inform site analyst.	FIP
FIP - CONNECTION BROKEN ON ACN acn.	The network connection has been broken unexpectedly. The system has halted the file transfer. acn Application connection number (octal).	Rerun your job. Inform site analyst.	FIP
FIP - CONVERT MODE N/A FOR filename.	The application timed out waiting for resources to become available. filename The affected file.	Try again. If problem persists, inform site analyst.	FIP
FIP - CTRL WORD FORMAT ERROR ON filename.	The data block being sent to or being received from the network has faulty internal control information. The system has halted the file transfer. filename The affected file.	Inform site analyst.	FIP
FIP - DISABLE WARNING RECEIVED.	The system has halted the file transfer because the network is shutting down immediately.	Retry the file transfer after the network is reactivated.	FIP
FIP - GT 4 FILE TRANSFERS INITIATED.	The system is attempting too many file transfers simultaneously. The system has not initiated the file transfer you are requesting.	Inform site analyst.	FIP
FIP - filename HAD AN RHF I/O ERROR.	The file transfer was terminated because an I/O error was detected by RHF. See dayfile for further information. filename The affected file.	Inform site analyst.	FIP
FIP - filename HAD RHF FET PARAMETER ERR.	The system detected an error in transferring data to or from the network. The system has halted the file transfer. filename File being transferred.	Inform site analyst.	FIP
FIP - HOST ABN MISCOMPARE ON filename.	The local RHF sent or received a block that had an application block number (ABN) out of sequence. filename The affected file.	Inform site analyst.	FIP
FIP - INITIATING XFR OF filename.	The system has initiated the transfer of file filename. filename File being transferred.	None.	FIP
FIP - INVALID PARTIAL BLK ON filename.	The local RHF sent or received a short block that did not have an EOR, EOF, or EOI mark. filename The affected file.	Inform site analyst.	FIP
FIP - filename IS AN EMPTY FILE.	The system attempted to transfer an empty file. The system has halted the file transfer. filename File being transferred.	Ensure that the file filename is not empty and rerun job. Inform site analyst if the problem persists.	FIP

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
FIP - LAST BLOCK TOO BIG ON filename.	The network block being transferred is too large. The system has halted the file transfer. filename File being transferred.	Inform site analyst.	FIP
FIP - LOGIC ERROR IN routine.	The system detected an error in the specified routine.	Inform site analyst.	FIP
FIP - NET ABN MISCOMPARE ON filename.	The local NAD received a block with an application block number (ABN) out of sequence. filename The affected file.	Inform site analyst	FIP
FIP - NETOFF DURING FILE TRANSFER.	An internal error occurred during your file transfer. The file transfer was not completed successfully.	Inform site analyst.	FIP
FIP - filename ON INVALID DEVICE.	You have file filename assigned to an inaccessible device. The system has halted the file transfer. filename File being transferred.	Reassign file filename to an accessible device and rerun job. Inform site analyst if the problem persists.	FIP
FIP - OUTPUT BLOCK NOT DEL ON ACN acn.	The remote system did not receive the network message or data block before the time-out period elapsed. The system has halted the file transfer. acn Application connection number (octal).	Retry the file transfer. Inform site analyst if the problem persists.	FIP
FIP - PREMATURE TERMINATION RCVD ON acn.	The system detected an error during a file transfer and halted the file transfer. acn Application connection number (octal).	Retry file transfer. Inform site analyst if the problem persists.	FIP
FIP - PROTOCOL ERROR DETECTED.	An unrecognized or unexpected network message has been received. The file transfer is ended.	Retry file transfer. Inform system analyst if problem persists.	FIP
FIP - RHF FET PARAM. ERR ON filename.	RHF detected an error in the FET specified by the calling application. filename The affected file.	Inform site analyst.	FIP
FIP - SECOND FILE XFR ON ACN acn.	The system attempted a file transfer on a connection that already has a file transfer in progress. The system halted the second file transfer. acn Application connection number (octal).	Inform site analyst.	FIP
FIP - TIMED OUT WAITING FOR NETWORK.	The network failed to respond before the time-out period elapsed. The system halted the file transfer.	Ensure that the network and remote system are active and retry the file transfer. Inform site analyst if the problem persists.	FIP
FIP - TRANSFER OF filename COMPLETE.	Self-explanatory. filename File being transferred.	None.	FIP
FIP - TRANSFER OF filename IN PROGRESS.	Self-explanatory. filename File being transferred.	None.	FIP
FIP - UNDEFINED ERROR.	The system encountered an unexpected error.	Inform site analyst.	FIP
FIP - UNKNOWN BLOCK ERROR ON filename.	The local RHF was unable to successfully send or receive a block to the local NAD. The reason for the rejected block is unknown. filename The affected file.	Inform site analyst.	FIP
FIP - XFR TERM WITH ERR, IDLEDOWN.	The network is shutting down and your file transfer ended unsuccessfully.	Retry the file transfer when the network becomes active.	FIP
MHF, ABORTED	Fatal Error. Consult the MHF job dayfile to determine the error.	Correct the problem and enable MHF in the RHF application display.	MHF
MHF, BUFFER FOR NLD TOO SMALL, LOCAL NAD CH=xx	A fault in MHF prevents automatic loading or dumping of the local NAD on channel xx.	Correct MHF and restart RHF.	MHF
MHF, CONNECT BROKEN, RNAD yy, LOCAL NAD CH=xx	The connection between the local NAD on channel xx and remote NAD yy (hexadecimal) was broken because of a network or remote NAD failure.	None.	MHF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
MHF, CONNECT REJECT, RNAD yy, LOCAL NAD CH=xx	The connection between the local NAD on channel xx and remote NAD yy (hexadecimal) was rejected because of a network or remote NAD failure.	None.	MHF
MHF, DEVICE ENABLE SWITCH OFF, LOCAL NAD CH=xx	The device enable switch of the local NAD on channel xx is turned off, disabling the NAD.	Turn on device enable switch.	MHF
MHF, ERROR IN ROUTINE xxxxxxx	Fatal error. Fault in MHF routine xxxxxxx.	Correct MHF and restart RHF.	MHF
MHF, EST ENTRY UP OR NOT RESERVED, NAD CH=xx	A CVL error response indicates that the EST entry of the local NAD on channel xx is not in the proper state for NAD memory dumping or controlware loading.	Inform site analyst.	MHF
MHF, INIT PARAMETER BAD IN RECORD xxxxxxx.	MHF found a faulty memory-size or system-buffer-count value in the controlware initialization parameter record xxxxxxx on the RHF configuration file. The NAD is not loaded.	Correct the faulty initialization parameter record on the RHF configuration file.	MHF
MHF, INIT PARAMETER RECORD MISSING xxxxxxx	MHF did not find record xxxxxxx on the RHF configuration file. A subsequent message will indicate that the load succeeded with default parameters.	Correct RHF configuration file.	MHF
MHF, INVALID NLD RESPONSE yyB, LOCAL NAD CH=xx	MHF received an unexpected response code (yy) when trying to communicate with the local NAD on channel xx.	Inform site analyst.	MHF
MHF, INVALID STATUS IN EST ENTRY, NAD CH=xx	An error response from NLD indicates that the EST entry of the local NAD on channel xx is not in the proper state for NAD memory dumping or controlware loading.	Inform site analyst.	MHF
MHF, NAD ACCESS ERROR (NLD ERROR 4) NAD CH=xx	MHF could not dump or load the local NAD on channel xx because of a fault detected by NLD. MHF will retry the dump or load.	Inform site analyst if message persists.	MHF
MHF, NAD C/W FAULTY - xxxxxxx.	MHF found a fault in controlware record xxxxxxx on the system: no 77-table, 52-table, or incorrect length. The NAD is not loaded.	Correct the controlware record and restart RHF.	MHF
MHF, NAD C/W MISSING - xxxxxxx.	MHF did not find controlware record xxxxxxx on the system when attempting to load a NAD.	Add controlware record to system and restart RHF.	MHF
MHF, NAD DUMP FILE BAD - xxxxxxx	MHF could not automatically dump NAD memory because permanent file xxxxxxx was faulty.	Correct (rename or purge) the permanent file, and turn on NAD in EST.	MHF
MHF, NAD ON CHANNEL xx DISABLED	MHF has abandoned further attempts to dump or load the local NAD on channel xx. Consult the preceding dayfile messages to determine the error.	Correct the problem and restart RHF.	MHF
MHF, NAD ON CHANNEL xx DUMPED FILE yyyyyyy RECORD zz	The memory of the local NAD on channel xx has been copied to record zz (decimal) of permanent file yyyyyy. (Use DMPNAD to list the dump record.)	None.	MHF
MHF, NAD ON CHANNEL xx LOADED	MHF has loaded controlware into the local NAD on channel xx.	None.	MHF
MHF, NAD PROCESSOR STOPPED (DC=yyyy), NAD CH=xx	The local NAD on channel xx stopped unexpectedly, with dead code yyyy (hexadecimal).	None.	MHF
MHF, NDT NOT AVAILABLE FROM RHF.	Fatal error. MHF could not obtain the Network Description Table from RHF.	Correct MHF or RHF and restart RHF.	MHF
MHF, NETON REJECT - reason	Fatal error. MHF failed to connect (NETON) with RHF for one of the following reasons:		MHF
	RHF UNAVAILABLE	Start RHF. Inform site analyst if message persists.	
	MHF ALREADY ACTIVE	Inform site analyst if no other copy of MHF is active.	
	MHF DISABLED	Enable MHF in RHF's tables.	

