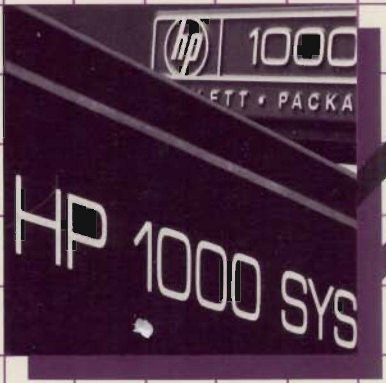
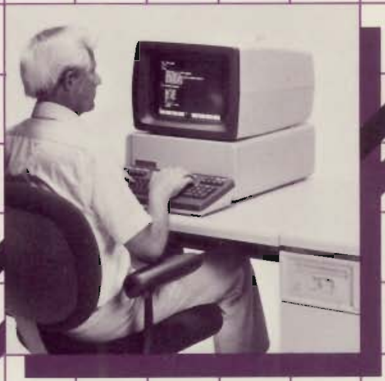
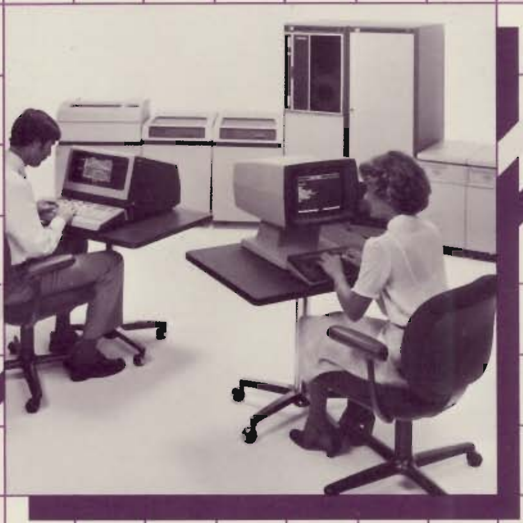
HP AdvanceNet

HEWLETT PACKARD

DS/1000-IV
User's Manual







DS/1000-IV

User's Manual

HEWLETT
PACKARD

19420 Homestead Road, Cupertino, California 95014

Part No. 91750-90002

Printed in U.S.A. March 1985

PRINTING HISTORY

The Printing History below identifies the Edition of this Manual and any Updates that are included. Periodically, Update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this Printing History page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past Updates, however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all Updates.

To determine what manual edition and update is compatible with your current software revision code, refer to the appropriate Software Numbering Catalog, Software Product Catalog, or Diagnostic Configurator Manual.

First Edition	Oct 1980	
Update 1	Jul 1981	
Update 2	Oct 1981	
Update 3	Apr 1982	
Reprint	April 1982	(Updates 1 through 3 incorporated)
Update 4	Jul 1982	
Update 5	Jan 1983	
Update 6	Jun 1983	
Reprint	Jun 1983	(Updates 4 through 6 incorporated)
Update 7	Nov 1983	
Update 8	Jun 1984	
Second Edition	Mar 1985	

Preface

The Distributed Systems 1000-IV User's Manual is the primary reference source for programmers and operators who will be writing or maintaining programs within a Distributed Systems 1000-IV (DS/1000-IV) network. This manual should also be read by Network Managers before designing a DS/1000-IV network so that they will have a clear understanding of the full implications of various DS/1000-IV functions and features.

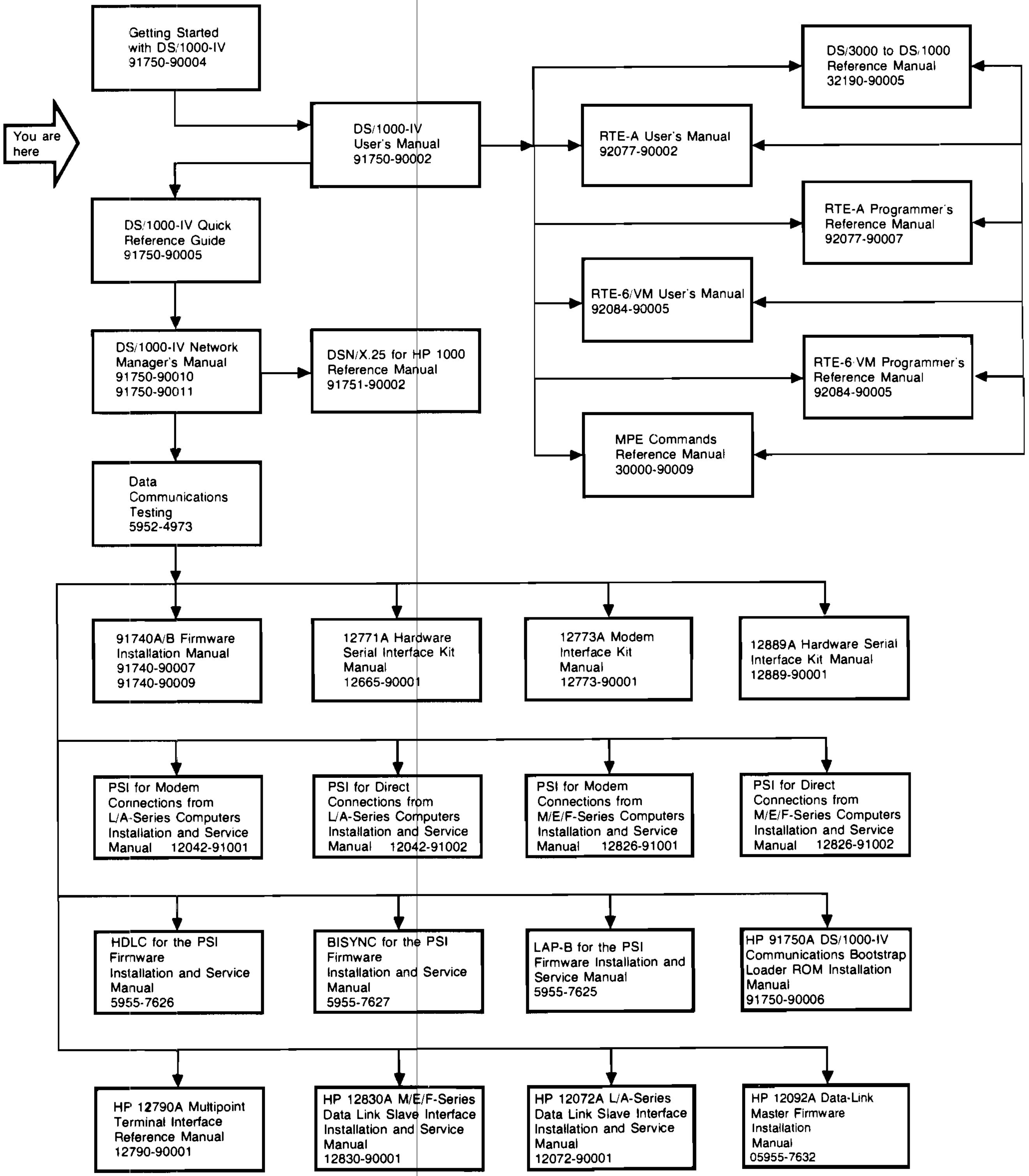
The purpose of this manual is to define the rules and procedures for developing and executing programs within a DS/1000-IV network. It is assumed that readers of this manual already have a working knowledge of the Real Time Executive (RTE) and Multi-Programming Executive (MPE) operating systems used in the HP 1000 and HP 3000 computer systems, respectively.

The DS/1000-IV User's Manual is divided into nine parts and two appendices as follows:

Chapter 1	Gives an overview of Distributed Systems and a summary of the capabilities of DS/1000-IV.
Chapter 2	Describes the commands, format, parameters, and usage of operator interface program REMAT; used to send RTE commands or special DS/1000-IV commands to any HP 1000 RTE node.
Chapter 3	Describes the commands, format, parameters, and usage of operator interface program RMOTE; used to direct operator commands to a HP 3000 node.
Chapter 4	Describes the Remote File Access subroutines with which you can accomplish management of remote disc and non-disc files from your program. The format for these calls is given along with examples.
Chapter 5	Describes the DEXEC calls which provide you with the ability to control I/O devices located anywhere in your network. The formats, parameters, and usage are explained.
Chapter 6	Describes the Program-to-Program (PTOP) calls which provide a data exchange interface between two programs located in different areas of your network. Their format, parameters, and usage are explained.

Chapter 7	Describes the utility programs provided with DS/1000-IV.
Chapter 8	Describes the loaders available for remote program loading and relocation.
Chapter 9	HP 3000 Intrinsics are discussed in Chapter 9.
Appendix A	HP Character Set
Appendix B	Error Codes and Messages

DOCUMENTATION MAP DS/1000-IV



Getting Started with DS/1000-IV 91750-90004
You are here
DS/1000-IV User's Manual 91750-90002
DS/1000-IV Quick Reference Guide 91750-90005
DS/1000-IV Network Manager's Manual 91750-90010 91750-90011
DSN/X.25 for HP 1000 Reference Manual 91751-90002
Data Communications Testing 5952-4973
DS/3000 to DS/1000 Reference Manual 32190-90005
RTE-A User's Manual 92077-90002
RTE-A Programmer's Reference Manual 92077-90007
RTE-6/VM User's Manual 92084-90005
RTE-6/VM Programmer's Reference Manual 92084-90005
MPE Commands Reference Manual 30000-90009
91740A/B Firmware Installation Manual 91740-90007 91740-90009
12771A Hardware Serial Interface Kit Manual 12665-90001
12773A Modem Interface Kit Manual 12773-90001
12889A Hardware Serial Interface Kit Manual 12889-90001
PSI for Modem Connections from L/A-Series Computers Installation and Service Manual 12042-91001
PSI for Direct Connections from L/A-Series Computers Installation and Service Manual 12042-91002
PSI for Modem Connections from M/E/F-Series Computers Installation and Service Manual 12826-91001
PSI for Direct Connections from M/E/F-Series Computers Installation and Service Manual 12826-91002
HDLC for the PSI Firmware Installation and Service Manual 5955-7626
BISYNC for the PSI Firmware Installation and Service Manual 5955-7627
LAP-B for the PSI Firmware Installation and Service Manual 5955-7625
HP 91750A DS/1000-IV Communications Bootstrap Loader ROM Installation Manual 91750-90006
HP 12790A Multipoint Terminal Interface Reference Manual 12790-90001
HP 12830A M/E/F-Series Data Link Slave Interface Installation and Service Manual 12830-90001
HP 12072A L/A-Series Data Link Slave Interface Installation and Service Manual 12072-90001
HP 12092A Data-Link Master Firmware Installation Manual 05955-7632

NOTE

DS/1000-IV cannot access non-FMGR files. Files used for CBL, DSVCP, Forced Cold Loads, REMAT, RFA, RMOTE and all other DS/1000-IV utilities must be FMGR files.

Table of Contents

Chapter 4 DS/1000-IV Remote File Access

Chapter 5 DS/1000-IV DEXEC Calls

List of Illustrations

List of Tables

Chapter 1 General Description

Introduction to Computer Networks

A computer network is a collection of many types of equipment and software. The major components are generally designated as nodes and links. A node is a computer system with its associated operating system and appropriate communication software connected to other nodes by communication links. Messages are sent to other computers over these communication links which may be physically hardwired or modem connections. The link includes the interface boards and cables.

One significant feature of a network is resource sharing. That is, elements at each node are accessible from other nodes in the network. These elements could include disc files, printers, mag tapes, terminals, etc. By pooling resources, equipment can be better utilized. Greater processing efficiency can be obtained when individual computers are dedicated to a specific type of processing. Any work of that specific type can be routed through the network for processing at the appropriate node.

Introduction to DS/1000-IV

The HP Distributed Systems 1000-IV (DS/1000-IV) is an integrated set of high-level facilities and procedures which support distributed network connections between HP computer systems. These connections can be made over hardwired lines or over common carrier facilities via modems.

DS/1000-IV provides the network user with many advantages over single systems:

* Remote mass storage of, and access to, data and programs to provide increased capability and convenience to networked systems.

* Communication between user-written application programs to provide the capability for synchronization and interaction of programs not located in a single system or geographic area. This will allow you to split processing among several computers (e.g., One side can read raw data from sensors, perform simple validity checks and conversions and then, perhaps compress the data into large blocks for more efficient transmission. The other side could unpack the data and enter it into a data base.) Program-to-Program, PTOP, communication allows users to develop network utility programs for their special applications.

* Sharing of unit-record peripheral devices by more than one program or terminal operator.

* Programmatic scheduling of programs in remote HP 1000 computer systems.

* Remote command processing by terminal operators.

* User program preparation and system generation can be done at a single HP 1000 system and used at target HP 1000 systems elsewhere in the network.

* Store-and-Forward capabilities allow communication and data transfers between any two HP 1000 nodes in the DS/1000-IV network, regardless of network topology.

* Access to interactive or batch-mode programs in other computers without those programs being physically resident in your local node.

The Hewlett-Packard Distributed Systems capability dealt with in this manual is divided into two parts: DS/1000-IV and DS/3000. DS/1000-IV software allows interconnections between HP 1000 Computers with RTE operating systems. The RTE operating system can be an RTE-IVB, RTE-IVE, RTE-MIII, RTE-6/VM, RTE-L, RTE-XL, or RTE-A.

DS/1000-IV software also allows an HP 1000-series computer with any of the operating systems listed above (except RTE-L) to have network communications with an HP 3000 Series II, III, 30, 33, 40, 44, 64 or 68 equipped with DS/3000 software.

This manual describes how to use the communication capability between an HP 1000 computer system and another HP 1000 or an HP 3000. The DS/3000-to-DS/1000 Reference Manual (refer to the documentation guide at the beginning of this manual) describes use of the link from the HP 3000 side.

DS/3000 is a network of HP 3000 Series II, III, 30, 33, 40, 44, 64 or 68 computers with Multiprogramming Executive (MPE) operating systems. DS/3000 is described in the DS/3000 Reference Manual (refer to the Documentation Map).

Structure

When the network is initialized, each system in the network is given a node number and paths between nodes are set up by the Network Manager. The Network Manager uses the program DINIT (described in the Network Manager's Manual) to do this. For example, assume there are seven nodes in the network. There are several ways the nodes can be arranged as shown in Figure 1-1.

Each HP 1000 node in the network can communicate with any other HP 1000 node. An HP 1000 node can communicate with an HP 3000 node only if the HP 1000 has a direct link to the HP 3000. For example, in Figure 1-1D, assume node 7 is an HP 3000. Only nodes 5 and 6 can communicate directly with node 7. Nodes 1, 2, 3, and 4 can put data in a file at node 5 or 6, and then node 5 or 6 can pass the information to node 7. However, nodes 1, 2, 3, and 4 cannot place the information directly into a file at node 7.

The network structure may include DSN/X.25 for interfacing to a Packet Switching Network (PSN). The DSN/X.25 interface protocol has been adopted as an international standard to facilitate global communication. For more information on DSN/X.25, refer to the DSN/X.25 for HP 1000 Computers manual set shown below.

91751-90001	Software Numbering Catalog
91751-90002	Reference Manual
91751-90003	Advanced Design Guide

```
                         A.  STRING

              1     2     3     4     5     6     7
              o-----o-----o-----o-----o-----o-----o


          B.  RING                            C.  STRING

       7             6                       2       7
       o-------------o                       o       o
      /               \                       \     /
     /                 \                       \   /
  1 o                   o 5                     \1/
     \                 /               3  o-----o-----o 6
      \               /                        / \
   2 o------o------o                          /   \
            3      4                         /     \
                                          4 o       o 5


      D.  HIERARCHICAL                      E.  MIXTURE

             7                           1     2     3
             o                           o-----o-----o
            / \                          |     |
           /   \                         |     |
          /     \                        o-----o----------o
     5 o         o 6                     4     5 \       / 6
      / \       / \                               \     /
     /   \     /   \                               \   /
    o     o   o     o                                o
    1     2   3     4                                7
```

Figure 1-1. Network Structures

Store-And-Forward

When one node sends a request to another node, the two nodes do not have to be directly linked together (except in the case of an HP 1000 to HP 3000 link). There may be one or more intervening nodes.

For example, in Figure 1-1C, if node 2 sends a request to node 7, node 1 first receives the request from node 2 and then forwards it to node 7. This process is referred to as Store-and-Forward. In DS/1000-IV this process is transparent since you simply provide the node of the computer system you wish to communicate with without needing to worry about any of the other intermediary nodes.

Software Requirements

DS/1000-IV software product number 91750A is required for full availability of the features described in this manual. A complete list of the contents of DS/1000-IV 91750A (software modules, manuals, software module revision codes, and media options) exists in the DS/1000-IV Software Numbering Catalog (refer to the Documentation Map).

Hardware Requirements

The DS/1000-IV software operates in the HP 1000, M, E, F, and L-Series Computers. At least one Communications Interface Kit (or Card) is required at each node in the network. The Network Manager's Manual (refer to Documentation Map for manual part number) describes the specific hardware requirements for each system in the Distributed Systems Network. Special procedures for system generation, network initialization, and network modification are also presented.

Adding Nodes to Existing Networks

DS/1000-IV 91750A software includes new and enhanced capabilities that were not available in previous HP DS/1000 products. Network nodes equipped with HP 91750A software can fully utilize the capabilities of the software. Message converters in DS/1000-IV software allow nodes with DS/1000-IV software to communicate with network nodes that have pre-DS/1000-IV (91740A/B) software without loss of any existing 91740A/B DS/1000 capability. Therefore, DS/1000-IV nodes may be added to expand existing networks comprised of 91740A/B software.