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
	MHF UNKNOWN TO RHF	Add MHF to the RHF configuration file and restart RHF.	
	MHF UNAUTHORIZED	Inform site analyst if no other copy of MHF is active.	
	INVALID VALUE FOR ACN	Correct MHF.	
	MHF NETTED ON ALREADY	Correct MHF.	
	UNRECOGNIZED ERROR	Correct MHF or RHF.	
MHF, NO EST ENTRY DEFINED, LOCAL NAD CH=xx	MHF found no EST entry for a local NAD on channel xx.	Correct RHF configuration file or EST.	MHF
MHF, STARTED	Informative message.	None.	MHF
MHF, STOPPED	Informative message.	None.	MHF
MHF, WAIT FOR NAD DUMP FILE - xxxxxxx	MHF is waiting for exclusive access to file xxxxxxx in order to dump NAD memory.	Inform site analyst if the message persists.	MHF
MHF, WAITING FOR ACCESS TO LOCAL NAD, Ch=xx	MHF is waiting for maintenance access to the local NAD on channel xx.	Inform site analyst if the message persists.	MHF
NDR - ADDRESS ERROR.	The calling program specified an invalid address.	Inform site analyst.	NDR
NDR - ILLEGAL CALLER.	Only RHF is allowed to call NDR.	Inform site analyst.	NDR
NDR - ILLEGAL FUNCTION.	RHF specified an invalid function.	Inform site analyst.	NDR
NDR - INVALID CONNECT REQUEST.	A NAD sent an invalid reply to a system connect request.	Inform site analyst.	NDR
RHF, jobn ACNacn ACCOUNTING OVERFLOW-ADD 32767 TO message	Application connection number acn of the application with job name or job sequence name jobn has had an accounting overflow. 32767 must be added to the appropriate field of the accounting message that is issued when a connection is terminated. message is one of the following: - BLOCKS SENT - BLOCKS RECEIVED - ACKS SENT - ACKS RECEIVED This message is issued every time an overflow occurs.	None.	RHF
RHF, jobn ACNacn DISCONNECT BLOCKS SENT=bssss RECEIVED=brrrr RHF, jobn ACKS SENT=kssss, ACKS RECD= krrrr, PATH ID=id, CH=cc	The application with job name or job sequence name jobn terminated its application connection number acn. The connection had path-id id and used the NAD on channel cc. acn Application connection number in octal. bssss Number of blocks sent by the application to RHF for transport across the LCN. brrrr Number of blocks received by the application from RHF. kssss Number of acknowledgements sent by RHF for blocks sent by the remote application (and later received by the local application). krrrr Number of acknowledgements received by RHF for blocks sent by the application. id Connection path identifier in hexadecimal. cc Channel number in octal.	None.	RHF
RHF, APPLICATION DISABLED FOR NETON	The requested application is disabled. RHF does not abort the application.	None.	RHF
RHF, jobn APPLICATION IS NOT NETTED ON	The application with job name or job sequence name jobn requested an RHF function before requesting a NETON to RHF. RHF aborts the application.	None.	RHF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
RHF, jobn APPLICATION NOT VALIDATED FOR CTRL/INFO/R.	The application with job name or job sequence name jobn issued a CTRL/INFO/R supervisory request without the required system origin privileges.	Inform site analyst.	RHF
RHF, BAD TCU ON PATH id, PATH TURNED OFF.	Informative message to operator that the connection path identifier is turned off. id Path ordinal corresponding to the path assigned to the connection by RHF.	Inform customer engineer and site analyst. Verify that the LT, RT and NAD address parameters for the identified path are correct.	RHF
RHF, jobn BUFFER ADDRESS ERROR IN CTRL/INFO/R.	The buffer specified in the CTRL/INFO/R supervisory request for the network description table is outside the requesting application's field length. RHF aborts the application.	None.	RHF
RHF, jobn CONNECT TO applnam LID=lid PID=pid REJECTED RHF, jobn - rejmess.	RHF rejected connection request to application applnam for the reason given in the reject message rejmess. jobn Job name or job sequence name. applnam Name of application requested. lid Logical identifier of the remote mainframe. pid Physical identifier of the remote mainframe.	Refer to the separate listing of the last line message (RHF, jobn - rejmess) for the appropriate action.	RHF
DESTINATION DOES NOT RESPOND.	No response received from remote host.	Contact remote host operator to determine cause of error or inform a site analyst.	
LID NOT DEFINED AT SOURCE.	The requested LID does not exist in RHF's tables.	Inform site analyst.	
LID/PID/NAD UNAVAILABLE AT DESTINATION.	One of the following occurred: - For the remote RHF the requested LID or the requestor's PID is not valid. - Requested LID or requestor's PID is disabled in remote RHF configuration. - Remote NAD is not enabled in remote RHF configuration. - Path on which connection request was received is not enabled.	Contact remote host operator or inform a site analyst.	
LID/PID/NAD/DISABLED AT SOURCE.	Connection request denied because of one of the following: - LID or PID disabled. - Path is not enabled. - Local NAD is OFF in EST.	Determine the cause of the error by checking the EST and RHF's path and ID displays. If local NAD is OFF in EST, use DMPNAD and LOADBC to dump and load. Correct the problem and retry.	
NAD RESOURCE LIMIT REACHED.	Informative message.	Wait and retry.	
NO NEW CONNECT REQUESTS - MAX REACHED.	No new connection requests allowed because the maximum number of connections has been reached.	Inform site analyst. Maximum number of connects for the application must be increased in the RCFGEN file.	
REMOTE RHF SHUTDOWN IN PROGRESS.	Remote RHF is being shut down. New connections are not accepted.	None.	
REQUESTED APPLICATION NOT AVAILABLE.	Requested remote application is invalid, not running, disabled, no additional connections are allowed to the running application, or no new applications could be started on the remote mainframe.	Take corrective action and retry.	
RHF SHUTDOWN IN PROGRESS.	RHF is being shut down.	None.	
SUBSYSTEM PASSWORD REMOTE REJECT.	The remote RHF rejects the local RHF subsystem password.	Inform site analyst.	
UNKNOWN REMOTE RHF REJECT CODE xx.	Connection request rejected, and RHF does not recognize the reason for the reject. xx Rejection code in hexadecimal.	Inform site analyst.	