The "level" of each node in the DS network may be found by displaying the NRV (Nodal Routing Vector) table (see DSINF in the Network Manager's Manual). References to Level 0 in the table equate to pre-DS/1000-IV software (91740A/B) and references to Level 1 equate to DS/1000-IV software (91750A).

Not all of the features available in DS/1000-IV nodes, such as Dynamic Message Rerouting and Message Accounting, can be used when Level 0 nodes are involved. Refer to DS/1000-IV Network Manager's Manual for more specific information.

Addressing and Paths of Communication

There are three specific ways to address a node:

1) node number
2) "neighbor addressing"
3) specify '-1' to indicate the local node

Each HP 1000 node in the network is assigned a node number (address) during initialization. The node number is used to direct a request to a particular node and is always an integer in the range 0 to 32767. In the examples of Figure 1-1, the nodes are numbered from one to seven. They could have been assigned 10, 31, 18, 56, 99, 203, 47, or any other number in the range zero through 32767 as long as each node has a unique number. However, it is recommended that zero be avoided as a node number. Zero is often used as a default value and may cause some interactive and minor programming difficulties.

The second method of addressing, "neighbor addressing", can only be used between neighboring nodes (i.e., two nodes connected by a single link; nodes 1 and 2 in Figure 1-C are neighbors, as are nodes 1 and 3). To address a neighbor, the negative logical unit number of the link between them is used in the nodal address parameter.

The third method of addressing is to give the nodal address as -1. This informs the system to direct the request to the local node. This is useful when developing programs because they can be debugged using the local system before they are loaded into a remote node.

If the node to be addressed is an HP 3000, it must be a neighbor node (there must be no intervening nodes). If it is connected via a BISYNC or HSI link, the BISYNC or HSI LU is used to address the HP 3000. If the HP 3000 is connected via an X.25 link, the X.25 network address is used. In Remote File Access (RFA) calls or RMOTE (HP 1000 to HP 3000 communications) operator commands, an address parameter may be used but is not required (the default is the first BISYNC or HSI HP 3000 LU specified during DS initialization.) In Program-to-Program calls (PTOP) the negative logical unit number of the HP 3000 link is used if the HP 3000 is connected via an HSI or BISYNC link; the X.25 address is used if it connected via X.25.

The possible paths of communication are determined by the topology of the network. In some networks, there may only be one path from one node to another; in others, there may be several. The Network Manager determines the optional routing either by setting up the routing tables, or by making use of the Dynamic Message Rerouting feature in the optional re-routing software.

In Figure 1-1A, node 1 communicates with node 5 by going through nodes 2, 3, and 4. In Figure 1-1B, node 1 can go through nodes 2, 3, and 4 to communicate with node 5, or go through nodes 7 and 6.

Remote I/O Mapping

Remote I/O Mapping is used to "map" or redirect requests destined for an LU pointing from a unit record device on one HP 1000 system to a unit record device on another HP 1000 within a DS/1000-IV Network. It was developed to accommodate terminal-less nodes in a DS/1000-IV network, but can also be used for other applications as will be illustrated later in this section.

At system generation time, certain (EQT, subchannel) pairs (or DVTs in RTE-L, XL and A systems) are established as being mappable. Consequently, any LUs pointing to these mappable entities are mappable. Since you can point LUs to different (EQT, subchannel) pairs (or DVTs) using the system LU command, practically any unit record device may be mappable.

Whenever an I/O request is made to a mappable LU from a local system, this request is "mapped" to a destination LU on a destination system. The particular destination system and LU the request is mapped to is defined by the program IOMAP, which will be discussed later in this section. When the request on the destination system completes, any status, driver type, transmission log, or data read from the destination request are returned to the same local system. In simpler terms, the I/O request made on the local system is effectively mapped into a request on the destination system.

For an example, let us assume we have a node that does not have a printer. It is possible to map a mappable EQT on this particular local node to an LU pointing to a line printer on a destination node. Consequently, any requests made to a local LU pointing to the mappable EQT will be redirected to the LU of the line printer on the destination node.

Another use of Remote I/O Mapping is with "terminal-less" nodes. This may be desired in instances where systems are placed into harsh environments and it may be undesirable, for many reasons, to have terminals directly connected to the computer system. A mappable LU may be created so that any requests made to the system console, such as error reporting, will be instead mapped to a terminal on a remote

Control of the desired mappings is accomplished using the program IOMAP. Supplying the proper parameters allows you to map a local mappable LU to an LU on a destination node which is pointing to a unit record device. This program is usually included as part of the local system's WELCOM file so that the proper LUs will be set-up automatically at system "boot-up" time. This program is not interactive and because of the complexity and network considerations involved, it should be run only by the Network Manager.

For additional details on Remote I/O Mapping, refer to the appropriate chapter in the Network Manager's Manual.

Remote Data Base Access

IMAGE/1000 data bases in the distributed systems network are also accessible through a remote version of QUERY. The remote data base may be accessed either through REMAT or programatically. To access the remote data base with REMAT, you need only switch to the node where the data base resides, log on to a remote session, and access the data base via remote QUERY. The operator commands for accessing the data bases are given in a later chapter of this manual.

Operator Commands

REMAT is a program that allows an operator at any HP 1000 node to communicate with another HP 1000 node. The operator may use any RTE operator commands, some File Manager type commands and additional commands specifically designed for network communication. These commands are described in the REMAT Operator Commands Chapter of this manual.

RMOTE is the program that allows an operator to send commands to an HP 3000. RMOTE connects an HP 1000 terminal as a Virtual Terminal (with some limitations) to the HP 3000. Most MPE operator commands can be used from the HP 1000. Details for using RMOTE are contained in the RMOTE Operator Commands Chapter of this manual.

A description of RTE and MPE commands is not given in this manual. Refer to the appropriate RTE or MPE reference manual for the format and usage of these commands.

Program Calls

The DS/1000-IV software allows you to develop programs that can use the resources of any node in the network. These program calls are similar in format to the corresponding RTE File Manager, RTE EXEC and MPE File System calls. Refer to Chapter 4 through 7 for a description of the program calls.

Remote File Access

The Remote File Access (RFA) calls provide you with the means to manipulate standard and extended files at any node in the network. The RFA calls directed to an HP 1000 node will perform the following functions:

* Create or purge a file
* Open or close a file
* Read from or write to a file
* Position a file forward, backward, or to a record
* Rename a file
* Control a device file
* Get the position of a file
* Get a list of all mounted cartridges

A second set of RFA calls are used for access to files at an HP 3000 node. These calls will perform the following functions:

* Read or write a label
* Read from or write to a file
* Read directly from or write directly to a file
* Open or close a file
* Lock or unlock a file
* Check the status of RFA operations

* Control a file

* Update a file

* Position a file

Details of the RTE and MPE file systems are not given in this manual beyond the simple calling sequence. Consult the appropriate system manual for further information.

Remote EXEC

In an RTE operating system the EXEC calls are used by programs to interact with the system executive. Using similar remote EXEC (DEXEC) calls a program can do the following:

* Schedule another program

* Read from or write to an I/O device

* Control an I/O device

* Terminate a program's execution

* Get the status of a program, an I/O device, or a partition in memory

* Get the time of day

Program-to-Program Communication

The Program-to-Program (PTOP) calls allow a program running at one node to send data or receive data from a program running at another node. The PTOP calls provide a means for direct, immediate communication of information between executing programs.

PTOP calls have the following features:

* One program can communicate with a mixture of remote nodes concurrently.

* Data can be manipulated by the receiving program before storage in a file, or modified by a sending program after retrieval from a file and before transmission.

* Large buffers may be transmitted or received.

* Higher throughput than with RFA calls is realized.

Error Handling

A FORTRAN-callable subroutine, DSERR, can be used during HP 1000 to HP 1000 communication; DSERR will return erroneous results if the remote system is an HP 3000. DSERR returns a 24 word ASCII error buffer to the user. This buffer contains the numeric error code, the Error Code Qualifier and the node number where the error was detected. An example of the error format follows:

```
        DS ERROR:  XXyy(n) REPORTING NODE zzzzz
```

where:

XX = subsystem identifier:

DS	=	DS/1000-IV error
FM	=	FMP error
RS	=	Remote Session error
IO	=	RTE I/O error
SC	=	RTE scheduling error
SM	=	Session Monitor error

yy = the error identifier

n = the error qualifier

zzzzz = the node number

See the appendix on errors for more information.

Dynamic Message Rerouting

Dynamic Message Rerouting capability is available in the DS/1000-IV software and may be optionally installed in each node. When routing between nodes is established at system initialization time, the Network Manager determines route "cost" values and these values are entered into the Nodal Routing Table.

Link costs should be determined by the time it takes to transmit a message between two nodes in the network. Transmission time may be derived from the Network Manager's consideration of such factors as the speed and quality of the line, or by experiment. As a general rule, link costs should be inversely proportional to line speeds (baud rates).

A high-speed, hard-wired, point-to-point transmission path would would generally be more efficient (less costly in terms of time) for good data transmissions than an available lower-speed modem link. The selected route will be the one with the lowest "cost" factor. (Refer to the Network Manager's Manual for a more detailed discussion on route costs.)

Using the network structure of Figure 1-2 as an example, assume that the characteristics of the network indicate that node-1-to-node-4 communications through node 2 has the lowest cost factor. That is, any other routing between nodes 1 and 4 could be more costly in terms of the stated cost criteria; line speed, number of intervening nodes, etc.

Now, suppose that the link between nodes 1 and 2 goes down for some reason, or node 2 is taken off line. The transmission path with the least cost is now unavailable for node 1-2-4 communications. The DS/1000-IV software detects this condition and automatically reroutes node 1-4 messages via an alternate path that is the next most efficient (least costly) path. In the example, with the node 1-2 link down, node 1 messages must now go through node 6 to node 5 (no other choices exist). The software at node 5 may select the node route 5-4 or the node route 5-2-4, depending upon the cost factors of these individual paths.

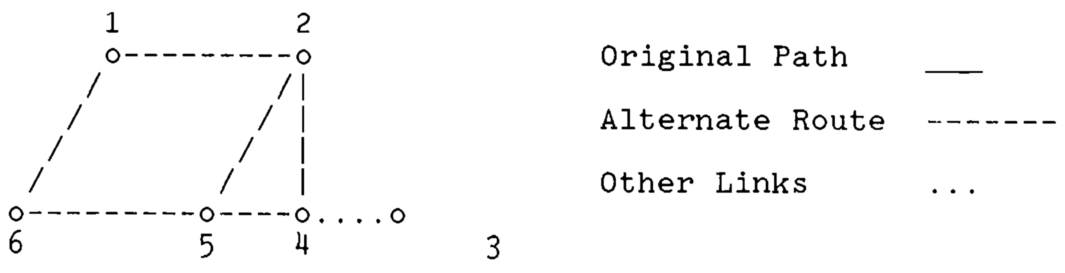


Figure 1-2. Network with Dynamic Message Rerouting

To accomplish dynamic message rerouting, each node in the network must be aware of the network status at all times. Whenever a change in link or node status is detected, each node in the network notifies its neighbor node of the new routing table information, and each node updates its routing tables with the new network status, link LU, and cost factor information. The rerouting algorithm maintains three tables: Nodal Routing Vector (NRV), Link Vector (LV), and Cost Matrix (CM).

The Nodal Routing Vector is used by DS/1000-IV software to route messages to destination nodes. The Link Vector is used to store such information as the link LU (of the communications path), the status of the link (up or down, or whether Dynamic Message Rerouting is enabled or disabled), the "cost" value associated with the link, and the system time used by the Up/Down Counter. The Cost Matrix gives the cost to each destination over all possible outgoing links.

The use of "LU addressing" does not circumvent rerouting. At lower levels in the software, the node number of the neighbor is substituted for the negative link LU; the software may not even use the specified LU, if a better path exists, but the message will be delivered to the correct node if a physical path exists to it.

Dynamic Message Rerouting Considerations

Dynamic Message Rerouting is transparent to the user of the network. It must be noted, however, that dynamic message rerouting can only be implemented in nodes that have DS/1000-IV software installed with the rerouting option. DS/1000 nodes that have 91740A/B software and 12771A or 12773A Serial Interface Kits instead of 91750A software and HDLC Network Interface Cards do not have the capability to reroute messages. They do, however, have capability to store-and-forward messages that have been rerouted through them. DS/1000 nodes which have been upgraded to 91750a software can reroute on link or node failures, but will not be automatically reinstated.

It is important to be aware that dynamic message rerouting is effective only when there is a change in network topology. That is, dynamic message rerouting does not change routing based on link traffic, it only changes routing when a link or node fails.

Message Accounting

The Message Accounting feature is optionally available in the DS/1000-IV software. If the Message Accounting software modules have been generated in, during system initialization the Network Manager may elect to use the message accounting capability. It is an end-to-end protocol that views the network as a series of logical channels connecting end-points, without regard to the logical or physical paths between the points.

Message accounting keeps track of messages generated at a node and generates retransmissions in the event the messages do not reach destination nodes because of line failure, short-term power outages, or other interruptions that might cause non-receipt of the messages at the destination node. Duplicate messages are detected and cancelled, eliminating the duplication of messages that have been received but not acknowledged.

Session Monitor

The Session Monitor features of RTE-IVB and RTE-6/VM are available to the DS/1000-IV user, either on a local or remote basis. Briefly, the features of Session Monitor are:

1) controlled access to the system and its resources on an individual and group basis

2) improved separation between users

Remote Sessions

An RTE-IVB or RTE-6/VM system with Session Monitor maintains controlled access by requiring all users to create a session ("log on") under a predefined account name that has been given specific capabilities by the System Manager. Passwords provide additional protection from illegal use of arbitrary account names.

Remote access to an HP 1000 node that is under Session Monitor control maintains the same level of controlled access. This controlled access should be monitored by the Network Manager who may or may not be the same person as the System Manager. All RFA (Remote File Access), DEXEC, Program-to-Program (PTOP) calls and Operator Commands to that node require a prior "log-on". Remote access may be accomplished via:

1. The utility program DLGON (which creates a specific remote non-interactive session).

2. The REMAT ATTACH (AT) command.

3. The REMAT SWITCH (SW) command with a user-name qualifier appended to the requested node number.

4. The REMOTE 'HELLO' command from an HP 3000.

5. If none of the above have occurred, DS/1000-IV automatically establishes a session under a specific user name specified by the Network Manager for this purpose during network initialization. (This is described in the next section.)

Store-and-forward through a session monitor node does not cause creation of a default session at that node.

Once a session has been created each user is assigned a unique session ID, which is that user's key to access the remote node. You can have a non-interactive session at several HP 1000 nodes concurrently (up to 16), but only one session may exist between any one program and each of up to 16 remote nodes with which it communicates at any one time.

Access to a remote HP 3000 node is accomplished by the HP 3000 REMOTE 'HELLO' command. From the HP 3000 end, you can send a HELLO command to a remote HP 1000 session monitor node to gain non-interactive access to a specific account.

All accessed files must reside on an existing and mounted private, group, or system global disc cartridge. System discs LUs 2 and 3 can be read but not modified.

Users of remote session monitor must consider that the remote session is non-interactive. That is, some features of the system are not available as they would be if the user was interacting with a terminal at the remote node. Specifically, remote session file manager (FMGR) commands to a remote node are not supported.

Default Sessions

A default session is established for existing programs that are not modified to handle remote log-on activity, access for which no prior log-on was made by DLGON or REMAT, and for access by nodes with pre-91750A software.

When a user request arrives at a session monitor node, and this request has not been preceded by a request to create a session, the request causes a session to be created under the default user-name specified during network initialization. All subsequent requests from this user will be directed to this default session. When the user terminates, the system will automatically release the session.

Requests from nodes with pre-91750A software are assigned to a shared default session. Once this particular session is created, the session remains active until the System Manager uses the ACCTS program to shutdown the session. Shared access to this session appears to the user as a non-session monitor system.

There are cases when a default session assignment may not necessarily result in creation of a new session for the default user-name. When a user program issues its first remote request without explicitly creating a remote session at the destination node, the system attempts to find an existing remote session to which the user logically belongs. Attempts are made in the following order:

1) If a father/grandfather/etc. program already has a session at the destination node, default to that session.

2) If any other program scheduled from this terminal has a session at the destination node, default to that session.

3) If the program or an ancestor was scheduled from a session at the destination node or if the session under which this program is running was created from the destination node, default to that session.

Transportability

Many times it is useful to develop a program at one node and run it at another node. Sometimes you may want to take one program, duplicate it, and run it at several different nodes. When a program can be executed at one node or another without change to the source code, it is considered transportable. Due to the possible different environments at various nodes, a transportable program must be processed by the appropriate loader before it can be executed, and those resources upon which it depends must exist at both nodes.

To illustrate transportability, the following example refers to the Hierarchical Network shown in Figure 1-3.

Assume a program is developed at node 10 and stored in a file at the HP 3000. A program at Node 11 uses RFA (Remote File Access) to get the program from the file at the HP 3000. (Note that node 10 cannot just pass the program to node 11 because the only communication line between the two nodes is through the HP 3000 node.) Node 10 then compiles, relocates, and loads the program into nodes 1 and 2. Node 11 compiles, relocates, and loads the program into nodes 3 and 4. When the program is running, it will pass data to the RTE-IVB nodes (nodes 10 and 11).

```
                        (HP 3000 NODE)

                 [31]                        [32]
                  /                            \
                 /                              \
                /                                \
               /                                  \
              /                                    \
         [22]                                      [22]

      (node 10)                                 (node 11)
       RTE-IVB                                   RTE-IVB

      [25]    [26]                          [25]    [26]
       /        \                            /        \
      /          \                          /          \
     /            \                        /            \
    /              \                      /              \
   /                \                    /                \
 [21]               [21]              [21]               [21]

(node 1)          (node 2)         (node 3)          (node 4)
RTE-MIII          RTE-L/XL          RTE-IVE            RTE-A

                                     NOTE:  Numbers enclosed in
                                            brackets, [xx], represent
                                            the Logical Unit assigned
                                            to the communication link.
```

Figure 1-3. Hierarchical Network

One way to make the program transportable is to refer to the RTE-IVB nodes by the negative of the LU number through which they are addressed (-21 in Figure 1-3) instead of their node numbers. Thus, when node 1 sends data to node 10, it uses the negative of the LU (-21) as the nodal address. Nodes 2 through 4 also use 21 as the nodal address of the RTE-IVB nodes (nodes 10 and 11).

Another way is to provide some means for passing the appropriate node number to the program at run-time, such as by parameters or System Common.

If a program was running at nodes 10 and 11, data could be sent to the RTE-MIII nodes by specifying -25 as the nodal address. The RTE-L nodes can be addressed by specifying -26. The HP 3000 node is addressed through LU 22.

The use of logical unit numbers allows the program to be run at each node without changing the source code to reflect where the program is running. This method exploits symmetry in the network topology to reduce programming effort.

As a second example, assume a program is to be transportable and is to share resources (card reader and line printer). The program can be written and developed at, say, node 1. It will read from a card reader at node 10, process the data, and print the results on a lineprinter at node 10. This program is transportable because it will run at node 2 also.

System Initialization

The process for generating, booting up, and initializing each system in the Distributed Systems Network is described in the Network Manager's Manual. The Network Manager's Manual should be read carefully and understood by anyone who is entrusted with these system responsibilities.

Loading User Programs On-Line

User programs that make use of DS/1000-IV facilities (DEXEC, remote file access, program-to-program communication, or the utility subroutines) must be given access to SSGA (labeled System Common in RTE-L, RTE-XL and RTE-A) when they are loaded. Failure to do so will produce a LOADR error (/LOADR: L-SS).

When replacing a user slave program, be sure it is dormant using REMAT's "SO" (Slave Off) command. Its master must not attempt to communicate with it until after the replacement is complete. At that time, it must POPEN the slave again in order to pass the new class number parameter.

If a link to an HP 3000 is present, the version of POPEN used at generation time will be larger than the one required for RTE-to-RTE communication. In the case where programs are loaded which only communicate with other RTE programs, you may use LOADR to search the library which contains the shorter POPEN routine ($DSLB3) before searching the RTE library. You will save at least one page of memory by doing so. If a particular program communicates only with an HP 3000, using it as a slave, you can search $D3KL2 to avoid inclusion of the RTE-to-RTE communications subroutines.

DS/1000-IV facilities do not prevent user programs from being loaded as large background. However, user programs may not be loaded as type 6 (Extended Background) programs on RTE-6/VM, nor may any DS/1000-IV facility (DEXEC, remote file access, program-to-program communication, or the utility subroutines) be called from a type 6 program.

On RTE-A systems with multiuser environment, programs that are controlled by PTOP master calls, or by DEXEC 9, 10, 23, 24, 12 or 99 calls, must be loaded as system utilities.

Segmented Programs

Segmented programs must contain the modules #NAT, #MAST, #SLAV, D3KMS and HELLO in the main program if they are used. These modules may be accessed using the following entry points:

MODULE	ENTRY POINT
#NAT	#NAT
#MAST	#MAST
#SLAV	#SLAV
D3KMS	D3KMS
HELLO	HELLO/BYE

Refer to the reference manual of the loader you are using for more information on how to control module placement.

Segmented PTOP Master Programs

If a PTOP master program is segmented, you must call the PTOP "POPEN" call from the main program. If the master program is communicating with a slave program on an HP 3000, you must call both "POPEN" and the utility call "HELLO" from the main program.

Chapter 2
Remat Operator Commands

Introduction

The operator has two programs available for use in sending commands to other nodes in the network: REMAT and RMOTE. The REMAT program is used in sending RTE commands or special DS/1000-IV commands to any RTE node. The RMOTE program is used in sending MPE commands or special DS/1000-IV commands to an HP 3000. The RMOTE operator commands are contained in Chapter 3 of this manual.

Command Syntax

There are several parameters associated with each command. Some of these parameters are required, while others are optional. Parameters shown inside brackets are optional while those outside the brackets are required. If you do not use an optional parameter, but do use following parameters, use a comma as a place holder. When you do this, the parameter takes a default value. Conventions used in this manual for illustrating command syntax are shown in Table 2-1.

Table 2-1. Command Syntax

UPPER-CASE LETTERS	Literal values that must be specified exactly as shown.
lower-case letters	Type of information to be supplied by the user; most parameters are in this form.
[,parameter]	Optional parameters are enclosed in brackets; the system supplies a default value if omitted.
,parameter1 ,parameter2	One and only one of the stacked parameters must be specified.
[,parameter1] [,parameter2] [,parameter3]	All bracketed parameters are optional; only one may be specified. If all are omitted, the system supplies a default value.
[,param1[,param2]]	Series of optional parameters; the last parameter may be omitted without indication; imbedded parameters must be indicated by a comma when omitted.
...	Ellipses indicate that the previous parameter or series of bracketed parameters can be repeated.

The REMAT Program

REMAT is a program that is used to direct operator commands to any RTE node in the network. When the commands are directed to the local node, REMAT displays a dollar sign ($) prompt. When the commands are directed to a remote node or the results of the commands are to a remote node, REMAT displays a pound sign (#) prompt. When the command is entered, REMAT scans its own list of operator commands first. If the operation code of the command is not in the REMAT list, REMAT assumes it is an RTE operator command and forwards it to the operating system.

All REMAT commands, with the exception of BC, EX, LC, LL, and TR, are directed to a specific node with the SWitch command. Until the SWitch command is given, all commands are directed to the local node.

REMAT-command access is provided to a remote node that is under Session Monitor control. REMAT commands that access specific cartridge resources and Opcommand (Operator Command) requests are directed to the non-interactive "session" created by the SW or AT commands. For this reason, some of the REMAT commands will be limited to the particular capability level of the session account created by the AT and SW command.

If SW or AT are not used to create a session, any REMAT commands will be directed to a default session which was specified during network initialization. The REMAT user can use the SW command with no parameters to determine whether a default session was created.

A user can access up to 16 nodes concurrently and each node can have a single non-interactive session created on behalf of the user.

Most messages that result from a command sent to a remote node are returned to the node that originated the command.

```
********
* NOTE *
********
```

Physical node numbers in this chapter are designated by <node xx>, where xx is the node identifier assigned by the Network Manager. Since the SWitch command allows you to logically operate at any node in the network, it is necessary to be able to specify that information uniquely. The designation for your logical location is NODE1 (the origin node) and NODE2 (the destination node).

REMAT is scheduled with the RTE ON or RU command.

FORMAT

```
             [,namr[,log[,list[,severity code]]]]
    RU,REMAT
             [,input[,log[,list[,severity code]]]]
```

For RTE-MIII or IVE, the format to read from a file is:

```
    RU,REMAT,fi,le,nm[,list[,severity code]]
```

namr	A file name and optionally security code and cartridge identifier which provides all input commands. All commands must be proceeded by the '$' prompt.
input	The LU of the system input device. Default is Session LU 1.
log	The LU of the interactive message logging device. Default input LU (if interactive) or the value returned by LOGLU.
list	The LU of the list device. Default is LOGLU.
severity code	The error reporting code. 0 - echo all commands and report all errors (default). 1 - inhibit command echo.
fi,le,nm	The name of the file that contains the input commands. All REMAT commands within the file must be preceeded by the '$' prompt.

REMAT may also be scheduled from a program with DEXEC calls.

FORMAT

```
CALL DEXEC(IDEST,ICODE,REMAT,INPUT,LOG,LIST,SEVERITY CODE)
```

REMAT can also be scheduled programmatically with a command file.

FORMAT

For RTE-IVB, RTE-IVE, RTE-6/VM, RTE-L, RTE-XL, AND RTE-A:

```
CALL DEXEC(IDEST,ICODE,REMAT,IP1,IP2,IP3,IP4,IP5,IBUFF,ILEN)
```

IBUFF	Contains the command file namr in the form '8H,,TRFILE'
ILEN	The length (positive words or negative characters) of IBUFF

For RTE-MIII

```
CALL DEXEC(IDEST,ICODE,REMAT,2HFI,2HLE,2HNM,LIST,SEVERITY  CODE)
```

IDEST The node number where REMAT is to be scheduled.

ICODE The request code.

9	to schedule REMAT with wait
10	to schedule REMAT without wait
23	queued schedule with wait
24	queued schedule without wait

REMAT A three word array containing the ASCII characters "REMAT".

The other parameters correspond exactly to the parameters in the RU,REMAT command.

NAMR

A special parameter, namr, is used to identify a file or device. It uses subparameters to specify the security code (if one exists) for a file and the cartridge on which the file resides. The namr parameter is the standard namr used by RTE File Manager, and is used only for FMGR files.

FORMAT

```
namr =
  filename[:security code[:crn[:file type[:file size[:record size]]]]]
                                \                                     /
                                 \----------------\/-----------------/
    or                                   file creation only


namr = logical unit number
```

Some commands require the filename format, while others will accept either an LU or a file name. Consult the particular command description for specifics.

Unless specifically noted, each subparameter has a default value of zero. This value is selected so that, as closely as can be predicted, it provides for the most general case. This means that in many cases all subparameters can be omitted and the file will be completely specified by name alone.

SUBPARAMETERS

filename

Six character ASCII file name; restricted as follows:

* only printable characters, space through _
* plus (+), minus (-), colon (:) or comma (,) are not allowed
* first character must not be blank (space) or a number
* embedded blanks are not allowed
* must be unique to the disc

security

File security code: positive or negative integer or two ASCII characters representing a positive integer; range is from -32767 through 32767; security code may be:

zero	file is unprotected (default).
+integer	protected against alteration (write protected) or purging; may be read with any security code or none; may be altered or purged only with correct or negative (2's complement) of correct code.
-integer	file is fully protected (read and write protected); may only be referenced with correct negative code.

crn

Cartridge identifier: positive or negative integer or two ASCII characters represented as a positive integer; range is from -32767 through 32767; used to identify an FMP disc; it may be:

zero	first available disc that satisfies the request is used (default).
+integer	cartridge reference number by which the disc is identified.
-integer	logical unit number associated with the disc.

file type

Positive integer in range 0 through 32767: default depends upon command.

0	non-disc file
1	fixed length 128-word record
2	fixed length records; user defines length
3	variable length records; sequential access, automatic extents
4	ASCII code and source program (otherwise like type 3 files)

5	relocatable binary code (otherwise like type 3 files)
6	save program file (otherwise like type 1 files)
7	absolute binary (otherwise like type 3 files)
8-32767	user defined

file size

Decimal number of blocks in range of 1 through 32767: a block is 128 words; indicates space allocated to file. The minimum is 1.

record size

Decimal number of words in range 1 through 32767: applies only to type 2 files; type 1 files use 128-word records, other types use variable length records.

logical unit number

Positive integer specifying logical unit number of non-disc device.

REMAT Commands

The REMAT commands are similar to the RTE File Manager commands. Eight commands control REMAT: ATtach, EXit, DEtach, list local node, change list/log device, SWitch and TRansfer. The other commands control program execution, files and directories. Table 2-2 briefly describes each command.

NOTE: REMAT returns only the first 40 characters of system messages. Longer messages, from RTE-A systems, are truncated.

Table 2-2. REMAT Commands

Command	Description
AT	Attach to remote HP 1000 account
BC	Broadcasts a message to all connected nodes
CL	Lists the cartridge directory
CR	Creates a file
DE	Detach from a remote HP 1000 account
DL	Lists a file directory
DU	Dumps from a file or LU to an LU
EX	Exits REMAT
FL	Closes a disc file
IO	Lists system I/O configuration (RTE-L/XL/A only)
LC	Displays local nodal address
LI	Lists a file
LL	Changes or displays the list and/or log devices
LO	Loads a program (RTE-MIII/L/XL/A only)
PL	Program list (RTE-MIII/L/XL/A only)
PU	Purges a file
QU	Queue schedules a program without wait
QW	Queue schedules a program with wait
RN	Renames a file
RW	Schedules a program with wait

Table 2-2. REMAT Commands (Continued)

Command	Description
SD	Shuts down a remote session
SL	Lists slave programs
SO	Terminates slave program
ST	Creates and stores a file
SW	Switches (or displays) destination of commands
TE	Sends operator message
TR	Transfers Control

Any other commands issued to REMAT are passed to the operating system for processing. If the operating system does not recognize the command, an 'OP CODE ERROR' message is returned.

AT — Attach

Attaches to a remote HP 1000 account. Do not use this command if you want the default account, if the remote node does not have session, or if the remote node is an RTE-A system.

FORMAT

```
     AT,[user.group[/password]][,user.group[/password]]

               (NODE1)                 (NODE2)
```

user.group/password	The account and optional password in the corresponding node for which a non-interactive session is to be created and to which subsequent REMAT commands will non-interactively attach. Previous session at this node will be released. For non-session access, this parameter is the password only. User, group, and password may each be up to 10 characters in length. (Must be specified at either NODE1 or NODE2.) This command is illegal if directed to the local node.

The ATtach command can be used instead of specifying an account name in the SWitch command:

```
 #SW,9,,sc                                #SW,9:user.group,,sc
 #AT,user.group        equals             #
 #
```

where:

    sc = the network user's security code

The ability to change accounts within the same node is very useful.

Up to 16 remote nodes may be concurrently accessed under non-interactive session per user. If you attempt to access more than 16 remote nodes at once, an RS03(0) error, Exceeded Local Limit for Remote Sessions, is returned.

Referenced remote files must reside on a private or group cartridge mounted to the specified account or on a globally mounted (system) disc.

If you wish to gain non-session access, the AT command may be issued in the form:

```
          #AT,[/password][,/password]

                (NODE1)    (NODE2)
```

The passwords are those set up for non-session access at each node by the Network Manager. If no password is required, enter: #AT,/

BC — Broadcast

Broadcast a message to all nodes. (Same as the RTE system command TELLOP, but message is sent to all nodes enabled at network initialization.)

FORMAT

```
      BC, message
```

message	= An ASCII string of up to 72 characters in length.

The message is sent to the system console (system LU 1) of each remote node, one at a time. Before sending the message to a node, the message

```
     BROADCASTING TO NODE # <node number>
```

is printed. If an error occurs, it may be one of the following types.

Error	Sub-parameter	Cause of Error
DS05:	(1)	Terminal at remote node is tied up, waiting for input. In this case, the message will eventually be printed, as soon as the terminal becomes available.
	(2)	Any of the other causes for a master time-out error.
DS01/DS02:		Link to remote node is not connected or remote node not enabled for communication.

In the following example, node 2 sends the message:

"BROADCAST" TEST MESSAGE

to all nodes which are enabled (i.e., 1, 3, 4, 10, 20, and 30). REMAT returns error messages for all except nodes 1 and 4:

```
:RU,REMAT
$SW,1,,sc
#BC,"BROADCAST" TEST MESSAGE
BROADCASTING TO NODE#     1
BROADCASTING TO NODE#     3
REMAT:  DS01
BROADCASTING TO NODE#    20
REMAT:  DS05
BROADCASTING TO NODE#    30
REMAT:  DS01
BROADCASTING TO NODE#    10
REMAT:  DS01
BROADCASTING TO NODE#     4
```

where: sc = the network user's security code

The following appears at the system console of each node that receives the message:

```
=N    2:  "BROADCAST" TEST MESSAGE
```

where:

=N   2 = the node number which sent the message (in this case node 2)

CL — Cartridge List

Lists mounted cartridges at the currently switched node (NODE1) on a local list LU. If there is a session at the remote node, only those cartridges available to the account to which you are attached will be listed.

FORMAT

```
CL[,lu]
```

lu	the logical unit number for the local list device. It is defaulted to the current list device set up by LL command, or the listlu assigned by REMAT.

For example:

```
:RU,REMAT
$SW,1,,DS
#CL
  LU   LAST TRACK    CR     LOCK  REMOTE NODE= 00001

  02      00202      00002
  03      00202      00003
  24      00202      01000
  25      00202      01001
  26      00202      00050
  27      00202      32767
  28      00202      01905
```

CR — Create a File

Creates a disc file on the currently switched node (NODE1). No data is transferred to the file.

FORMAT

```
CR,namr
```

namr	The file descriptor: Must not be a logical unit number, omitted parameters default to zero. File type and file size must be specified as greater than zero. Refer to namr description at the beginning of this chapter.

Refer to the appropriate Programmer's Reference Manual for the types of files that can be created.