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
RHF, jobn CONNECTED TO applnam LID=lid PID=pid ACN=acn PATH ID=id CH=cc.	RHF accepted connection request to application applnam. jobn Job name or job sequence name. applnam Name of application requested. lid Logical identifier of the remote mainframe. pid Physical identifier of the remote mainframe. acn Application connection number in octal. id Connection path identifier in hexadecimal. cc Channel number in octal.	None.	RHF
RHF, jsn COULD NOT BE ABORTED.	The application with job sequence name jsn has been aborted by RHF for committing a fatal error, and has now committed a second fatal error. RHF forces a NETOFF of the application.	Inform site analyst.	RHF
RHF, DUPLICATE NETON REQUEST.	Two NETON requests were made for the same application without an intervening NETOFF request. RHF aborts the application.	Remove the second NETON request or add the missing NETOFF request.	RHF
RHF, jsn FATAL SSF ERROR. FC = fc, RC = rc.	The application with job sequence name jsn has been aborted by RHF after RHF received an SSF error rc on an SSF request fc. fc Function code. rc Reason code.	Inform site analyst.	RHF
RHF, FATAL SSF ERROR. FC = fc, RC = rc.	RHF has aborted itself or the offending application upon receiving a fatal reason code rc from an SSF request fc. fc Function code. rc Reason code.	Inform site analyst.	RHF
RHF, INVALID APPLICATION CALL TO RHF.	An application issued an invalid RHF call. The call may contain an incorrect RHF function, a request (other than NETON) from an application with an end-of-job connect status, or an incorrect word count in the RHF call. RHF aborts the application.	Correct error in application and retry.	RHF
RHF, jobn INVALID APPLICATION CALL TO RHF.	The application with job name or job sequence name jobn issued an invalid RHF call. The call may contain an incorrect RHF function, or an invalid word count, or the calling application may have an end-of-job connect status. RHF aborts the application.	None.	RHF
RHF, INVALID APPLICATION NAME ON NETON.	An application issued a NETON request using an application name that was not in RHF's configuration or that contained incorrect characters. RHF aborts the application.	Correct name in the application NETON call or add the application name to RHF's configuration.	RHF
RHF, jobn INVALID APPLICATION TABLE ADDRESS.	In an RHF call the application with job name or job sequence name jobn used an incorrect application table address. The address may be out of range or may point to another application table. RHF aborts the application.	Correct error in application.	RHF
RHF, INVALID APPLICATION TABLE ADDRESS.	In an RHF call an application used an incorrect application table address. The address may be out of range or may point to another application table. RHF aborts the application.	Correct error in application.	RHF
RHF, INVALID CONTROL MESSAGE FOR applnam ON ACN acn RECEIVED.	An incoming control message for application applnam on application connection number acn is not a valid control message. applnam Name of application requested. acn Application connection number in octal.	Inform site analyst.	RHF
RHF, INVALID FET PRAM. FET = address.	RHF has aborted this application for issuing a request to RHF that included a FET address or FET buffer pointer from a NETXFR request that was not within the applications field length. address The actual address that RHF found to be invalid. For FET buffer pointer errors, the address is the FET address.	Inform site analyst.	RHF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
RRHF, jsn INVALID HEADER ADDRESS address	System error.	Inform site analyst.	RHF
RHF, INVALID HEADER ADDRESS address.	RHF has aborted this application for issuing a request to RHF that included a header address from a NETGET or NETPUT request that was not within the applications field length. address The actual address that RHF found to be invalid.	Inform site analyst.	RHF
RHF, INVALID MINACN/MAXACN ON NETON.	The value for the minimum or maximum ACN in the NETON request is outside the range specified for the application. RHF aborts the application.	Correct the minimum or maximum ACN in the application's NETON request.	RHF
RHF, jsn INVALID SSF UCP ADDRESS address.	RHF has aborted an application with job sequence name jsn for issuing a request to RHF that included an SSF UCP address from any address given in an application request which RHF uses in an SSF request that was not within the application's field length. address The actual address that RHF found to be invalid.	Inform site analyst.	RHF
RHF, INVALID SSF UCP ADDRESS address.	RHF has aborted this application for issuing a request to RHF that included an SSF UCP address from any address given in an application request which RHF uses in an SSF request that was not within the application's field length. address The actual address that RHF found to be invalid.	Inform site analyst.	RHF
RHF, jsn INVALID TEXT ADDRESS address.	RHF has aborted an application with job sequence name jsn for issuing a request to RHF that included a text address (ta or ta + tlmax) from a NETGET or NETPUT request that was not within the application's field length. address The actual address that RHF found to be invalid.	Inform site analyst.	RHF
RHF, INVALID TEXT ADDRESS address.	RHF has aborted this application for issuing a request to RHF that included a text address (ta or ta + tlmax) from a NETGET or NETPUT request that was not within the application's field length. address The actual address that RHF found to be invalid.	Inform site analyst.	RHF
RHF, LID TABLE EMPTY.	The NOS central memory LID table is empty.	Inform site analyst.	RHF
RHF, MHF APPLICATION DISABLED.	Application MHF is disabled in RHF's tables, preventing automatic dumping and loading of a NAD that has stopped.	Use APPL display ENABLE command to enable MHF application.	RHF
RHF, MHF APPLICATION NOT DEFINED.	Application MHF is not defined in RHF's tables, preventing initial loading of local NADs and automatic dumping and reloading of a NAD that has stopped.	Correct the RHF configuration file by defining application MHF, and restart RHF.	RHF
RHF, NAD CODE CONVERSION ENABLED (EST=est,NB=yy,NP=zz).	Informative message. est EST oridnal of the NAD (octal). yy Number of convert mode buffers (hexadecimal). zz Number of convert mode paths (hexadecimal).	None.	RHF
RHF, NAD CODE CONVERSION NOT AVAILABLE (EST=est).	RHF unsuccessfully attempted to enable code conversion in the NAD on EST ordinal est.	Inform site analyst.	RHF
RHF, NAD on ESTest HAS BEEN TURNED OFF.	Informative message to operator that the NAD on EST ordinal est is turned off. est EST entry for NAD in octal.	Inform a customer engineer and a site analyst. If AUTODUMP and AUTOLOAD are enabled, MHF will attempt to dump and load the NAD.	RHF
RHF, jobn NETOFF AS applnam.	Informative message indicating application applnam with job name or job sequence name jobn ended access to RHF.	None.	RHF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
RHF, jobn NETON AS applnam ACCEPTED ACN=mina/maxa.	Informative RHF message indicating successful NETON of the application with job name or job sequence name jobn and application name applnam. The minimum and maximum ACN values specified in the NETON request were mina and maxa, respectively.	None.	RHF
RHF, jobn NETON AS applnam REJECTED ACN - mina/maxa. RHF, jobn - rejmess.	The application with job name or job sequence name jobn made a NETON request with application name applnam and minimum and maximum ACN values of mina and maxa, respectively. RHF rejected the NETON request for the reason given in the reject message rejmess.	Refer to the separate listing of the last line message (RHF, jobn - rejmess) for the appropriate action.	RHF
APPLICATION DISABLED FOR NETON	The requested application is disabled. RHF does not abort the application.	Inform site operator to enable application and retry.	
DUPLICATE NETON REQUEST	Two NETON requests were made for the same application without an intervening NETOFF request.	Remove the duplicate NETON or add a NETOFF request.	
INVALID APPLICATION NAME ON NETON	RHF does not recognize the application name in the NETON request. RHF aborts the application.	Retry and specify a valid application name.	
INVALID MINACN/MAXACN ON NETON	The value of the minimum or maximum ACN in the NETON request is outside the range specified for the application. RHF aborts the application.	Correct the value for the minimum or maximum ACN.	
NETON SECURITY VIOLATION	The application is not validated to do a NETON request. RHF aborts the application.	None.	
NO MORE SPACE FOR NETON	All allowable applications with the requested application name are netted on.	Retry the NETON request later.	
RHF, NETON SECURITY VIOLATION	An application is not validated to do a NETON request. RHF aborts the application.	None.	RHF
RHF, NO APPLICATION ADDRESS IN RHF CALL--EXTRA CHARGE	Informative message indicating an application issued an RHF request (other than NETON) without specifying an application table address in the RHF call. The application is charged less if it specifies its application table address in each RHF call.	None.	RHF
RHF, jobn NO APPLICATION ADDRESS IN RHF CALL--EXTRA CHARGE	Informative RHF message indicating the application with job name or job sequence name jobn issued an RHF request (other than NETON) without specifying an application table address in the RHF call. The application is charged less if it specifies an application table address in each RHF call. This informative message is issued only once after the first RHF call from the application with no application table address.	None.	RHF
RHF, NO MORE aname SPACE FOR NETON	The NETON is rejected because all allowable applications with the requested application name aname are currently netted on. RHF does not abort the application.	Retry the NETON request later.	RHF
RHF, NO MORE TABLE SPACE FOR NETON	RHF rejects NETON because there is no more table space available. RHF does not abort the application.	Retry the NETON request later.	RHF
RHF, jsn NOT IN RHF*S TABLES.	The job sequence name jsn that RHF used in an invalid SSF request does not exist in RHF's tables.	Inform site analyst.	RHF
RHF, QUEUED MESSAGE LIMIT EXCEEDED	An application exceeded the maximum number of supervisory messages that are queued in RHF. RHF aborts the application.	Modify the application to issue more frequent NETGET's for the supervisory messages queued in RHF.	RHF
RHF, jobn QUEUED MESSAGE LIMIT EXCEEDED	The application with job name or job sequence name jobn is aborted if the number of supervisory messages queued in RHF exceeds the limit.	None.	RHF

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
RHF, REJECTED CONTROL MESSAGE FOR applnam ON ACNacn RECEIVED	The NAD rejected a control message sent by application applnam on application connection number acn. applnam Application name of requestor. acn Application connection number in octal.	Inform a customer engineer and a site analyst.	RHF
RHF, REMOTE CONNECT REQUEST FROM aplname ON pid TO applnam LID=lid RHF, NAD=nn, CH=cc, BUFF=b, TCU=nnnn, DEST=d, PATH ID=id, ACC=cccc RHF, REQUEST ACCEPTED.	Informative message indicating a connection request from a remote host has been accepted by RHF. aplname Application name of requestor. pid Physical identifier of remote mainframe where request initiated. applnam Name of application requested. lid Logical identifier requested (valid lid for remote mainframe pid). nn Address of NAD issuing request in hexadecimal. cc Channel number of receiving NAD in octal. b Buffer size of the allocation request in octal. 0 516 bytes 1 2064 bytes 2 4128 bytes nnnn Binary bit pattern which indicates trunks that may be used to communicate back to the requesting NAD. d Destination device physical address in hexadecimal. id Connection path identifier in hexadecimal. The NAD gives this id to the connection path. cccc Access code in hexadecimal.	None.	RHF
RHF, REMOTE CONNECT REQUEST FROM aplname ON pid TO applnam LID=lid RHF, NAD=nn, CH=cc, BUFF=b, TCU=nnnn, DEST=d, PATH ID=id, ACC=cccc RHF, REQUEST REJECTED - rejmess.	Informative message indicating rejection by RHF of a remote host's connection request for the reason given in the reject message rejmess. aplname Application name of requestor. pid Physical identifier of remote mainframe where request initiated. applnam Name of application requested. lid Logical identifier requested. nn Address of NAD issuing request in hexadecimal. cc Channel number of receiving NAD in octal. b Buffer size of the allocation request in octal. 0 516 bytes 1 2064 bytes 2 4128 bytes nnnn Binary bit pattern which indicates trunks that may be used to communicate back to the requesting NAD. d Destination device physical address in hexadecimal. id Connection path identifier in hexadecimal. The NAD gives this id to the connection path. cccc Access code in hexadecimal.	Refer to the separate listing of the last line message (RHF, REQUEST REJECTED - rejmess) for the appropriate action.	RHF
INVALID PASSWORD ppppppp	Self explanatory. ppppppp Password.	Inform site analyst.	
PATH OR NAD UNAVAILABLE	One of the following is not in the RHF configuration or is disabled. - Remote NAD. - Local NAD. - Path between the remote and local NADs. This message may also be issued if no TCU enables in the RHF configuration for this path are in common with the TCU enables specified by the requesting NAD.	If appropriate, enable corresponding elements in RHF configuration, or correct TCU enables in RHF configuration.	