For example:

`#CR,MYFILE:-25:100:4:10`	MYFILE is type 4, has a security code of -25, is created on cartridge 100, and has 10 blocks of disc space.
`#CR,URFILE:::2:20:72`	URFILE is type 2, uses 20 blocks and each record is 72 words.

DE — Detach

Detach from a remote HP 1000 account.

FORMAT

```
DE[,N1][,N2]
```

N1 specifies the current NODE1

N2 specifies the current NODE2

There are variations in the use of this command that permit the user to detach from either or both nodes.

```
#DE             detaches from both NODE1 and NODE2.
#DE,N1          detaches from NODE1 only.
#DE,N2          detaches from NODE2 only.
#DE,N1,N2       detaches from both NODE1 and NODE2.
```

The DE command will not, of course, log you off your local session account.

Once a DE command is issued, any further commands to this node without a prior ATtach command will be executed under the default session.

It is not usually necessary to use this command since REMAT will automatically log you off any remaining active remote sessions when you exit REMAT. It is useful if you are finished with a remote session and do not want to exit REMAT to release that connection in order to have access to another node.

DL — Directory List

Lists the file directory of the currently switched node (NODE1) on the local list device.

FORMAT

```
       [,cartridge[,master security code[,lu]]]
   DL      or
       ,namr[,master security code[,lu]]
```

cartridge	Cartridge identifier: must be numeric, positive for cartridge reference number, negative for logical unit number; if omitted or zero, the directories of all cartridges mounted to the default or specific account are listed.
master security code	The FMP master security code. If specified correctly the security code of each file will be listed.
namr	File descriptor: minus signs (-) may be used in the file name to specify a match with any single character.
lu	The logical unit number of the local list device: If specified, the directory will be listed on this lu. Default is the list device specified when REMAT was scheduled, or with the LL command.

DL lists a file directory that resides at NODE1. NODE1 is specified in the SW command.

A remote directory list can be performed three ways (assume that SW was used and NODE1 is a remote node which has Session Monitor):

a. Take default session:

```
#DL
```

Lists only files residing on cartridges mounted to the default user name specified during network initialization, including the default user's private or group cartridge or system discs.

b. Non-interactively attach to remote account:

```
#AT,user.group
#DL
```

Lists only files residing on those cartridges that are mounted to the user's private or group cartridge or system discs.

c. Gain non-session access:

```
#AT,/password
#DL
```

Lists only non-session discs.

Using a filter allows you to selectively list the directory. When this format is used the following conditions must be met before a given file entry will be listed:

a. The file name must match the name portion of namr except that a minus sign, "-", used in namr "matches" any character.

b. Zero as a subparameter matches any actual subparameter. If a non-zero subparameter is used, it must match the files' actual parameter.

c. The standard security code match is used, i.e.;

   -n matches n and -n
   +n matches n only

For example:

```
#DL,A-----:::3
```

Lists all file names of six characters or less that begin with an 'A' and are type 3 files.

```
#DL,FI--
```

Lists all files of four characters or less that begin with 'FI'. No file type filter is given.

The format of a directory list is shown in the following example:

```
    ILAB= SPOOL    REMOTE DLIST:  NODE=  00001   CR#=  00050

    NAME      TYPE      #BLKS/LU     OPEN TO      SCODE
    JOBFIL    00002     00004         SMP          00000
    SPLCON    00002     00005         SMP         -00032
    SPOL02    00003     00024                      00012
    SPOL03    00003     00024                     -32767
    SPOL04    00003     00024                     -00342
    SPOL05    00003     00024                      00000
    &RUNOF    00003     00049         RFAM
```

The SCODE column will be blank if the master security code is not specified in the command.

The DL command also lists file extent numbers and the name of the program to which a file is open. If that name is RFAM, then the file is open for access by one or more programs at NODE1 or at remote nodes. If more than one program is sharing the file, only the first program is shown.

DU — Dump

Transfers data from a file or logical unit to a logical unit.

FORMAT

```
DU,namr,lu[,format]
```

namr	The file or logical unit from which the records will be dumped. (Refer to namr description at the beginning of this chapter.)
lu	The logical unit to which the records will be dumped.
format	Format of the data being transferred. Default is derived from namr, if namr is a file, otherwise default is ASCII. The format may be:

AS	ASCII
BR	Binary relocatable; checksum is performed
BN	Binary; no checksum is performed
BA	Binary absolute; checksum is performed

DU transfers data from a file or logical unit at NODE1 to a logical unit at NODE2. NODE1 and NODE2 are specified by the SWitch command.

If namr is a terminal, a slash (/) prompt is output to signal readiness for the next line. For systems using long inter-computer cables or modems or if the system is very busy, there may be a perceptible delay while the previous line is transmitted. The next input should not be made until REMAT is ready. An end-of-file mark is made by typing control-D. (Pressing the 'D' key while holding down the control key.)

For example:

#DU,FILE,7	Dump the contents of FILE to LU 7.
#DU,4,6	Dump from LU 4 to LU 6.
#DU,%FILE:sc,4,BR	Dump the file %FILE which has a security code of sc to LU 4 in binary relocatable format.

EX — Exit

Terminates REMAT.

FORMAT

```
      EX
```

When the EX command is entered, any sessions created by this REMAT, and still active at remote nodes, are logged off.

```
********
* NOTE *
********
```

It is also possible to interrupt the current operation of REMAT using the BReak command from the RTE breakmode prompt. It is recommended that this command be used to break a REMAT operation rather than the OF,REMAT,1 command. Use of the OF command does not clean up system resources. BR is only effective with the REMAT commands DL, DU, LI, ST, and TR.

FL — Close a Disc File

Closes a disc file.

FORMAT

```
      FL,namr,node
```

namr	The file descriptor: The crn subparameter of namr must be supplied and be non-zero. Refer to namr description at the beginning of this chapter.
node	The node number at which the user to whom the file is being closed resides. If node= -1, the file is closed to all users.

This command flushes a table entry for a file which exists at NODE1 as specified in the SWitch command. This effectively closes the file to the users at the nodes specified.

When a file is opened, an entry is made in an internal table specifying which files are open, to whom they are open, and at which node the user/program is located. If a program is aborted or fails to close a file, then files that were open to it remain open. If the open was exclusive, no other program can use the file. This command is provided to give the operator the ability to remotely close a file that may have been inadvertently left open by a remote user/program. This is an extremely powerful command in that it will facilitate the closing of files which were not open to the operator. The file is closed and the proper entry in the internal table is deleted.

To ensure that the operator does not inadvertently close the wrong file, this command is only allowed from an interactive input device. The following is asked:

```
  CLOSE <FILENM> AT NODE XX TO USERS AT NODE YY?

                    or

  CLOSE <FILENM> AT NODE XX TO USERS AT ALL NODES?
```

The operator responds with YES or NO. If NO is typed, the command is ignored. If YES is typed, the file is closed and REMAT prints a message indicating the number of programs that were running but left the file open.

For example:

```
   #FL,MYFILE::20,-1                    Close MYFILE on cartridge 20 at
                                        NODE1 to all users.

   CLOSE MYFILE AT NODE 10 TO USERS AT ALL NODES?

   YES

   #RFAM ENTRIES FLUSHED = 1            One program/user left the file
                                        open.

   #FL,MYFILE::-43,10                   Close MYFILE on LU 43 to users
                                        at node number 10.

   CLOSE MYFILE AT NODE 10 TO USERS AT NODE 10?

   YES

   #RFAM ENTRIES FLUSHED = 2            Two programs/users left the
                                        file open.
```

IO — List I/O Configuration

Lists the system I/O configuration from an RTE-L/XL/A system.

FORMAT

```
      IO
```

When the IO command is entered, a list of the RTE-L, XL or A system I/O configuration from NODE1 (as specified in the SW command) is listed on the list LU at the local node. If NODE1 is any system other than an RTE-L, XL or A, an error will be returned.

For example:

```
     #IO


  LU     DVT  S.C.    DP#1  PRIO  DVT ADR D.TYPE-  DEVICE     I.TYPE  LU
   1      1    20        1   77      30735    05  KEYBD CTL DV    00   1
   2     --------------------- LU UNASSIGNED --------------------      2
   3     --------------------- LU UNASSIGNED --------------------      3
   4      3    20        2   77      31026    20  SER RECORDNG    00   4
                                       .
                                       .
                                       .
                                       .
```

NOTE: If you use this command on RTE-L/XL/A systems and NODE1 is set to your local node, REMAT sends the reply to system lu 1.

LC — Local Node

Displays the local node number.

FORMAT

```
      LC
```

LC allows you to determine the node number at which you are currently physically working. This node number is assigned at network initialization time by the Network Manager using DINIT. (DINIT is described in the Network Manager's Manual.)

The node number is displayed at the log device.

Example:

```
        #LC

        LOCAL NODE  =  10
```

LI — List

Prints the contents of a file on a logical unit.

FORMAT

```
      LI,namr[,lu]
```

namr =	The file descriptor: If the file is protected by a negative security code, it must be specified. Refer to namr description at the beginning of this chapter.
lu =	The optional logical unit where the file is to be listed.

LI lists a file, which exists at NODE1, on a logical unit at NODE2. NODE1 and NODE2 are specified by the SWitch command.

For example:

```
   #LI,FILENM,6

   FILENM  TYPE: 3   NUMBER OF BLOCKS:  12   LOCATED AT NODE:  123

   00001 THIS IS THE FIRST LINE OF THE FILE
   00002 THIS IS THE SECOND LINE OF THE FILE
   00003 THIS IS THE THIRD LINE OF THE FILE
   00004 THIS IS THE FOURTH LINE OF THE FILE
   00005 THIS IS THE FIFTH LINE OF THE FILE
   00006 THIS IS THE SIXTH LINE OF THE FILE
   00007 THIS IS THE SEVENTH LINE OF THE FILE
   00008 THIS IS THE EIGHTH LINE OF THE FILE
   00009 THIS IS THE NINTH LINE OF THE FILE
   00010 THIS IS THE TENTH LINE OF THE FILE
```

LL — Change List LU (Logical Unit)

Changes or displays the LU of the list and/or log device.

FORMAT

```
      LL[,list[,log]]
```

list	The logical unit number of the new list device.
log	The logical unit number of the new log device.

LL allows you to change the current local list and/or log devices that were set when REMAT was scheduled. If both parameters in the LL command are omitted, the current values of the list and log devices are displayed on the log device. The default list and log LU is the LU of the user terminal.

For example:

```
     #LL
     LIST LU =     6         LOG LU =    1
     #LL,1,8
     #LL
     LIST LU =     1         LOG LU =    8
```

LO — Load a Program

Loads a program into an RTE-MIII or memory-based RTE-L/XL/A system.

FORMAT

```
   MIII-System: LO[,namr[,partition[,pages]]]

   L/XL/A-System: LO,namr
```

namr	The name of the file or logical unit which contains the absolute program to be loaded. The first five characters of the file name are used as the name of the program in building the program's ID segment. If an lu is specified, the program name is taken from the NAM record. Default is LU4 in RTE-MIII systems. In RTE-L/XL/A systems, namr is not optional and loading from an LU is illegal. In session nodes, the file must reside on a system cartridge.
partition	The partition number in which the program is to execute. This parameter is only used in RTE-MIII.
pages	The smallest partition (specified by the number of pages) in which the program can execute. This parameter is only used in RTE-MII. Two commas must separate the namr from pages if the partition parameter is omitted.

LO loads an absolute program file at NODE1 into an RTE-MIII or a memory-based RTE-L/XL/A system at NODE2. NODE1 and NODE2 are specified in the SWitch command.

When either partition number or number of pages is entered, the program is loaded into a partition. When both of these parameters are omitted, a non-partition load is performed.

This request is illegal when sent to a system other than an RTE-MIII or non-memory based RTE-L/XL/A.

If the LO command is entered at a node other than an RTE-MIII and namr is specified as (or defaulted to) a logical unit, a local type-0 file describing this logical unit must have been previously created. The name of the file must be LU..xx, where xx is the 2-digit logical unit number (eg., LU..04).

Since RTE-MIII and memory-based L, XL and A systems use only the first five characters for the program name, you can use the sixth character to indicate different versions of a program.

For example:

`#LO,PRGNM1::20,,15`	Load PRGNM1 from cartridge 20 at NODE1 to a partition of at least 15 pages at NODE2. PRGNM1 is run with the name PRGNM. (RTE-MIII)
`#LO,PRGNM2`	Load PRGNM2. The file name is PRGNM2, the program name is PRGNM. (RTE-MIII or RTE-L/XL/A)

For RTE-L/XL/A systems, this command only works if APLDR was loaded from %APLDL (the full-function version of APLDR).

If %APLDX (the reduced version of APLDR) was loaded, then you must use APLDR and RPRTL to load programs. See the On-Line Loaders chapter of this manual for information on these programs.

NOTE: If you use this command on RTE-L/XL/A systems and NODE1 is set to your local node, REMAT sends the reply to system lu 1.

PL — Program List

Lists all programs in an RTE-MIII, L, XL, or A system and reports how many ID segments are available for use.

FORMAT

```
        MIII-System: PL[,list lu[,option]]

      L/XL/A-System: PL[,option]
```

list lu	The logical unit number of the local list device. Default is the current list device specified in the LL command or the list lu assigned by REMAT.
option	List option. In an RTE-MIII system option is an integer: If option=0, all programs, priorities and bounds are listed (default value). If option=1, partitions and their programs, partition size and page numbers are listed. Option can equal 1 only in an RTE-MIII system. In RTE-L, XL and A systems, option is in ASCII. Option may be one of the program status codes described under the command PL (list programs) in the User's Manual for the appropriate operating system. Only those programs with the specified status will be listed. If no option is specified, all programs, their status, priority, and point of suspension are listed.

This command is the standard RTE-MIII/L/XL/A 'PL' command. The format of the list is identical to that of the standard APLDR. PL is illegal when sent to other than an RTE-MIII/L/XL/A system. An example is shown below:

#PL,6,1	List on LU 6 the partitions, programs partition size and page number.

NOTE: If you use this command on RTE-L/XL/A systems and NODE1 is set to your local node, REMAT sends the reply to system lu 1.

PU — Purge

Removes a file from the disc.

FORMAT

```
PU,namr
```

namr File descriptor: file name and optional security code and crn. Refer to namr description at the beginning of this chapter.

The file (at NODE1) and all its extents are removed from the system. Type-0 files cannot be purged by a remote operator.

For example:

#PU,FILENM	Purge the first file found at NODE1 with name FILENM
#PU,FILENM::30	Purge FILENM on cartridge 30

QU - Queue Schedule Without Wait

Queue schedules a program to run without wait.

FORMAT

```
        QU,pname[,p1[,p2[,p3[,p4[,p5]]]]]

        or

        QU,pname[,string]
```

pname	The name of the program to be scheduled.
p1,...,p5	Up to five optional parameters to be passed to the program.
string	ASCII string to be passed to the program. The command line, including the '$' prompt if commands are being read from a non-interactive device or file, must not exceed 80 characters.

The QU command causes REMAT to queue schedule the requested program. If the requested program is known to the system at NODE1 (as defined by the SW command) but is not currently available, REMAT waits until it becomes available and schedules it. REMAT does not wait for the requested program to terminate, however, and returns to its command file or terminal for another command. This is equivalent to a DEXEC 24 request.

See the section on the RU command for additional information. For information on timing and other constraints, see the section on DEXEC 9, 10, 23 and 24 in Chapter 5.

QW - Queue Schedule With Wait

Queue schedules a program to run with wait.

FORMAT

```
QW,pname[,p1[,p2[,p3[,p4[,p5]]]]]

or

QW,pname[,string]
```

pname	The name of the program to be scheduled.
p1,...,p5	Up to five optional parameters to be passed to the program.
string	ASCII string to be passed to the program. The command line, including the '$' prompt if commands are being read from a non-interactive device or file, must not exceed 80 characters.

The QW command causes REMAT to queue schedule the requested program with wait. If the requested program is known to the system at NODE1 (as defined by the SW command) but is currently not available, REMAT waits until it is available and schedules it. REMAT then waits until the program completes before returning to the command file or terminal for another command. QW is equivalent to a DEXEC 23 request.

See the section on the RU command for additional information. For information on timing and other constraints, see the section on DEXEC 9, 10, 23 and 24 in Chapter 5.

RN - Rename

Changes a file name to a new name. None of the file characteristics are changed except the name.

FORMAT

```
RN,namr,nuname
```

namr	File name, security code (if one exists), and crn of an existing file. Refer to namr description at the beginning of this chapter.
nuname	New file name unique to the disc cartridge. Security code and cartridge identifier can not be altered.

RN renames a file which exists at NODE1 as specified in the SWitch command.

If the file has a non-zero security code, the proper security code must be included. If crn is specified, only that cartridge is searched for the existing file; otherwise all cartridges mounted to the account are searched and the first file found with the correct name is renamed.

RW — Run With Wait

Schedules a program to run with wait.

FORMAT

```
      RW,pname[,p1[,p2[,p3[,p4[,p5]]]]]

      or

      RW,pname[,string]
```

pname	The name of the program to be scheduled.
p1,...,p5	Up to five optional parameters to be passed to the program.
string	ASCII string to be passed to the program. The command line, including the '$' prompt if commands are being read from a non-interactive device or file, must not exceed 80 characters.

The RW command schedules-with-wait a program located at the node currently defined by the SW command as NODE1. This request causes the named program to be duplicated and renamed (cloned) when the destination node is an RTE-IVB or RTE-6/VM remote node with Session Monitor.

If the scheduled program passes back return parameters upon completion, these will be displayed in octal and in ASCII on the log LU. Return parameters, if any, will be displayed in the following format:

```
    xxxxxx xxxxxx xxxxxx xxxxxx xxxxxx  aabbccddee
```

where xxxxxx represents the octal and aabbccddee the ASCII representation of the return parameters. If the program was aborted, the first set of xxxxxx is 100000.

This command is used (especially when REMAT is running under a session) to schedule and interact with remote programs such as EDITR, QUERY, and DSINF. Any remote program scheduled with wait (so that the node number from which the request originated is saved) that does its interactive dialogue through DEXEC I/O calls to the originating node will have its I/O mapped back to the originating session (if any). When the user is operating in a local session, the remote programs do interactive I/O with session LU 1.

This command is also very useful when you wish to run a local program that does not call DLGON but want to have it access a specific remote session. Whenever a program is run locally as a "son" of REMAT by scheduling the local program with the RW command, the son's requests, by default, will be directed to the session created by REMAT. See the section on default sessions in the General Description Chapter of this manual.

For example, if your local node is 20 and you wish a program, PROGA, to access a specific session at node 40, the following sequence of commands may be given:

```
:RU,REMAT
$SW,20,40:HP.GENERAL/SD,sc
#SW,LO
$RW,PROGA
```

where:

sc = the network user's security code

This example shows REMAT scheduling a local program which does not itself do a remote log-on but is assigned by default to the specific remote session created by the REMAT SW or AT command.

See the section on DEXEC 9,10,23 and 24 in Chapter 5 for information on timing and other constraints for programs remotely scheduled with wait.

> **NOTE**
>
> The RW command should be used to schedule interactive programs from REMAT. If an interactive program is scheduled with the ON or RU command, then REMAT does not wait for the program to finish executing. Any input from the terminal is sent either to REMAT or to the interactive program depending on which program issues a read request first. This causes confusion and errors.

SD — Shut Down A Session

Shuts down a session created by DS at NODE1.

FORMAT

```
    SD,session ID,NMSC
```

Session ID	The session ID at NODE1 that you wish to shut down. (Your session ID can be found by using the DSINF RS command.)
NMSC	The Network Management Security Code at the node where REMAT is running.

SD removes the session from the DS table that describes sessions created at NODE1 (the POOL), and logs the session off. The SD command may be issued from any node in the network.

```
+-----------------------------------------------------------------+
|           ***************************************                |
|           *             WARNING                 *                |
|           ***************************************                |
|                                                                  |
|   The SD command only releases DS tables at NODE1, not           |
|   at the node which created the session.  Its purpose            |
|   is to clean up when catastrophic events (such as               |
|   rebooting) occur at the creating node, causing                 |
|   discrepancies between the creator's PNL and NODE1's            |
|   POOL. DO NOT SHUT DOWN A SESSION WITH THE SD COMMAND           |
|   FOR ANY OTHER REASON.                                          |
+-----------------------------------------------------------------+
```

SL — Slave List

Lists all Program-to-Program (PTOP) slave programs.

FORMAT

```
SL[,list lu]
```

list lu	The logical unit number of the local list device. Default is the list device specified in the LL command or the listlu assigned by REMAT.

SL lists all programs currently executing at NODE1 (specified in the SW command) under slave Program-to-Program control. The format of the slave list is shown in the following example:

```
ACTIVE SLAVE PROGS

PROGA
PROGB
NEXTP
ABCPR
```

SO — Slave Off

Terminates a PTOP slave program.

FORMAT

```
SO[,program name]
```

program name	The program to be terminated. If no program is specified, ALL current PTOP slaves are terminated at NODE1. NODE1 is specified in the SWitch command.

If the program that is specified is not currently using PTOP as a slave, no action will result.

ST - Store

Transfers data and creates a disc file.

FORMAT

```
      ST,namr1,namr2[,format[,mode]]
```

namr1	File descriptor of existing file or logical unit number. Data is transferred from namr1. Refer to namr description at the beginning of this chapter.
namr2	File descriptor to which data is transferred. Can not be an LU number. The following namr2 subparameters have non-standard default values. File type - If zero or not specified, defaults to the type of namr1 if namr1 is a file. If namr1 is not a file, default is type 3. File size - If given, must be positive; default is 10 blocks if namr1 is an LU, or is equal to the size of namr1 when namr1 is a file.
format	The format of the data being transferred. Default is derived from namr1 if namr1 is a file, otherwise default is ASCII. The choices are: AS ASCII BR Binary relocatable; checksum is performed BN Binary; no checksum is performed BA Binary absolute; checksum is performed
mode	Transfer mode. May be entered when both namr1 and namr2 are files. A non-zero value causes both files to be opened as type 1 to increase data transfer rate. Extents are copied if both source and destination nodes support extendable type 1 and 2 files.

The STore command transfers a file from NODE1 to a file, created as specified in the namr 2 parameter, at NODE2. NODE2 is defined in the SWitch command. If the first record read is an EOF or EOT, the file is created with no data.

If namr1, which exists at NODE1, is a terminal, a slash (/) prompt is output to signal readiness for the next line. The next line should not be typed until the prompt is displayed. An end-of-file mark is made by entering a control-D.

If the file name specified in namr2 already exists, in NODE2 a file will be created with the first two characters replaced with 2 periods (..) and a warning message will be displayed.

If the mode parameter is not given or is set to zero, NODE1 source file record lengths are limited to 128 words. A record length exceeding 128 words will result in a -104 error code (record size error). This record size limitation may be overcome by setting the mode parameter to a non-zero value if the source file has no extents.

Non-zero Transfer Mode

A non-zero transfer mode parameter normally results in increased line efficiency since variable record length files will be transferred in 128-word data blocks rather than record-by-record. However, you should use this mode used only if the destination nodes support extendable type 1 and 2 files. If they do not, the destination file will be corrupt; however, no error message is printed.

NOTE: You can not use non-zero transfer mode for storing files with odd-byte length records from RTE-A systems to non RTE-A systems.

Examples:

#ST,1,FILE:::3:10	Create FILE on first available cartridge as type 3 with 10 blocks. Input from LU 1
#ST,XYZ::2,XYZ::17	Transfer data from XYZ on cartridge 2 at NODE1 to the same type of file, XYZ on cartridge 17, at NODE2. File XYZ on cartridge 2 is not affected.
#ST,XYZ,ABC,BR,1	Transfer the binary relocatable data from XYZ at NODE1 to new file ABC at NODE2. Both files are opened as type 1 to increase transfer rate.

SW — Switch

Changes or displays the origination and/or destination nodes of subsequent REMAT commands.

FORMATS

```
1)  SW
2)  SW[,NODE1[,NODE2]],security code
3)  SW,LO[cal]
4)  SW[,NODE1[:user.group/password]][,NODE2[:user.group
                       /password]],security code
```

NODE1	If a REMAT command requires one node then it uses NODE1. If the command requires two nodes, then NODE1 is the node from which action originates when a REMAT command is issued. This is a positive node number or the negative logical unit number of a neighboring node.
NODE2	If a REMAT command requires two nodes: the node to which the results of the action taken are destined when a REMAT command is issued. This is a positive node number or the negative logical unit number of a remote or the local node.
security code	The local node's network security code assigned to the node at initialization time.
LO(cal)	A literal parameter that forces NODE1 and NODE2 to the local node number.
user.group/password	The account and optional password for which a non-interactive session will be created at the specified node and to which subsequent REMAT commands will non-interactively attach. The previous session at this node will be released. This parameter is not required if the default account is desired. For non-session access, this parameter is the password only. This parameter is illegal for the local node. User, group, and password may each be up to 10 characters in length.

SW without parameters causes the current values of NODE1 and NODE2, together with account name if any, to be displayed on the log device.

For example:

```
    $SW
    NODE1 =       4            NODE2 =      10
    ACCOUNT NAME =             ACCOUNT NAME =
    USER.GENERAL               KELLY.FP
```

The account password is not displayed by the SW command.

If the default session was created at a node, "DEFAULT" will be displayed instead of an account name. If no session is created, "<NONE>" is displayed.

When REMAT is initially executed, it assumes commands are destined for the local operating system. When the SW command is entered, and an account is specified for a node, REMAT will create a non-interactive session for the named account at the remote node. Up to 16 remote nodes may each have one non-interactive session per user, at the same time.

Subsequently referenced remote files must reside on a mounted private or group cartridge to which you have access or on a system disc. If a given node already has an active session created from this REMAT, the session will remain intact if you switch "away" from the node and switch back to it later. To avoid a second log on (with log off of the prior session), do not repeat the account name when switching back to a prior node.

For example, if you wanted to STore a file from node 4 to node 10,

```
+-------+      +-------+      +--------+     +-------+      +-------+
|node 2|------|node 6|------|node 10|------|node 8|------|node 4|
+-------+      +-------+      +--------+     +-------+      +-------+
You are                      Copy of file                 Original
here.                        is needed here               file is here
```

you would issue this sequence of commands:

```
:RU,REMAT
$SW,4:user1.account,10:user2.account,sc
#ST,original file, new file
```

where:

sc = the network user's security code

The password is defined by the system manager at the destination node during network initialization. In non-session mode, you may only access files on system global cartridges but you will have full operator command capability.

If you wish to gain non-session access, the SWitch command may be issued in the form:

```
#SW,4:/password,10:/password,sc
```

All REMAT commands except EX,LC,LL, and TR are affected by the SW command. Table 2-3 shows the effect of the SWitch command on the REMAT commands. Broadcast command BC sends a message to all nodes and is not affected by command SW.

Table 2-3. Effect of SW

COMMAND	NODE1	NODE2	LOCAL
AT	Session created here (optionally)	Session created here (optionally)	
BC	Message delivered at all nodes	Message delivered at all nodes	Msg. delivered at all nodes
CL	Cartridges mounted here		List here
CR	File created here		
DE	Detaches session here (optionally)	Detaches session here (optionally)	
DL	File directory here		List here
DU	From file or lu	To LU	
EX			Only used here
FL	File closed here		
IO	I/O configuration listed here (RTE-L/XL/A)		
LC			Only used here
LI	From file	To lu	
LL			Only used here
LO	From file	To RTE-MIII or RTE-L/XL/A	
PL	From RTE-MIII or RTE-L/XL/A		List here
PU	File purged here		
QU	Schedule program here		

Table 2-3. Effect of SW (cont.)

COMMAND	NODE1	NODE2	LOCAL
QW	Schedule program here		
RN	File renamed here		
RW	Scheduled program		
SD	Shut down session here		
SL	Slave program here		List here
SO	Slave program here		
ST	From file or lu	To file or lu	
SW			Only used here
TE	Message sent here		
TR	Transfer to file at any node	Transfer to file at any node	Trnsfer to file at any node

TE — Send Message

Sends a message to a remote system console.

FORMAT

```
TE,message
```

message An ASCII string of up to 72 characters in length.

The message is sent to NODE1 (refer to SWitch command) and displayed on the system console. The message is prefixed with the following:

```
(FROM NODExxxxx):  message text
```

where:

xxxxx	is the node where the message originated.
message text	is up to 72 characters of ASCII-coded text.

TR - Transfer Control

Transfers control of REMAT to a file at any node or a logical unit at the local node.

FORMAT

```
       [,namr[,node]]
    TR
       [,-integer]]
```

namr	The logical unit of a local input device or a file name with security code and crn optionally specified. Refer to namr description at the beginning of this chapter.
node	Positive node number or negative logical unit number of a node where the command file exists. If namr is an LU, node is ignored.
-integer	Negative integer that denotes a transfer back through the stacked transfer files. Current command file is not included in the count.

TR with no parameters returns control to the previous command file or device. If this is a file, the records already read are skipped. If there was no previous input device or file, REMAT terminates. An EOF in the command file also causes a return to the previous command file.

The "node" parameter and the namr subparameters "crn" and "security code" can only be set on the initial transfer in a nested set of transfers (including a transfer to a file designated in a RU,REMAT command). If the initial transfer was not contained in a RU,REMAT command, you can reset the node and namr parameters by backing down past the initial transfer (to the base) and then starting a new transfer set.

Up to seven input devices or files may be stacked within the same node and then transferred to. If the parameter "-integer" is greater than the number of devices or files in the stack, REMAT terminates.

In the command file, all REMAT commands must be preceeded by the '$' prompt.

For example:

$TR,TRANS1:DS:202,15	Transfer command input to file TRANS1 which has a security code of DS and is on cartridge 202 at node 15.
$TR,TRNS2,-16	Transfer command input to file TRNS2 at the node connected with LU 16.
$TR,-5	Transfer back five steps through the stack.

Operator Commands Example

To become familiar with REMAT commands you must use them. Two examples follow to help familiarize you with them. The operator inputs are underlined. In addition to the commands listed in Table 2-3, all operator commands are available.

:RU,REMAT	Schedule REMAT
$LC	REMAT prompts with a '$'. Find out your local node number.
LOCAL NODE= 10	For this example, assume your local node is node 10. Your own may be different.
$SW	Find out the value of NODE1 and NODE2 and the account name you are logged on under, if any. The node numbers should be equal to the local node number and the account names should be the same as the name you initially logged onto the system as.

```
   NODE1 =      10       NODE2 =      10
   ACCOUNT NAME =        ACCOUNT NAME =
   USER.GENERAL          USER.GENERAL
```

Find out the number of the other nodes in the network. There are three ways to do this:

1) Ask the Network Manager
2) Go to the other node, run REMAT, and give the 'LC' command
3) Run DSINF and issue the 'NR' command

`$SW,,15:JIM.DS,DS`	Switch the value of NODE2 to a remote node and account. (In this example, 15 is a remote node number and JIM.DS is the remote account name.) DS is the security code of the local node. This will probably be different for your node. You must ask the Network Manager for the security code.
`#SW`	List the new node values and account names for node 10 and 15.

```
   NODE1 =       10        NODE2 =       15
   ACCOUNT NAME =          ACCOUNT NAME =
   USER.GENERAL            JIM.DS
```

`#ST,1,FILE1`	The '#' is the new prompt. Create a file at node 15 and transfer data to it from your local terminal.
`/$TI`	The / is your prompt for input. Since this is to be a transfer file, all commands must begin with a '$'.
`/$SW`	
`/$CL`	Entries into the command file.
`/$TR`	
`/`	A Control-D marks an End-of-file.
`#SW,15,10`	Switch the value of NODE1 to be the remote node (node 15) and NODE2 to be the local node (node 10). No network security code is needed if it was given in a previous SW command.

```
#DL,FILE1                Search the directory at node 15 for FILE1.

    ILAB= WHFILE   REMOTE DLIST:  NODE= 15   CR#=  00002  DIR TRKS= 01

    NAME     TYPE    #BLKS/LU  OPEN TO
    FILE1    00003    00010


#LI,FILE1                List FILE1 to your terminal.

0001   $TI
0002   $SW
0003   $CL
0004   $TR

#TR,FILE1,15             Transfer to  the command file, FILE1,  at node
                         15.  It  makes no difference what  node values
                         you have set in the SW command.
```

The following is an example of what should appear on your terminal when the transfer file is in control.

```
   $TI
   1980 164   8  56  54
   $SW
   NODE1 =    15          NODE2 =    10
   ACCOUNT NAME =         ACCOUNT NAME =
   JIM.DS                 USER.GENERAL
   $CL
     LU  LAST TRACK   CR    LOCK  REMOTE NODE= 00015

     44      00202   17226
     02      00202   00002
     03      00255   00003
     47      00074   21328

   $TR

#RN,FILE1,TEST1            Rename FILE1 to TEST1.

#ST,TEST1,4                Store the command file to LU 4

#PU,TEST1                  Purge TEST1.

#EX                        Exit REMAT.
```

The example above does not use all the REMAT commands. It is intended only to show you some of the things you can do with REMAT.

Chapter 3
RMOTE Operator Commands

Introduction

RMOTE is a program used to direct commands from the HP 1000 to the HP 3000. All of the MPE (Multiprogramming Executive) commands can be used in addition to eight RMOTE commands (EX, LL, MO, RU, RW, SW, SV, and TR). The SW command switches the destination of the commands from an HP 3000 to the local node and back. When the commands are directed to an HP 3000, RMOTE displays a pound sign (#) prompt. When the commands are destined for the local node, RMOTE displays a dollar sign ($) prompt. Any RTE command is entered in response to the '$' prompt. RMOTE commands can be entered in response to either the '#' or '$' prompt. However, RU, ON, or RW are allowed only in response to the '$' prompt.

RMOTE provides a limited virtual terminal capability, enabling you to interact with an HP 3000 in a manner similar to a local HP 3000 user. RMOTE does not provide a true virtual terminal capability because certain terminal characteristics such as Break, control-Y, block mode reads, and line cancel are not the same on RTE. As a result, applications that make use of or depend on these features may not function correctly when run from RMOTE.

This manual assumes that you have knowledge of the HP 3000. You should be familiar with the HP 3000 operator command, DSCONTROL and the HP 3000 command DSLINE. You should therefore refer to the MPE Commands Reference Manual (refer to the Documentation Map at the front of this manual for manual part number) for additional information on the HP 3000.

For information on using DS/3000 to communicate with an HP 1000 system, refer to the "DSN/DS HP 3000 to HP 1000 Reference Manual for HP 3000 Users."