MESSAGE	SIGNIFICANCE	ACTION	ROUTINE
PID/LID NOT AVAILABLE	Either the PID of the requestor is not in the RHF configuration or it is disabled or the LID requested is not in the LID table.	If appropriate, enable PID or LID in RHF configuration or add LID to LID table.	
REQUESTED APPLICATION UNAVAILABLE	Requested remote application is invalid, not running, disabled, no additional connections are allowed to the running application, or no new applications could be started on the remote mainframe.	Take corrective action and retry.	
RHF SHUTDOWN IN PROGRESS	New connections are not made during the RHF shutdown process.	None.	
RHF, SSF ERROR, jsn NOT IN SYSTEM.	RHF issued an SSF request that referenced an application, with job sequence name jsn, that is not currently known to the system.	Inform site analyst.	RHF
RHF, THE FOLLOWING CONTROL MESS. IS FOR AN UNKNOWN PATH. RHF, xx...xx	A control message for which no active path could be found in any application connection table entry was receivced by RHF from a local NAD. xx...xx is the contents of the control message in hexadecimal.	Inform site analyst if problem persists.	RHF
RHF, THE FOLLOWING CONTROL MESS. IS NOT SUPPORTED. RHF, xx...xx	A control message which could not be recognized by RHF was received from a local NAD. xx...xx is the contents of the control message in hexadecimal.	Inform site analyst if problem persists.	RHF
RHF, THE FOLLOWING CONTROL MESS. WAS REJECTED BY THE NAD. RHF, xx...xx	A control message issued by RHF could not be delivered by a local NAD and was returned as a rejected control message. xx...xx is the contents of the control message in hexadecimal.	Inform site analyst if problem persists.	RHF
RHH01 - INVALID FUNCTION CODE.	The calling program specified an invalid function code.	Inform site analyst.	RHH
RHH02 - INVALID PARAMETER BUFFER ADDRESS.	The calling program specified a parameter block address that was either zero or not within the caller's field length.	Inform site analyst.	RHH
RHH03 - USER NOT SYSTEM ORIGIN.	The calling program does not have a subsystem job entry point.	Inform site analyst.	RHH
RHH04 - CALLED FROM CP 0.	This routine cannot be called from control point 0.	Inform site analyst.	RHH
RHH05 - NOT CALLED BY SUBSYSTEM CONTROL POINT.	A program specified a function that can be used only by RHF.	Inform site analyst.	RHH
RHH06 - NO SUBSYSTEM CONTROL TABLE FOUND	RHH was attempting to find RHF's control point number, but a subsystem table has not been defined.	Rebuild CMR with a subsystem table.	RHH
RHH07 - INVALID SUBSYSTEM NUMBER	RHH was given a subsystem number that is not within the valid range of subsystem numbers.	Inform site analyst.	RHH
RHH10 - NO LOCAL NADS.	RHF does not have any local NADs in the local NAD table.	Inform site analyst. The RCFILE for RHF must be corrected.	RHH
RHH11 - NAD TABLE OUT OF RANGE.	The address of the local NAD table was not within RHF's field length.	Inform site analyst.	RHH
RHH12 - CHANNEL NOT FOUND ERROR IN LNT ORDINAL ord.	Informative message that indicates which LNT entries in RHF do not have a corresponding EST entry.	None.	RHH
RHH13 - CONFIGURATION ERROR.	The equipment status table (EST) contains more than one NAD entry with the same channel.	Inform site analyst. The CMRDECK must be changed.	RHH
RHH21 - MORE THAN ONE LNT WITH SAME CHANNEL.	The local NAD table (LNT) in RHF has more than one local NAD entry with the same channel specified.	Inform site analyst. RCFILE for RHF must be changed to avoid duplicate channel entries.	RHH
RHH22 - INVALID EST ORDINAL.	The calling program asked to update an equipment status table (EST) entry that does not exist.	Inform site analyst.	RHH
RHH23 - EST IS NOT A NAD.	The calling program asked to update an equipment status table (EST) entry that was not for a NAD.	Inform site analyst.	RHH

GLOSSARY B

This glossary defines terms unique to the description of the software presented in this manual. It also contains terms whose interpretation within this manual is different than that commonly made.

ABH

Refer to Application Block Header.

ABN

Refer to Application Block Number.

ABT

Refer to Application Block Type.

Acknowledged Data Block (non-PRU data block)

A data block for which the receiving host acknowledges delivery.

ACN

Refer to Application Connection Number.

ACT

Refer to Application Character Type.

ALN

Refer to Application List Number.

ANAME

Refer to Application Name.

Application Block Header (ABH)

A single 60-bit word description accompanying every block passing between an application program and RHF.

Application Block Number (ABN)

A field in the application block header. An application-assigned number used to identify a particular data block.

Application Block Type (ABT)

A field in the application block header defining the accompanying block as either data or supervisory, null or not null, and indicating if this is the last block of a message.

Application Character Type (ACT)

A field in the application block header defining the type and size of text characters.

Application Connection Number (ACN)

A number assigned by RHF to identify a particular logical connection within an application program.

Application List Number (ALN)

An application program-assigned number used to identify a particular group of logical connections belonging to the application program.

Application Name (ANAME)

Up to seven letters or digits (the first must be a letter) used to identify an application program. It is used by another application program when connection to the application is requested.

Application Program

A program resident in a host computer that provides an information storage, retrieval, and/or processing service using the data communication network through the services of RHF.

Block

In the context of network communications, a portion or all of a message. A message is divided into blocks to facilitate buffering, transmission, error detection, and correction of variable length data streams.

Block Header

Refer to Application Block Header.

Byte

A group of bits. Unless prefixed (for example, a 6-bit byte), the term implies an 8-bit grouping of data. When used for encoding character data, a byte represents a single character.

Character

Unless otherwise specified, references to characters in this manual are to 7-bit ASCII code characters.

Connection

Refer to Logical Connection.

Connection Number

Refer to Application Connection Number.

Data

Any portion of a message created by the source, exclusive of any information used to accomplish transmission of such a message.

Data Block Clarifier (DBC)

A field in the network block header accompanying each data block. The data block clarifier is used to identify unacknowledged data blocks and higher level file structures.

DBC

Refer to Data Block Clarifier.

Destination

The application program designated to receive the message.

Facility Interface Program (FIP)

A group of routines that resides in the application program's field length. These routines translate and buffer communications between the application program and the network, using the system control point feature of the operating system.

FET

Refer to File Environment Table.

File Environment Table (FET)

A table defined by an application program to communicate with operating system input/output routines or with RHF.

FIP

Refer to Facility Interface Program.

HA

Refer to Header Area.