PSI BISYNC Connections

DINIT initializes PSI BISYNC connections (12793A, 12834A, 120734A and 12082A cards). To alter line characteristics or to close the line, you must use the program DSLIN. There are two modes of operation for PSI BISYNC lines: primary and secondary. In primary mode, the HP 1000 initiates communication with the HP 3000. In secondary mode, the HP 1000 waits for the HP 3000 to initiate communication (with the DSCONTROL and DSLINE commands); users on the HP 1000 can not establish remote sessions on the HP 3000 until the HP 3000 DSCONTROL command has been issued.

When you enable a PSI BISYNC lu with DINIT, the PSI line is put in secondary mode. The default DSLIN line characteristics (shown in the following pages) are used. To open a line in primary mode or to change line characteristics, you must run DSLIN. The following pages explain how to run DSLIN.

Primary Mode

If the line is not up (neither side has initiated communication) and DSLIN has been RP'ed, calling the DS/1000-IV subroutine HELLO causes DSLIN to be scheduled in primary mode with the default line characteristics. DSLIN messages are sent to the scheduling terminal if the severity code is less than two.

To close the line, you must run DSLIN with the close parameter (or use DSMOD on RTE-MIII systems).

To initialize the line in primary mode with non-default line characteristics, run DSLIN as described in the following pages.

Secondary Mode

Whenever the physical link goes down, the PSI BISYNC card automatically re-enables the card in secondary mode. Thus, putting a line in secondary mode is like issuing the DSCONTROL xxx;OPEN command on an HP 3000.

Closing the Line

If the HP 3000 sends a termination request, the line is closed when there are no active sessions on the line.

To close a line on an RTE-MIII system, run DSMOD and use the /L command. This closes the line and then re-enables the card in secondary mode.

On all other systems, run DSLIN and specify that you want to close the lu. If there are no other users on the line, DSLIN sends a termination request to the HP 3000 and prints:

```
              LU xxx WAS CLOSED
```

DSLIN then terminates. If the HP 3000 accepts this request, the line closes. The physical link (for example, a modem line) also closes. You can use the DSINF LU and VA commands (documented on the Network Manager's Manual, Vol. II) to verify that the line is closed.

If there are other users on the line when you try to close it, DSLIN prints:

```
              SESSIONS STILL OPEN ON LU xxx.
              THE LU WAS NOT CLOSED.
```

DSLIN then terminates.

DSLIN

DSLIN is scheduled with the RTE ON or RU command. DSLIN may also be scheduled from a program with an EXEC or DEXEC call.

For RTE-MIII, you must use the default values when initializing the HP 3000 LU. The run string for DSLIN under RTE-MIII is:

```
             RU,DSLIN,[LU[,"S" or "P"]]
```

For all other operating systems, the run string is:

```
                           ["-"], [S or P]
             RU,DSLIN,[LU],
                           [Command file]
                where:

                       LU  = the positive HP 3000 LU

                       "-" = use the default parameters

                       S   = Secondary mode

                       P   = Primary mode
```

To close a line (non-MIII systems), the run string is:

```
            RU,DSLIN,[LU],CLOSE

               where:

                      LU  = the positive HP 3000 LU
```

You can not close a line by running DSLIN with a command file. The word CLOSE must be fully entered.

Interactive DSLIN example:

```
 :RU,DSLIN

HP 3000 LU TO INITIALIZE: xx
```

where xx = the positive HP 3000 LU

When DSLIN prompts for a number which has a default, it includes the default value in brackets. Type /D or press return to use this default.

The next question is only asked if you are running DSLIN interactively. Do not include an answer for it in DSLIN command files.

```
OPEN OR CLOSE THIS LU?
```

If you reply CLOSE, DSLIN tries to close the line, as described in the previous section and terminates.

If you reply OPEN, DSLIN proceeds to the next question.

/D = OPEN

```
COMMUNICATIONS BLOCK SIZE [nnnn] : xxxx
```

where xxxx is between 304 and 4096 words

Any value specified must be a multiple of 16 which is less than the library generated in. If the number specified is not a multiple of 16, the next higher multiple of 16 is used.

An out of range value results in the following message:

```
INPUT VALUE OUT OF RANGE MUST BE BETWEEN 304 AND 4096.
DEFAULT VALUE OF nnnn USED.
```

The default value (nnnn) depends on the buffer size library specified at generation time; $D3KRB: 304, $D3KBB: 1072, $D3KMB: 4096

If a value which is not a multiple of 16 is specified, DSLIN uses the next highest multiple and prints the message:

BLOCK SIZE CHANGED TO xxxx.

The maximum buffer size of the PSI BISYNC card is 1608 words; this is not equivalent to the communications block size. For more information, refer to the Network Manager's Manual, Volume I.

MAXIMUM RETRY COUNT [8] : xx

where xx = the number of times a message will retry if an acknowledgement fails to arrive.

/D = 8

CONNECT TIMER (SECONDS) [255] : xxx

where xxx = the time allowed for response before the line is placed in secondary mode.

/D = 255 seconds

LOCAL ID SEQUENCE: xxxxxxxxxxxxxxx

There is one local ID sequence with up 15 ASCII characters.

For all local and remote IDs, /D indicates a null (0 length) ID sequence and /E indicates no more ID sequences are to be used.

```
REMOTE ID SEQUENCE 1 : xxxxxxxxxxxxxxx
REMOTE ID SEQUENCE 2 : xxxxxxxxxxxxxxx
        .
        .
        .
REMOTE ID SEQUENCE 15 : xxxxxxxxxxxxxxx
```

There are up to 15 remote ID sequences allowed with up to 15 ASCII characters each. For multiple HP 3000s connections each with unique ID sequences, the values specified here are searched for any match.

At this point, DSLIN sends the configuration information to the I/O board, then reads back the board's parameters. If the board type is not 0, DSLIN prints:

```
THE BOARD ON LU nn DOES NOT CONTAIN BISYNC FIRMWARE!
```

and terminates. (DSINF's LU,nn,PA command reports which firmware is on the board.)

If the board reports a buffer size different from the one specified, DSLIN prints

```
BISYNC BOARD ON LU nn REPORTS BUFFER SIZE OF wwww WORDS
```

```
INITIALIZE STATION AS PRIMARY OR SECONDARY?  x
```

where x = S or P

If "S" is the response, the BISYNC board is initialized in secondary mode. The 1000 accepts remote sessions from the 3000, but it does not allow 1000 users to set up remote sessions to the HP 3000 until the HP 3000 has established communication with the HP 1000. DSLIN responds as follows:

```
CONNECTING AS SECONDARY STATION ON LU xx
AWAITING CALL ON LU xx
```

If "P" is the response, the BISYNC board will be initialized in primary mode and DSLIN will respond as follows:

```
HP 3000 BISYNC LINK ON LU nn READY FOR CONNECTION
```

The HP 1000 now sends an initialization request to the HP 3000. If the HP 3000 has been initialized for 1000/3000 communication it answers the request and the link is established. You may then establish a remote session on the HP 3000.

If the message is sent but the HP 3000 does not respond to the DS/3000 initialization request within 100 seconds, DSLIN prints

```
LINE IS UP BUT 3000 IS NOT REPLYING
```

disconnects the link, and tries to reconnect as a secondary station.

DSLIN goes into a loop where it waits two seconds, writes a DS/3000 initialization request, and checks the returned I/O status until the message is sent or a timeout occurs. To prematurely terminate this loop and force a secondary connect, set DSLIN's break flag with the RTE BR command. When DSLIN detects its flag, it prints

```
   BREAK FLAG SET.
```

DSLIN then tries to connect as a secondary station.

If the primary connect request times out (this is usually because the MODEM connection was not made or, on a direct connection, because the HP 3000 DSLINE was not opened) DSLIN prints

```
   PRIMARY CONNECT TIMED OUT.
```

and tries to connect as a secondary station.

If the connection is made, DSLIN responds as follows:

```
   LINE IS UP WITH BUFFER SIZE nnn.
```

RMOTE

RMOTE is scheduled with the RTE ON or RU command. RMOTE may also be scheduled from a program with an EXEC or DEXEC call.

FORMAT

```
                [input LU[,log LU[,severity code]]]
     RU,RMOTE             or
                [,namr[,log LU[,severity code]]]
```

In RTE-MIII systems, the format to read from a file is:

```
     RU,RMOTE,fi,le,nm[,security code[,crn]]
```

input LU	The logical unit number of the input and $STDIN device. Default is the Multi-Terminal Monitor LU or the value returned by LOGLU.
namr	The file which may optionally be specified to provide all input commands. All local commands must be preceded by the '$' prompt and all remote commands must be preceded by the '#' prompt.
log LU	The logical unit number of an output device for $STDLIST request and for logging an error detected on a command from the input device. The default log LU is the input LU (if interactive) or the value returned by LOGLU.
severity code	The error message code. Default is 0. If an invalid code is entered, it is defaulted to 0. The options are:

0 - all commands are echoed on the log device. Any error causes an appropriate error message to be printed.

1 - inhibit command echo on log device.

2 - inhibit messages to log device, unless error is severe enough to cause control to transfer to log device for command input.

fi,le,nm	The file which may optionally be specified to provide all input commands. All local commands must be preceded by the '$' prompt and all remote commands must be preceded by the '#' prompt. In RTE-MIII systems, input files cannot be specified as a namr.
security code	The security code of the file specified in filenm.
crn	Cartridge identifier of the file specified in filenm.

If, when reading from a non-interactive input LU or command file, the RTE break flag is set, control transfers to the top of the command stack or (if top of stack is non-interactive) RMOTE terminates.

RMOTE upshifts all parameters that follow RMOTE commands, except for the "NOW" option of the "RU" and "ON" commands.

MPE Commands

RMOTE provides a remote interface to all MPE operator commands. These commands are described in the MPE Commands Reference Manual (refer to the documentation map at the front of this manual for manual part number).

Table 3-1 is a list of MPE commands. This list is subject to change so you should refer to the MPE manual for a complete list of current commands, their parameters, and meanings.

Table 3-1. MPE Commands

FUNCTION	COMMAND
Running Sessions	HELLO BYE ABORT RESUME EOD EOF
Determining session/job status and resource usage	SHOWJOB REPORT
Managing Files	FILE RESET BUILD LISTF RENAME PURGE SAVE STORE RESTORE ALTSEC RELEASE SECURE DATA

Table 3-1. MPE Commands (Continued)

FUNCTION	COMMAND
Running subsystems and user programs	BASIC BASICOMP BASICPREP BASICGO COBOL COBOLPREP COBOLGO FORTRAN FORTRANPREP FORTRANGO RPG RPGPREP RPGGO SPL SPLPREP SPLGO SPOOK PREP PREPRUN RUN EDITOR RJE SEGMENTER SETDUMP RESETDUMP
Determining device and device file status	SHOWDEV SHOWIN SHOWOUT
Requesting utility operations	TELL TELLOP SETMSG SPEED PTAPE GETRIN FREERIN SHOWTIME
DS/3000 commands	DSLINE REMOTE

RMOTE Operation

The '$' prompt means the commands entered will be processed locally at the RTE node. The '#' prompt means the commands entered (except SW, LL, MO, SV, TR, ON, RU, RW and EX) will be sent and processed at an HP 3000 node.

When entering commands to an HP 3000 ('#' prompt) you may enter SW to perform RTE functions locally ('$' prompt), then SW again to resume HP 3000 commands without having to log on to the 3000 again. If a 3000 LU is not specified, the default is the first BISYNC LU specified at DS initialization.

Whenever RMOTE reads a command line with an asterisk (*) in column 1, RMOTE treats the line as a comment.

When switching to a 3000, you may specify the 3000s LU or X.25 address. When RMOTE starts, the default 3000 LU is the first LU specified at DS initialization. After you have established a session at the 3000 (by a HELLO), the LU parameter is not needed. If an LU is provided which is not the LU of the 3000 being used, RMOTE prints "INVALID REMOTE LU." If you wish to logon to a different 3000, you must logoff your current session (with a BYE) before switching to the new LU.

For Example:

```
    :RU,RMOTE
    $                  Local RTE command processing is now in effect.
    .
    .
    .
    $SW,<LU>           Switch to an HP 3000 node.
    #HELLO <ACCT>      Once the HELLO command is given, standard
    .                  HP 3000 command processing can take place.
    #SW
    $                  Local RTE command processing is now in effect
                       again.  Session on HP 3000 is still active.
    $SW,<LU>
    #                  Proceed with HP 3000 command processing again.
    .
    .
    #BYE
    #EX                Terminate HP 3000 session.

          or
```

```
:RU,RMOTE
$SW,<LU>
#HELLO <ACCT>
.                     All processing is done at a 3000 node.
.
.
#EX                   Terminate HP 3000 session (BYE is issued
                      automatically).
```

NOTE: If RMOTE terminates after a HELLO is issued but before a reply is received from the HP 3000, you may still have a session on the HP 3000. (This will happen if the HP 1000 goes down, the 1000-3000 communications link goes down, or if an "OF,RMOTE" command is issued before a reply is received. It is also possible for RMOTE to time-out prior to receiving a reply.) If a session is created in this way, it must be aborted at the HP 3000.

The break and control-Y functions of the HP 3000 are accessed through the program RMOTE. The break flag is checked each time a message is printed to, or read from, the terminal. To initiate the HP 3000 break or control-Y, you first press any key to gain the attention of RTE, then enter the command:

```
BR,RMOTE
```

RMOTE will display the following:

```
ENTER CONTROL REQ (B OR Y)
```

Enter B to break and Y for a control-Y.

NOTE: In session nodes, you just enter BR. The system will determine which program is your copy of RMOTE.

Commands

The commands listed in Table 3-2 are used to control RMOTE. The ON, RU, and RW commands require the '$' prompt. The HELLO and BYE commands require the '#' prompt. All other commands are entered in response to the '#' or '$' prompt. RMOTE commands may be either upper or lower case letters.

To avoid conflicts with User Defined Commands (UDCs), RMOTE will not recognize its own commands (SW, LL, MO, etc.) if extra characters are added. For example: use MO not MOVE; EX not EXIT; SW not SWITCH; etc.

When commands are read from a file or non-interactive LU, the first column must contain the proper prompt character ('$' or '#').

Table 3-2. RMOTE Commands

COMMAND	DESCRIPTION
EX	Terminates RMOTE execution
LL	Changes the $STDLIST device
**MO	Moves files between the HP 1000 and HP 3000
*RU or ON	Schedules a local HP 1000 program
*RW	Queues with wait a local HP 1000 program
SW	Switches the destination of RMOTE Commands
SV	Sets or changes severity code
TR	Transfers control to a file or logical unit at the local node

* These commands must be issued from the '$' prompt. ** Only with MOVE version of RMOTE (%RMOT1).

EX — Exit

Terminates RMOTE execution.

FORMAT

```
   EX
```

If a session established by RMOTE is running on the HP 3000, EX will first generate a BYE command, then end RMOTE's execution.

LL — Change List

Changes the $STDLIST device.

FORMAT

```
            LL,lu
```

lu	The logical unit number of the new $STDLIST device at the local node.

LL changes the list device at the HP 1000 node for list output from the HP 3000.

After this command is entered, a prompt may be displayed on the new $STDLIST device, but responses are read from the input device. (If the input device is interactive, the prompt will be re-displayed on it before the response is read.)

MO — Move

Moves files between an HP 1000 and HP 3000. The direction of the transfer depends on what system RMOTE is presently sending commands to. For example, a '$' prompt indicates that commands are being processed by an HP 1000. A "MO" command issued at a '$' prompt will transfer a file from an HP 1000 to an HP 3000. When a '#' is present, however, commands are being processed by an HP 3000. A "MO" command issued at a '#' prompt will transfer a file from an HP 3000 to an HP 1000.

In either direction, the system reports the number of records transferred on the terminal. (NOTE: Transferred files may not have record sizes larger than 768 bytes.)

In the event that a file of the same name exists on the system to which the new file is being moved, you will be asked whether or not you want to overwrite the remote file. If you do want to overwrite the file, type "yes" and the transfer will be completed. (NOTE: If the file being transferred is longer than the remote file, the remote file will not increase in size and data will be truncated.) If you do not want to overwrite the file, type "no" and the transfer will not take place.

HP 1000 TO HP 3000 FILE TRANSFER FORMAT

When a file is moved from an HP 1000 to an HP 3000 ('$' prompt), use the following command:

```
                                        --
           ,namr                 [:CC] |
        MO       ,filename[,[UN]       |
           ,LU                   [:SP] |
                                        --
```

namr or LU	The location in the HP 1000 of the file to be moved.
filename	The filename created on the HP 3000. If the length of the file to be transferred is more than 1023 records, a larger file size must be established on the HP 3000 before the move can be successfully completed. To set the file size, switch to the HP 3000 and type:

```
#FILE <filename>;DISC=<number of records>
```

then switch back to the HP 1000 to complete the move under the HP 1000 '$' prompt. (Any MPE file attribute can be changed from its default by using

	the FILE command.) This equation does not alter the characteristics of an existing file.
UN	Indicates the file is to be written as fixed length records. RMOTE uses a default size of 80 characters, but this can be overriden by the MPE 'FILE' command.
:CC	Indicates column 1 is to be used for carriage control at MPE.
:SP	Indicates the RTE file is spooling system format. I/O control embedded in the file will be translated to MPE carriage control characters.

```
********
* NOTE *
********
```

If input is from an interactive LU, RMOTE prompts for data with a '/'. Control D terminates data entry.

HP 3000 TO HP 1000 FILE TRANSFER FORMAT

When a file is moved from an HP 3000 to an HP 1000 ('#' prompt), use the following command:

```
             ,namr
MO,filename          [,UN]
             ,LU[:SP]
```

filename	The name of the HP 3000 file to be transferred to the HP 1000.
namr or LU	The HP 1000 destination of the HP 3000 file to be transferred. The length of this file may not exceed 16383 blocks.

a. If the "to" LU is a magnetic tape drive an EOF is written when the move completes.

b. If the "to" LU is a line printer, column one is interpreted as carriage control unless octal 200 (decimal 128) is added (to set the V bit). A top of form is written when the move completes.

:SP	Indicates output is to be spooled to the RTE LU (at priority 99). If this is specified, RMOTE creates a spooling file RM<LU><NN> on the spool disc, where <LU> is the logical unit number of the log device and <NN> is a number between 00 and 99 (e.g., RM0900). The file is purged when outspooling is completed.
UN	Tells the HP 3000 to remove characters to the right of column 72. (This is useful for MPE editor files kept numbered.)

Blanks are removed from the end of ASCII records. If an odd number of bytes are read, the record is padded with a blank.

If the RTE break flag is set during a file move, the operation is terminated and control returns to the input LU or command file. If the file move is from an HP 1000 to an HP 3000 and RMOTE is aborted, the file will never be created on the 3000. However, if the move is from an HP 3000 to an HP 1000, a partial file will be created on the 1000; but it will not be closed.

An advantage to using the MOve command (MO,filenm,1,UN) in moving files from the HP 3000 to the HP 1000 is the ease in listing files located on the 3000 to your terminal. This process eliminates the task of scheduling the 3000 editor, texting a file, and then listing it to your terminal.

The following is an example of an RMOTE command file using the MO command:

```
$SW
#HELLO DAVE.DATACOM
#*SET UP LINE PRINTER ON 3000 WITH NAME "LINEP"
#*USE CARRIAGE CONTROL
#FILE LINEP;DEV=LP;CCTL
#SW
$* FILE SPOOL IS IN SPOOLING FORMAT
$MO,SPOOLO:SC:116,LINEP,:SP
$SW
#* COPY3KS IS AN MPE EDITOR FILE.  MOVE TO LU 6 AND SPOOL IT.
#*   PRINT COLUMN 1, AND LEAVE OFF THE LINE NUMBERS.
#MO,COPY3KS,206B:SP,UN
#EX
```

RU, ON - Schedule Program

Schedules a local HP 1000 program.

FORMAT

```
        RU
          ,program[,NOW][,parameters]
        ON
```

program	Name of program to be scheduled.
NOW	If specified, schedules a program immediately. Commands with the now option are passed directly to RTE and are not upshifted.
parameters	Up to 5 parameters or a string to be passed to the program. If the first parameter is not provided, it is set to the value returned by LOGLU. If the fifth parameter is not provided, it is set to the negative Session Main Process number (SMP) obtained if RMOTE successfully establishes an HP 3000 session. If the NOW option is used, however, neither the SMP number nor the LOGLU value will be passed to the scheduled program.

RU and ON schedule a program on the local HP 1000. If the program is busy, a PROGRAM BUSY message is returned and control returns to RMOTE. If you wish to have RMOTE wait until the program is available, use the RW command.

If you schedule a program which uses more than four parameters and also needs the SMP number, you must design it to pick these parameters up from the scheduling string.

This command does not restore or "clone" programs. The program must have an ID segment before RMOTE can schedule it.

RW — Run with Wait

Queues, with wait, a local HP 1000 program.

FORMAT

```
        RW,program[,parameters]
```

program	Name of program to be scheduled.
parameters	Up to five parameters or a string to be passed to the program. If the first parameter is not provided, it is set to the value returned by LOGLU. If the fifth parameter is not provided, it is set to the negative Session Main Process number obtained if RMOTE successfully established a session (see PRCNM).

RW schedules a program, with wait, on the local HP 1000. If the program is busy, RMOTE waits until the program is available. If you do not want RMOTE to wait for a program to be available, use the RU command.

If you schedule a program which uses more than four parameters and also needs the SMP number, you must design it to pick up the parameters from the scheduling string.

This command does not restore or "clone" programs. The program must have an ID segment before RMOTE can schedule it.

SW — Switch

Establishes which 3000 a HELLO will be sent to or toggles between 3000 and 1000 destination for commands. You cannot establish sessions at two different 3000s at the same time. When switching to logon to a 3000, you may specify the 3000's LU or X.25 address.

FORMAT

```
        [,LU]
      SW
        [,X.25 address[,X.25 LU]]
```

LU	is the positive LU number of an HSI or Bisync link to a 3000.
X.25 address	is the X.25 network address assigned to a 3000. The character "#" must appear before the address and the

address can consist of up to 15 digits.

X.25 LU	specifies which physical X.25 link to use. Only necessary when more than one X.25 I/O board is in use. This is not the virtual circuit LU but is the network LU as defined by XINIT, the X.25 initialization program.

EXAMPLE: `$SW,#3110301057,57`

This command switches to the 3000 connected via the X.25 card associated with LU 57 that has address 3110301057. Once a HELLO has established a session at the 3000, the address and LU do not need to be provided when switching back to the session.

When RMOTE starts, the default is the first 3000 LU specified at DS initialization. After you have established a session at the 3000 (by a HELLO), the LU parameter is not needed. If an LU is provided which is not the LU of the 3000 being used, RMOTE prints "INVALID REMOTE LU." If you wish to logon to a different 3000, you must logoff your current session (with a BYE) before switching to the new LU.

SV — Severity

Sets or changes the severity code.

FORMAT

```
SV,severity
```

severity may be:

0 Display error codes and echo commands on log device (default).

1 Inhibit command echo on log device.

2 Command echoed only if RMOTE error occurred.

The default mode is to echo a command as it is entered on the input device and to log all errors on the same device. During interactive operation, commands are not echoed.

The severity code can be set to 1 when there is no advantage to echoing commands at the console, for instance when commands are entered in a batch job. If, in addition, it is desired to suppress messages unless they cause an error, the code can be set to 2.

TR — Transfer

Transfers command input to a file or logical unit at the local node.

FORMAT

```
     [,namr]
  TR
     [,-integer]
```

namr	the file name, with optional security code and cartridge identifier, or logical unit number of an input device where commands are to be read.
-integer	negative integer that denotes a transfer back through the stacked command files. The current command file is not included in the count.

TR with no parameters returns control to the previous command file or device. If this is a file, the records already read are skipped. If there was no previous input device or file, RMOTE terminates. An EOF in the command file also causes a return to the previous command file.

Up to seven input devices or files may be stacked and then returned to. If the parameter integer is greater than the number of stacked files RMOTE is terminated with an EX command. If there is an outstanding HELLO command, EX will generate a BYE command.

Commands in the transfer file must begin with the '$' or '#' prompt depending on which prompt would normally be displayed by RMOTE.

Operator Commands Example

To become familiar with RMOTE commands you must use them. An example follows to help familiarize you with some of the commands. The operator inputs are underlined.

:RU,RMOTE	Schedule RMOTE
$SW,67	Switch to the HP 3000 node which is associated with LU 67.
#HELLO USER.ACC	Log on.
#SHOWTIME	Any MPE command may be given. SHOWTIME gives you the current time at the HP 3000.
WED, SEPT 16, 1980, 7:45 PM	
#SW	Switch to the local node. Your HP 3000 session is still active.
$TI or $TM	This RTE command displays the system time and is sent to the local operating system for processing. Use the "TI" command on M/E/F-Series and the "TM" command on L and A-Series machines.
1980 307 7 45 19	This is the time displayed at the RTE node.
$SW	Switch back to the HP 3000 node.
#LISTF	Because you did not give an EX or BYE command, you have the same session as before.

```
  FILE1      FILE2      FILE3      FILE4      FILE5      FILE6
  FILE7      FILE8
```

	All files in your account are listed.
#BYE	End the session.
#EX	Exit RMOTE.
:	When the EX command is given, command processing returns to the RTE node.

Commands From an HP 3000

REMOTE commands from an HP 3000 that generate command requests or user programs that issue RFA/DEXEC/PTOP requests from an MPE session to an HP 1000 will be directed to the HP 1000 session created by the HELLO command or to the default session.

DS/3000 HELLO and BYE commands are converted to DS/1000 format.

HELLO

The HELLO command directed to an HP 1000 is of the form:

```
   :REMOTE HELLO user.group/password
```

Non-session access cannot be performed by MPE.

BYE

The HP 3000 REMOTE BYE command is converted to a "log-off" request. This user's session at the HP 1000 is then released.

Chapter 4
DS/1000-IV Remote File Access

Remote File Management

A library of DS/1000-IV Remote File Access (RFA) subroutines is available which you can use to accomplish management of remote disc and non-disc files from your own programs. This adds more flexibility to your program since it may run from other nodes in your network without requiring any major re-editing. When your program is relocated, required subroutines from the library are appended.

When a remote user normally gains access to a Session Monitor node without first calling DLGON, a non-interactive session is created under a default account name. That account has access to specific private and group cartridges and all system cartridges. In order to do Remote File Access, the desired files must reside on one of these accessible cartridges. The exception is for non-session access, in which case only files on system global cartridges can be accessed.

If a program requests files from different session nodes, each node will have a non-interactive session created. A single user can have a maximum of sixteen remote sessions active concurrently.

Several log-on, log-off utilities (discussed in the Utilities and Program Calls Chapter of this manual) exist to help in programatically accessing files on specific accounts. REMAT can also be used to obtain the remote session, then the user program can be run (with wait) locally as a 'son' of REMAT. When each RFA call is received by the remote node, the Remote File Access Monitor (RFAM) attaches to the account name for the duration of the request and detaches upon completion. If the first request has no session ID, a session will be created under the default user name.

To preserve compatibility with pre-DS/1000-IV software and provide flexibility in accessing smaller files, there are two sets of RFA calls for some functions. One set uses single-word parameters for record number and file size specification. These routines can only be used when the number of records specified in the call is not greater than 32767 records and when the file size specified is not greater than 32767 logical sectors.

The second set of routines perform the same functions but have some double-word parameters in their calls. These routines may be used when the number of records specified is 0 to (2**31)-1 or when the size specified is less than or equal to 32767 times 128 blocks.

Double word calls cannot be made to 91740 nodes. If you send an extended RFA call to a 91740 node, you will get an IERR = -25 returned, bad FCODE in request. (DSERR will display 'DS-25'.)

The relationships of both sets of RFA calls to each other is shown in a later section.

NOTE

> Program calls to DS/1000-IV RFA routines are not supported in RTE-6/VM type 6 (Extended Background) programs, or to non-FMGR files.

Call Syntax

Program calls to DS/1000-IV RFA routines and subroutines may be written in Assembly language, FORTRAN, FORTRAN IV or PASCAL. In this section, the calls are shown as they would be written in FORTRAN IV. These calls may be written in any of the other programming languages using the same parameters. When assembled or compiled, the same code is generated.

For FORTRAN, the general form of the calling sequence is:

```
CALL <subroutine name> (<p1> ,<p2> ,...  ,<pn>)
```

Parameters <p1> through <pn> define a real or integer array, a variable, or a value. The interpretation of a parameter is determined by its position within the list of parameters.

For Assembly language, the general form of the calling sequence is:

```
EXT <subroutine name>
 .          .
 .          .
 .          .
JSB <subroutine name>
DEF * + n+1
DEF <p1>  \
DEF <p2>   \
 .   .      \
 .   .       >    <parameter list>
 .   .      /
DEF <pn>   /
```

RTN <return location>

RTN is the label of the location where the called routine returns upon completion. The return point must always immediately follow the last entry in the parameter list (<pn>).

In the calling sequences described in the following paragraphs, optional parameters are enclosed in brackets. These brackets are not to be included in actual coding of the call within your programs.

DS/1000-IV Remote File Access Calls

The DS/1000-IV Remote File Access (RFA) calls are used for file manipulation and control. The RFA calls are similar to the FMP program calls described in the Programmer's Reference Manual for your RTE operating system. Variations between RFA and FMP calls are described in the following paragraphs.

The single-word parameter RFA calls shown below include alternate entry points for double-word parameter (extended) RFA calls as follows:

Single-Word Parameter call	Corresponding Double-Word Parameter call
DCRET	DXCRE
DCLOS	DXCLO
DREAD	DXREA
DWRIT	DXWRI
DAPOS	DXAPO
DPOSN	DXPOS
DLOCF	DXLOC

The extended (DX---) file size is not provided for HP 3000 to HP 1000 remote file access.

RFA Call Parameter Variations

The following list defines the RFA call interpretation of general parameters that vary from usage in the FMP calls:

Parameter	RFA Interpretation
IDCB	Within an RFA call, a 4-word Data Control Block (DCB) is required for each file opened. You must be careful not to modify this DCB array. Note that it exists within the same partition as your program. Programs having the standard FMP 144-word DCB are accepted although a 4-word DCB is the minimum required by DS/1000-IV. This DCB is used to store parameters when the file is opened, so that any calls that reference this DCB will obtain the same file. The true DCB (144 words) is maintained by the RFA Monitor at the node where the file resides.
IERR	A variable to which a value representing an error code is returned if an error condition is encountered during execution of an RFA call. The RFA error codes are defined in the error section located in the back of this manual. Upon successful completion of a DCRET call, the number of sectors allocated to the file is returned. Upon successful completion of a DOPEN call, a value representing the file type is returned. If a value less than zero is returned, DSERR may be called to obtain an ASCII message describing the problem. The DSERR call is documented in the Utilities and Program Calls Chapter later in this manual.
IL	A value that declares the length of data for a read or write request (DREAD or DWRIT). This value is limited to a maximum of 512 words for the multiple DCB environment, or 128 words for the single DCB environment. Unlike FMP calls, this parameter is required for DREAD and DWRIT calls.

ICR	For RFA calls, this parameter is a 2-word array (instead of the single-word value used in FMP calls). Word 1 declares the type of cartridge reference. It is an integer value that can be positive, negative, or zero. If positive, the file search is restricted to the cartridge reference declared by the specified integer value. In session monitor nodes, any private, group, or system cartridge available to this specific user's account will be searched. If negative, the file search is restricted to the logical unit number declared by the specified integer value. If zero, the file search is not restricted to any particular cartridge. For DCRET, if ICR is zero, the file will be allocated on the first cartridge encountered that has enough room for the file; calls other than DCRET search the cartridges in the order in which they were mounted until the file is found. Word 1 is identical to the ICR parameter described for FMP calls. Word 2 declares the address of the node at which the call is to be executed. If this value matches the address of the local node, or is -1, the call is executed locally. This parameter may also be specified as the negative value of the communications line logical unit number. If this parameter is omitted, word 1 defaults to zero, and word 2 defaults to -1 (local node).
IERLC	An error condition location variable. If an error condition is encountered the address of the node at which the error occurred is returned.

File Definition Calls

A file is defined in terms of a node number, a unique name, size (in number of blocks), type, and location. To create, open, close, or purge files, call one of the following routines:

DCRET/DXCRE	- Define a new file.
DOPEN	- Open a defined file.
DCLOS/DXCLO	- Close an opened file.
DPURG	- Purge a file from directory.

File Access Calls

To read from and write to opened files you use the file access calls. Whether reading or writing, you can transfer exactly one record or declare a specific number of words to be transferred. It is recommended that you transfer your data in words because this provides a consistent method of data transfer among the various file types. The file access calls consist of the following:

DREAD/DXREA	- Read data from a file.
DWRIT/DXWRI	- Write data to a file.

File Positioning Calls

You use these calls to position a file to a specific point. See the appropriate RTE Programmer's Reference Manual for a description of file positioning capabilities. The RFA file positioning calls consist of the following:

DLOCF/DXLOC	- Retrieve current record pointer and status for file.
DAPOS/DXAPO	- Position a file to a specific record address.
DPOSN/DXPOS	- Position a file to a relative record address.
DWIND	- Position (rewind) file to start of first record.

File Control Calls

To issue input/output control requests to Type 0 files, to obtain the status of all disc cartridges in the cartridge directory, or to rename a file you use this set of routines. The file control calls consist of the following:

DCONT - Transmit I/O control requests to Type 0 files.

DSTAT - Obtain status of all mounted cartridges.

DNAME - Rename a file.

Remote File Access Syntax

For a more detailed discussion of the following calls, you may find it useful to occasionally reference the equivalent FMP routines in the Programmer's Reference Manual for your RTE operating system.

DAPOS, DXAPO

You can set the address of the next record to be accessed within a file by calling this routine. The record position set may be defined via a prior call to DLOCF. DAPOS is equivalent to the FMP routine APOSN. This call cannot be used with Type 0 (non-disc) files.