Header Area (HA)

A 60-bit word within the application program containing the application block header for a NETPUT call or the area to receive the header for a NETGET or NETGETL call.

Host

A computer that executes application programs.

Initiator or Initiating Application

An initiator is an application program responsible for the initiation of an application- to-application connection using RHF.

Input Parameter

A parameter in a FIP call that provides input to the FIP routine. An input parameter can be a constant, an expression, or a symbolic address for such values. Input parameters are not altered by the completion of FIP processing.

LCN

Refer to Loosely Coupled Network.

LID

Refer to Logical Identifier.

List

A group of logical connections with the same application list number; the logical connections are linked together by RHF and treated as a single entity in NETGETL calls.

Local Host

The host to which the user is most directly associated. For example, to an interactive user the local host is the host to which the interactive terminal is connected.

Logical Connection

A logical message path established between two application programs. Until terminated, the logical connection allows messages to pass between the two entities.

Logical Identifier (LID)

Three-character identifier representing a host or hosts where a given function is expected to be performed. For example, a site may select A96 as the LID representing a host that has the ability to print upper/lower case output files. Since the LID is a logical identifier, the site may set/change the LID/host associations at will. A site may associate a LID with a number of hosts; a site may define a number of LIDs to be associated with one host or several hosts.

Loosely Coupled Network (LCN)

An LCN is a local network using two or more network access devices (NADs) to interface a variety of mainframes using a coaxial trunk.

Message

A logical unit of information, as processed by an application program. When transmitted over a network, a message can consist of one or more blocks.

NAD

Refer to Network Access Device.

Network

An interconnected set of network access devices.

Network Access Device (NAD)

The CDC 380 NAD is a hardware unit that, along with the controlware in the NAD and the operating system software, allows a mainframe to access the LCN.

Network Block Header

A 96-bit header that is prefixed to each data block by the sending RHF and is removed by the receiving RHF before delivering the data block to an application program.

Non-PRU Data Block

Refer to Acknowledged Data Block.

Physical Identifier (PID)

The unique three-character identifier of a specific host.

PID

Refer to Physical Identifier.

Protocol

A network protocol consists of the following elements:

- Syntax - The structure of commands and responses in either field-formatted or character-string form.
- Semantics - The set of requests to be issued, actions to be performed, and responses returned.
- Timing - The ordering of events.

PRU Data Block

Refer to Unacknowledged Data Block.

Remote Host

Any host other than the local host.

Remote Host Facility (RHF)

The remote host facility is an access method providing a network access method-like interface to its application programs. RHF provides the means for its application programs to access LCN and thus provides user functions such as permanent file transfers across LCN.

Return Parameter

A parameter in a FIP call that provides as input to the FIP routine the identification of a location to which FIP should transfer information. This location is within the application program's field length and outside the FIP portion of that field length. A return parameter cannot be a constant or a value in itself. Return parameters are always symbolic addresses. The time when transfer of information from FIP occurs depends on whether use of the parameter is global to all FIP routines or local to the call in which it is used.

RHF

Refer to Remote Host Facility.

Servicer or Servicing Application

A servicer is an application program started automatically by RHF upon request of an initiator on another host. Functions such as permanent file transfer can be accomplished by communciation and data transfer between initiator and servicer.

Source

The host computer program that creates a message.

Supervisory Message

A message block not directly involved with the transmission of data but which provides information for establishing and maintaining an environment for the communication of data between the application program and RHF, and through the network to a destination or from a source.

TA

Refer to Text Area.

TCU

Refer to Trunk Control Unit.

Text Area (TA)

The area within the application program that receives the message block text from a NETGET or NETGETL call or contains the message block text for a NETPUT call.

Text Length In Characters (TLC)

A field in the application block header specifying the number of character bytes of text in the message block.

Text Length Maximum (TLMAX)

Maximum length in central memory words of the data message text block that the application program will accept for processing.

TLC

Refer to Text Length In Characters.

TLMAX

Refer to Text Length Maximum.

Trunk

The communication line connecting two network access devices.

Trunk Control Unit (TCU)

A TCU is the hardware portion of a NAD that directly interfaces to a single trunk. There can be a maximum of four TCU's per NAD.

Unacknowledged Data Block (PRU data block)

A data block for which the receiving host does not acknowledge delivery.

RESERVED NAMES AND SYMBOLS C

This appendix defines those names and symbols that have special meaning to RHF. Improper use of these reserved names and symbols may not produce a diagnostic message but generally causes problems in coding an application program.

RESERVED NAMES

The following reserved program names should not be used in a NETON statement. These names are reserved by CDC for other uses within the network software.

ALL	FTFS	LOGOUT	NUL	QTFS
BYE	HELLO	MCS	NVF	RBF
CS	IAF	MHF	PTF	SCF
DOP	ITF	MLTF	PTFS	TAF
FTF	LOGIN	NS	QTF	TVF

RESERVED SYMBOLS

The symbols in table C-1 are predefined labels for supervisory message words or fields. Reference to the symbols should be made using display-code characters. Each symbol appears as it should be used in calls to NFETCH or NSTORE. The symbols are listed alphabetically within the table.

Each symbol consists of the characters identifying its field within a message, combined with characters identifying the specific message or group of messages. For example:

- All primary function code fields can be accessed through the symbol PFC.
- All fields in messages with the primary function code mnemonic CON begin with CON; the application list number field in such messages is, therefore, CONALN.
- All fields in the application block header word can be accessed through symbols beginning with ABH.

Some symbols are restricted for use in certain contexts. For example, the call

```
IVAL=NFETCH(0,L"CONEND")
```

returns the primary and secondary code value for the corresponding fields in a CON/END/R message. However, the call

```
CALL NSTORE(SMTA,L"CONEND",IVAL)
```

causes an error message indicating that the symbol CONEND is unrecognized. The symbol is unrecognized because its context is incorrect. The correct call to store the information is

```
CALL NSTORE(SMTA,L"PFCSFC",IVAL)
```

or the call

```
CALL NSTORE(SMTA,L"PFCSFC",L"CONEND")
```

Table C-1. Reserved Symbols (Sheet 1 of 4)

Symbol	Entity Defined by Symbol	Predefined Integer Value
ABHABN	Application block number field in application block header for all blocks.	None
ABHABT	Application block type field in application block header for all blocks.	None
ABHACT	Application character type field in application block header for all blocks.	None
ABHADR	Process number address field in application block header for supervisor program blocks (system use only). Application connection number field in application block header for all application program blocks.	None
ABHIBU	Input block undeliverable bit in application block header for input blocks.	None
ABHTLC	Text-length-in-character-units field in application block header for all blocks.	None
ABHWORD	Application block header word for all blocks.	None
ACK	Secondary function code field for FC/ACK/R.	2
ACRQ	Secondary function code field for CON/ACRQ messages.	2
BRK	Secondary function code field for FC/BRK/R.	0
CB	Secondary function code field for CON/CB/R.	5
CICT	Secondary function code field for DC/CICT/R.	0
CM	Primary function code field for convert mode control supervisory messages.	$C3_{16}$
CON	Primary function code field for connection management messages.	63_{16}
CONABL	Application block limit field in CON/REQ/R.	None
CONACN	Application connection number field in connection management messages.	None
CONACR	Primary and secondary function code fields for CON/ACRQ/R, including EB and RB fields as zero.	6302_{16}
CONACT	Application input character type field in CON/REQ/N.	None
CONALN	Application list number field in CON/REQ/N.	None

Table C-1. Reserved Symbols (Sheet 2 of 4)

Symbol	Entity Defined by Symbol	Predefined Integer Value
CONANM	Requesting application program name in CON/REQ/R. Requested application program name in CON/ACRQ/R.	None
CONCB	Primary and secondary function code fields for CON/CB/R, including EB and RB fields as zero.	6305_{16}
CONEND	Primary and secondary function code fields in CON/END/R, including EB and RB fields as zero.	6306_{16}
CONREQ	Primary and secondary function code fields in CON/REQ/R, including EB and RB fields as zero.	6300_{16}
CTRINF	Primary and secondary function code fields in CTRL/INFO/R, including EB and RB fields as zero.	$C10A_{16}$
CTRL	Primary function code field in control messages.	$C1_{16}$
DC	Primary function code field in DC/CICT/R.	$C2_{16}$
DCACN	Application connection number field in DC/CICT/R.	None
DCCICT	Primary and secondary function code fields in DC/CICT/R, including EB and RB fields as zero.	$C200_{16}$
EB	Error bit in all supervisory messages.	None
ENDD	Secondary function code field in CON/END messages.	6
ERR	Primary function code field in ERR/LGL/R.	84_{16}
ERRABH	Application block header word in ERR/LGL/R.	None
ERRLGL	Primary and secondary function code fields in ERR/LGL/R, including EB and RB fields as zero.	8401_{16}
ERRMSG	First message text word in ERR/LGL/R.	None
EXIT	Secondary function code field in CM/EXIT/R.	1
FC	Primary function code field in flow control supervisory messages.	83_{16}
FCABN	Application block number field in flow control supervisory messages.	None
FCACK	Primary and secondary function code fields in FC/ACK/R, including EB and RB fields as zero.	8302_{16}
FCACN	Application connection number field in flow control supervisory messages.	None

Table C-1. Reserved Symbols (Sheet 3 of 4)

Symbol	Entity Defined by Symbol	Predefined Integer Value
FCBRK	Primary and secondary function code fields in FC/BRK/R, including EB and RB fields as zero.	8300_{16}
FCINIT	Primary and secondary function code fields in FC/INIT/R, including EB and RB fields as zero.	8307_{16}
FCNAK	Primary and secondary function code fields in FC/NAK/R, including EB and RB fields as zero.	8303_{16}
FCRST	Primary and secondary function code fields in FC/RST/R, including EB and RB fields as zero.	8301_{16}
INFO	Secondary function code field in control messages.	A_{16}
INIT	Secondary function code field in FC/INIT/R.	7
INSD	Secondary function code field in SHUT/INSD/R.	6
LCONAC	Length in 60-bit words on CON/ACRQ messages.	2
LCONCB	Length in 60-bit words of CON/CB/R.	1
LCONEN	Length in 60-bit words of CON/END/R.	2
LCONRQ	Length in 60-bit words of CON/REQ/R.	4
LCORQR	Length in 60-bit words of CON/REQ/N and CON/REQ/A.	1
LDC	Length in 60-bit words of DC/CICT/R.	1
LERR	Length in 60-bit words of ERR/LGL/R.	3
LFC	Length in 60-bit words of flow control supervisory messages.	1
LGL	Secondary function code field in ERR/LGL/R.	1
LLST	Length in 60-bit words of list management supervisory messages.	1
LSHUT	Length in 60-bit words of SHUT/INSD/R.	1
LST	Primary function code field in list management supervisory messages.	$C0_{16}$
LSTACN	Application connection number field in list management management supervisory messages.	None

Table C-1. Reserved Symbols (Sheet 4 of 4)

Symbol	Entity Defined by Symbol	Predefined Integer Value
LSTALN	Application list number field in list management supervisory messages.	None
LSTOFF	Primary and secondary function code fields in LST/OFF/R, including EB and RB fields as zero.	$C000_{16}$
LSTON	Primary and secondary function code fields in LST/ON/R, including EB and RB fields as zero.	$C001_{16}$
LSTSWH	Primary and secondary function code fields in LST/SWH/R, including EB and RB fields as zero.	$C002_{16}$
NAK	Secondary function code field in FC/NAK/R.	3
OFF	Secondary function code field in LST/OFF/R.	0
ON	Secondary function code field in LST/ON/R.	1
PFC	Primary function code field in all supervisory messages.	None
PFCSFC	Primary and secondary function code fields in all supervisory messages, including EB and RB fields as zero.	None
RB	Response bit in all supervisory messages.	None
RC	Reason code field in all supervisory messages.	None
REQ	Secondary function code field in CON/REQ messages.	0
RST	Secondary function code field in FC/RST/R.	1
SEL	Secondary function code field in CM/SEL/R.	0
SFC	Secondary function code field in all supervisory messages.	None
SHUT	Primary function code field in SHUT/INSD/R.	42_{16}
SHUINS	Primary and secondary function code fields in SHUT/INSD/R, including EB and RB fields as zero.	4206_{16}
SHUTYP	Type of shutdown field in SHUT/INSD/R.	None
SPMSGO thru SPMSG9	The corresponding word zero through nine of any supervisory message.	None
SWH	Secondary function code field in LST/SWH/R.	2

FIP CALL SUMMARY

D

This appendix summarizes the formats of calls to FIP. The general format of each call is listed alphabetically. Refer to section 4 for complete descriptions of these call formats.

Compiler level call formats are:

```
CALL NETDBG (dbugsup,dbugdat,dbugstat,dbugxfr)

CALL NETGET (acn,ha,ta,tlmax)

CALL NETGETL (aln,ha,ta,tlmax)

CALL NETOFF

CALL NETON (aname,nsup,status,minacn,maxacn)

CALL NETPUT (ha,ta)

CALL NETUXFR (acn,fetadr,reqarea,fetfc,conmode)

CALL NETWAIT (time,flag)

CALL NETXFR (acn,fname,op,status,wait,dd,tmout,abl,facil,bsize,ackw,ickval)

CALL NETXFRC

CALL NSTORE (array,field,value)

ivalue=NFETCH (array,field)
```

Assembly level call formats are:

Location	Operation	Variable
label+	NETDBG	dbugsup,dbugdat,dbugstat,dbugxfr
label	NETDBG	dbugsup,dbugdat,dbugstat,dbugxfr,LIST
label+	NETDBG	LIST=label or register name
label+	NETGET	acn,ha,ta,tlmax
label	NETGET	acn,ha,ta,tlmax,LIST
label+	NETGET	LIST=label or register name

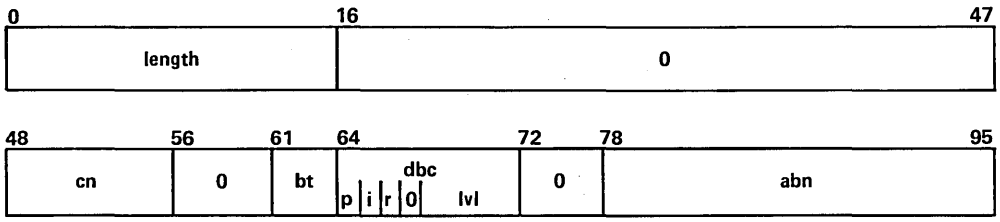
+ Label is optional for this macro call.

Location	Operation	Variable
label+	NETGETL	aln,ha,ta,tlmax
label	NETGETL	aln,ha,ta,tlmax,LIST
label+	NETGETL	LIST=label or register name
label+	NETOFF	
label+	NETON	aname,nsup,status,minacn,maxacn
label	NETON	aname,nsup,status,minacn,maxacn,LIST
label+	NETON	LIST=label or register name
label+	NETPUT	ha,ta
label	NETPUT	ha,ta,LIST
label+	NETPUT	LIST=label or register name
label+	NETUXFR	acn,fetadr,reqarea,fetfc,conmode
label	NETUXFR	acn,fetadr,reqarea,fetfc,conmode,LIST
label+	NETUXFR	LIST=label or register name
label+	NETWAIT	time,flag
label	NETWAIT	time,flag,LIST
label+	NETWAIT	LIST=label or register name
label+	NETXFR	acn,fname,op,status,wait,dd,tmout,abl,facil,bsize,ackw,ickval
label	NETXFR	acn,fname,op,status,wait,dd,tmout,abl,facil,bsize,ackw,ickval, LIST
label+	NETXFR	LIST=label or register name
label+	NETXFRC	
label+	NFETCH	array,field,Xj or Bj
label+	NSTORE	array,field=value

+ Label is optional for this macro call.

NETWORK BLOCK HEADER E

Each LCN network data block transmitted by one RHF and received by another RHF is prefixed by a 96-bit header in the following format:



Field	Description

length — Text length. This field contains the length of the text portion of the data block in bits, not including the 96-bit header. The maximum text length for a data block follows.

length	Data Block
16 344 bits	Acknowledged data block.
23 040 bits	Unacknowledged data block in binary format.
30 720 bits	Unacknowledged data block in character format.

cn — Connection number. This field contains the local path number of the RHF transmitting the data block. $1 \leq cn \leq 127$.

bt — Block type. This field can have one of the following values:

Value	Mnemonic	Description
1	BLK	Data block that is not the last block of a multiblock message.
2	MSG	Data block that is the last block or only block of a message.
3	BACK	Block acknowledgment. This field is used to identify a data block (type BLK or MSG) as one for which the receiving host acknowledges delivery.
4	CMD	Data block that contains control information or a service message.
5-7		Reserved for CDC.

Field	Description
dbc	Data block clarifier. This field is used to identify a data block (type BLK or MSG) as a special unacknowledged data block, and to identify higher level file structures controlling the data. This field contains the following subfields.

Subfield	Description
p	PRU block flag. When set, this flag identifies the block as an unacknowledged data block. When clear, this flag identifies the block as an acknowledged data block.
i	End of information (EOI) flag for unacknowledged data blocks.
r	End of record (EOR) flag for unacknowledged data blocks. This field must be set for subfield lvl to be meaningful.
lvl	End of record level number, used in combination with subfield r; $0 \leq lvl \leq 17_8$. $lvl = 17_8$ denotes the end of file (EOF).

abn Application block number. This field contains an 18-bit integer that identifies the data block for acknowledgment or flow control purposes.

For acknowledged data blocks (PRU block flag clear), the receiving RHF returns a BACK block type to the transmitting RHF with an application block number equal to that of the received data block.

For unacknowledged data blocks (PRU block flag set), this field contains a binary number that is incremented by one for each data block sent (modulo 2^{18}). The number always starts with zero and must be sequential through the end of information.

COMPATIBILITY WITH OTHER RHF ACCESS METHODS F

When using other RHF access methods to communicate with NOS 2 or NOS/BE RHF applications, the following RHF restrictions must be observed.

- Acknowledged data blocks are sent using a NETPUT call and received using a NETGET or NETGETL call. The text length for acknowledged data blocks cannot exceed 2043 8-bit bytes (16 344 bits).

- Unacknowledged data blocks are usually sent and received using the NETUXFR FET I/O interface. The maximum text length for unacknowledged data blocks in character format (DD = C6 or C8)+ is 3840 8-bit bytes (30 720 bits). The maximum text length for unacknowledged data blocks in binary format (DD = UU, US, or UH)+ is 2880 8-bit bytes (23 040 bits).

  If the data block clarifier field in the network block header does not have either the EOR or EOI flags set, the unacknowledged data block's text length must be exactly the maximum length for the selected data format (2880 or 3840 8-bit bytes). Data blocks with either the EOR flag set or the EOI flag set can have a text length ranging from zero to the maximum text length for the selected data format. However, character data blocks should always be a multiple of 8-bits.

  To ensure that application block numbers are in the correct sequence, RHF checks the number when receiving and generates the number when sending unacknowledged data blocks.

- Unacknowledged data blocks can be sent using a NETPUT call if the local acknowledge flag is set in the corresponding application block header. The text length for these unacknowledged data blocks cannot exceed 2043 8-bit bytes. Also, the PRU block and EOR flags must be set in the data block clarifier field of the network block header. The application program must ensure that the application block number specified in the application block header is incremented for each data block sent.

- Unacknowledged data blocks can be received using a NETGET or NETGETL call as long as the text length of these data blocks does not exceed 2043 8-bit bytes. The application program has no access to the contents of the data block clarifier field in the network block header. The application program is responsible for verifying the sequence of application block numbers received in the application block header.

- The NETXFR entry point transfers files using the data transfer phase of the RHF application to application protocol. This includes all protocol data blocks and block acknowledgements from the Start of Data (SS)++ command through the End Acknowledge (ER)++ command. All unacknowledged data blocks must obey the restrictions previously described for the NETUXFR FET I/O interface.

+ Refer to the Remote Host Facility Usage manual for information concerning the DD parameter in the MFLINK or MFQUEUE commands.

++ The SS and ER commands are defined in the RHF application to application protocol. They are included in this manual only as a reference to show what is included in the data transfer phase.

INDEX

COMMENT SHEET

MANUAL TITLE: CDC Remote Host Facility Access Method Reference Manual

PUBLICATION NO.: 60459990 REVISION: A

NAME:_____

COMPANY:_____

STREET ADDRESS:_____

CITY:_____ STATE:_____ ZIP CODE:_____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

☐ Please Reply ☐ No Reply Necessary

CUT ALONG LINE

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

)LD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

LD FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A

CONTROL DATA CORPORATION