CALLING SEQUENCE

```
   CALL DAPOS(IDCB,IERR,IREC[,IRB[,IOFF[,IERLC]]])
```

for the extended format version:

```
   CALL DXAPO(IDCB,IERR,IREC[,IRB[,IOFF[,IERLC]]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).

IREC	Next record; a variable set to the number of the next sequential record in the file (can be determined by a prior call to DLOCF or DXLOC). For DXAPO, IREC is a double word variable with the same meaning.
IRB	Relative block address of the next record (optional); a variable set to the block number of the next record. This parameter is required for files of Type 3, 4, 5, 7, or greater (variable length records). For DXAPO, IRB is a double word variable containing the block number of the next record.
IOFF	Block offset of next record (optional); a variable set to the offset in the block of the next record. This parameter is required for files of Type 3 or greater.

NOTE

Although DS/1000-IV doesn't check for IRB and IOFF parameters, they must be present for all files having variable record length to ensure correct operation.

IERLC	Optional error condition location (see RFA Call Parameter Variations).

DCLOS, DXCLO

You may close a file, following access operations, using a call to DCLOS. DCLOS is equivalent to the FMP call CLOSE. When a file is closed, the logical relationship between the file directory entry and the Data Control Block is eliminated. This results in the Data Control Block being available for association with other files.

To save disc space, a file that was created having more blocks than are actually required to accommodate the data within the file can be truncated upon closing. Upon closing the file, you cause the file to be truncated at the actual end of data. This action releases unused blocks (allocated when the file was created) to the control of File Manager. A file can be truncated only if the following conditions are true:

* The file is a disc file.
* The file is currently positioned in the main file's disc space, not in a file extent.
* The file was opened using the correct security code.
* The file was opened for exclusive use of the calling program.
* The number of blocks to be deleted should be less than or equal to the total number of blocks within the file. If they are equal, the file is purged.

CALLING SEQUENCE

```
        CALL DCLOS(IDCB,IERR[,ITRUN[,IERLC]])
```

for the extended format version:

```
        CALL DXCLO(IDCB,IERR[,ITRUN[,IERLC]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).
ITRUN	File truncation (optional); a variable containing an integer value that defines the number of blocks to be deleted from the file upon closing. If zero or not specified, no truncation occurs. If negative, only file extents are truncated. If greater than the number of blocks in the file, no action occurs. If equal to the number of blocks in the file, the file is purged. For DXCLO, ITRUN is an optional double word variable containing a double integer number of blocks to be deleted from the main file at closing.

IERLC	Optional error condition location (see RFA Call Parameter Variations).

DCONT

The RTE input/output control requests to Type 0 files are transmitted using this routine. DCONT is equivalent to the FMP routine FCONT.

CALLING SEQUENCE

```
CALL DCONT(IDCB,IERR,ICON1[,ICON2[,IERLC]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).
ICON1	Function code; a variable containing a numeric value defining an input/output function.
ICON2	Function sub-code (optional); a variable containing a numeric value. This sub-code is required for some functions (see the appropriate RTE Programmer's Reference Manual).
IERLC	Optional error condition location (see RFA Call Parameter Variations).

Function Code

Bits 6 through 10 of parameter ICON1 are used for the function code.

```
BITS  15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
     +---------------------------------------------------------------+
     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
     | 0 | 0   0   0 | 0         |           | 0   0   0 | 0   0   0 |
     |   |           |           |           |           |           |
     +---------------------------------------------------------------+
```

The function codes are defined in Table 4-1. These function codes are driver dependent and the appropriate RTE driver reference manual should be consulted for more information.

Table 4-1. DCONT Functon Codes.

FUNCTION CODE (OCTAL)	FUNCTION	DEVICE
00	Unused	Magnetic tape Cartridge Tape Unit
01	Write end-of file	Magnetic tape Cartridge Tape Unit
02	Backspace one record	Magnetic tape Cartridge Tape Unit
03	Forward space one record	Magnetic tape Cartridge Tape Unit
04	Rewind	Magnetic tape Cartridge Tape Unit
05	Rewind standby Rewind	Magnetic tape Cartridge Tape Unit
06	Actual device status	Magnetic tape Cartridge Tape Unit
07	Set end-of-tape	Paper tape/TTY
10	Generate leader/ Write end-of-file if not just written or not at load point	Paper tape/TTY Cartridge Tape Unit
*11	List output line spacing	Line printer
12	Write 3 inch inter-record gap	Magnetic Tape
13	Forward space one file	Magnetic tape Cartridge Tape Unit
14	Backspace one file	Magnetic tape Cartridge Tape Unit
15	Conditional top-of-form	Line printer or Display device
20	Enable teminal--allows terminal to schedule its program with any key stroke	Codes 20-27 are defined for a keyboard terminal (DVR00). Refer to the DVR00 manual,29029-60001 for other uses.
21	Disable terminal--inhibits scheduling of terminal program	

Table 4-1. DCONT Function Codes (Continued).

FUNCTION CODE (OCTAL)	FUNCTION	DEVICE
22	Set time-out--sets new time-out interval	
23	Ignore all further requests until: * the request queue is empty * an input request is received * a restore control request is received	
24	Restore output processing (this request is usually not necessary)	
26	Write end-of-data	Cartridge Tape Unit
**27	Locate file number	Cartridge Tape Unit

*When function code 11 is specified in ICON1, then ICON2 must be included in the parameter list to specify the particular line spacing.
**When function code 27 is specified in ICON1, then ICON2 must be included in the parameter list to specify the particular file number

Function Sub-Code

Function sub-codes are required for:

- line spacing, function 11
- finding files, function 27

If the function code is 11 octal, then FCONT expects a value in ICON2. This value controls output line spacing on the line printer or a keyboard display device:

0	to suppress line spacing on the next line
>0	to indicate the number of lines to space before the next line
<0	to page eject on line printer; space specified lines on keyboard device

If the function code is 27 octal, then FCONT expects a value in ICON2. This value declares the absolute file number to be located, in the range 1 through 255.

DCRET, DXCRE

The DCRET routine defines a new file. DCRET is equivalent to the FMP routine CREAT except as noted below. The call to DCRET results in the creation of a named file of a specific number of blocks on a disc of your choice. If the target node is under Session Monitor control, the specified disc must be mounted to your session. Upon successful completion of a call to DCRET, the file created remains open exclusively in update mode to the calling program. A DCRET call automatically opens the newly defined file only for updates by the calling program. For other types of access, you must open the file after creating it by using the DOPEN call. File information is entered into the appropriate File Directory. Type 0 files cannot be created via DCRET.

```
+-------------------------------------------------------------+
|                          NOTE                               |
|                                                             |
|      The DCRET routine does not close an open file          |
|      associated with the DCB. That is, if you already       |
|      have an open file in the specified DCB, you must       |
|      close the file before calling DCRET.                   |
|                                                             |
+-------------------------------------------------------------+
```

CALLING SEQUENCE

```
CALL  DCRET(IDCB,IERR,NAME,ISIZE,ITYPE[,ISECU[,ICR[,IERLC]]])
```

for the extended format version:

```
CALL DXCRE(IDCB,IERR,NAME,ISIZE,ITYPE[,ISECU[,ICR[,JSIZE
          [,IERLC]]]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).
NAME	File name; a 3-word array containing the ASCII-coded name of the file to be created.
ISIZE	File size; a 2-word array in which word 1 contains a positive value declaring the number of blocks to be allocated for this file. Word 2 is used only for Type 2 files. It contains the record length of the file specified in number of words. For DXCRE, ISIZE is a 2-entry array in which each entry is a double word integer. The first entry contains the file size in double word number of blocks. The second entry is used only for Type 2 files and is the double word record length.
ITYPE	File type; an integer variable. See file type listing below.
ISECU	File security code (optional); a variable in the range 0 through + or - 32767. A positive value declares file write protection. A negative value declares file read and write protection. A value of 0 declares no file protection.
ICR	Cartridge reference label (optional); a 2-word array which defines cartridge search and node destination conditions (see RFA Call Parameter Variations).

JSIZE	In DXCRE, an optional double word parameter in which the actual created file size in sectors is returned if DXCRE is successful.
IERLC	Optional error condition location (see RFA Call Parameter Variations).

File Type

Positive integer in range 0 through 32767; default depends upon command.

0	non-disc file
1	fixed length 128-word record
2	fixed length records; user defines length
3	variable length records; sequential access, automatic extents
4	ASCII code and source program (otherwise like type 3 files)
5	relocatable binary code (otherwise like type 3 files)
6	save program file (otherwise like type 1 files)
7	absolute binary (otherwise like type 3 files)
8-32767	user defined

DLOCF, DXLOC

This call is used to retrieve the location of the current record pointer within disc files. In addition to the pointer location, you can retrieve the status of a file, such as the sector count for disc files, the allocated logical unit number for disc and non-disc files, the file type, the record size for Type 2 files, and the read/write access code for Type 0 files. DLOCF is equivalent to the FMP routine LOCF.

CALLING SEQUENCE

```
      CALL DLOCF(IDCB,IERR,IREC[,IRB[,IOFF[,JSEC[,JLU[,JTY
               [,JREC[,IERLC]]]]]]])
```

for the extended format version:

```
      CALL DXLOC(IDCB,IERR,IREC[,IRB[,IOFF[,JSEC[,JLU[,JTY
               [,JREC[,IERLC]]]]]]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).
IREC	Next record; a variable to which a value is returned that represents the number of the next sequential record in the file. For DXLOC, a double word variable containing the next sequential record number.
IRB	Relative block number containing the next record (optional); a variable to which the number of the next block is returned. For Type 0 files, nothing is returned. For Type 1 files, IRB=IREC-1. If the file is extended, extents are accounted for within the value returned. For DXLOC, IRB is a double word variable with the same meaning.

IOFF	Offset of next record in the block (optional); a variable to which the location of the next word within the record is returned. For Type 0 files, nothing is returned.
JSEC	File size in sectors (optional); a variable to which the number of sectors allocated to the file at creation is returned. To determine the block count, divide the sector count by 2. For Type 0 files, nothing is returned. For DXLOC, JSEC is a double word variable.
JLU	Logical unit number (optional); a variable to which the logical unit number of the device upon which the file resides is returned.
JTY	File type (optional); a variable to which the file type determined at opening is returned.
JREC	Record size (optional); a variable to which the record length is returned for Type 1 and Type 2 files; or the read/write access code is returned for Type 0 files. This parameter is not applicable to a file type equal to or greater than Type 3.
IERLC	Optional error condition location (see RFA Call Parameter Variations).

DNAME

Renames an existing file. If the original file has a security code, the code must be supplied in the call to DNAME. The DNAME routine is equivalent to the FMP routine NAMF. If the target node is a Session Monitor node, the file must reside on a disc mounted to your session.

```
+-----------------------------------------------------------+
|                          NOTE                             |
|                                                           |
|      The DNAME routine does not close an open file       |
|      associated with the DCB. That is, if you already    |
|      have an open file in the specified DCB, you must    |
|      close it before calling DNAME.                      |
|                                                           |
+-----------------------------------------------------------+
```

CALLING SEQUENCE

```
CALL DNAME(IDCB,IERR,NAME,NNAME[,ISECU[,ICR[,IERLC]]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).
NAME	File's current name; a 3-word array containing the ASCII-coded current name of the file.
NNAME	File's new name; a 3-word array containing the ASCII coded new name of the file.
ISECU	File security code (optional); a variable in the range 0 through + or - 32767. This parameter must be declared if the file specified in the NAME parameter was created having a security code.
ICR	Cartridge reference label (optional); a 2-word array which defines cartridge search and node destination conditions (see RFA Call Parameter Variations).
IERLC	Optional error condition location (see RFA Call Parameter Variations).

DOPEN

You may open defined files to access by your programs using the DOPEN call. DOPEN is equivalent to the FMP call OPEN. Opening a file establishes a logical relationship between the file's entry in the file directory and the Data Control Block for the file. If the target node is under Session Monitor Control, the file must reside on a disc mounted to your session.

```
+---------------------------------------------------------------+
|                            NOTE                               |
|                                                               |
|         The DOPEN routine does not close an open file         |
|         associated with the DCB. That is, if you already      |
|         have an open file in the specified DCB, you must      |
|         close it before calling DOPEN.                        |
|                                                               |
+---------------------------------------------------------------+
```

CALLING SEQUENCE

```
      CALL DOPEN(IDCB,IERR,NAME[,IOPTN[,ISECU[,ICR[,IERLC]]]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).
NAME	File name; a 3-word array containing the ASCII-coded name of the file to be opened.
IOPTN	File access specifications (optional); a variable containing a numeric value that specifies non-standard conditions upon opening a file (see File Open Options below).

ISECU	If the file was created (see DCRET call description) with a non-zero security code, this parameter must be specified and it must match the original security code assigned to the file.
ICR	Cartridge reference label (optional); a 2-word array which defines cartridge search and node destination conditions (see RFA Call Parameter Variations).
IERLC	Optional error condition location (see RFA Call Parameter Variations).

File Open Options

If the IOPTN parameter is set to zero in the DOPEN routine, the file is opened by default to the following specifications:

1. Exclusive use - the file can be accessed only by the calling program at a specific node.

2. Standard sequential output - each record is written following the previous record, destroying any data beyond the current file position.

3. The file type defined at creation is used for access.

To open a file with other options, the IOPTN parameter is defined as follows:

BITS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						X	A	K	V	M			F	T	U	E

The following bits may be set for any file type:

E (bit 0)	= 0	File opened exclusively for this program
	= 1	File may be shared by up to seven programs

U (bit 1)	= 0	File opened for standard (non-update) write
	= 1	File opened for update
T (bit 2)	= 0	Use file type defined at creation
	= 1	File type is forced to type 1

The following bits are used for type 0 files only (they are ignored when opening other file types):

F (bit 3)	= 0	Use function code defined at creation
	= 1	Use function code defined in bits 6 - 10 of IOPTN

Bits 6 - 10 correspond exactly to the function code used for the RTE READ/WRITE EXEC call. These function codes are driver dependent and the appropriate RTE driver reference manual should be consulted for more information.

M (bit 6)	= 0	ASCII data
	= 1	Binary data
V (bit 7)	= 0	If M = 0 (ASCII) use column 1 for carriage control on line printer.
	= 1	If M = 0 (ASCII) print column 1 on line printer
K (bit 8)	= 0	Keyboard input is not printed
	= 1	Keyboard input printed as received

Exclusive Open

By default, a file is opened for exclusive use of the calling program. An exclusive open is granted to only one program at a time. If the call is rejected because the file is open to another program, you must make the call again; it is not stacked. Exclusive open is useful in order to prevent one or more programs from destructively interferring with each other.

Non-Exclusive Open

If more than one program needs to access the file, it should be opened non-exclusively by setting the IOPTN E bit. A non-exclusive open may be granted to as many as seven programs per file at one time. A non-exclusive open will not be granted if the file is already opened exclusively. Each time an open is requested for the file, all programs currently having the file open are checked. If any program is dormant, the file is closed to that program. That type of close does not free the DCB and does not post the contents of the DCB buffer to the file. Any open flag will also be cleared if it does not point to a valid ID segment (possible if the file was left open on a different system).

Update Open

In update mode, IOPTN U bit set, the block containing the record to be written is read into the DCB buffer before it is modified. This insures that existing records in the block will not be destroyed. This mode of open has no effect on reading or positioning. File type governs the choice between update and standard (non-update) open.

Update mode should be used to write to type 2 files. A type 2 file should be opened in standard mode only when originally writing the file or adding to the end of the file, and then only if it is to be written sequentially.

Update mode is ignored for type 1 files. Although, like type 2, they are designed for random access with fixed length records and the end-of-file in the last word of the last block. Each record is the same length as the block transferred so that there is no danger of writing over existing records.

For type 3 and above files, update mode is not generally used; most writes are sequential with an end-of-file mark written after each record. These files should be opened for update only if a record previously written to the file is being modified. In this case, care must be taken not to change the length of the modified record. If it is changed, a -005 error is issued. Regardless of the mode of open (update or standard), a record written beyond the end-of-file replaces the end-of-file and is followed by a new end-of-file.

Type 1 Access

Any file may be forced to type 1 access by setting the IOPTN T bit. Type 1 access is faster because it bypasses the Data Control Block buffer and transfers a sector of data directly to the user buffer defined as IBUF. The file type defined at creation is not affected; the file is treated as type 1 only for the duration of this open. You are responsible for any packing or unpacking of records in files forced to type 1. That is, if the records are less than 128 words, you must determine the start and end of each record.

DPOSN, DXPOS

This call is used to position any type of file to a point relative to its current position or to a specific record number. DPOSN is equivalent to the FMP routine POSNT.

CALLING SEQUENCE

```
    CALL DPOSN(IDCB,IERR,NUR[,IR[,IERLC]])

 for the extended format version:

    CALL DXPOS(IDCB,IERR,NUR[,IR[,IERLC]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Variations).
IERR	Error return; a variable (see RFA Call Variations).
NUR	Record count or record number; a variable containing either the number of records to skip forward (positive value) or backward (negative value), or the absolute record number to which the file is to be positioned (positive integer only). The IR parameter determines how NUR is interpreted. For DXPOS, NUR is a double word variable with the same meaning.
IR	A variable (optional) that, if zero, declares that NUR is the number of records to skip; if IR is not zero, NUR is interpreted as an absolute record number.
IERLC	Optional error condition location (see RFA Call Variations).

DPURG

To purge a file (remove it from the file directory), you may issue a call to DPURG. DPURG is equivalent to the FMP call PURGE. When called, DPURG sets a flag in the file's directory entry to indicate that the file is no longer accessible. If the target node is under Session Monitor control, the file must reside on a disc mounted to your session.

CALLING SEQUENCE

```
CALL DPURG(IDCB,IERR,NAME[,ISECU[,ICR[,IERLC]]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).
NAME	File name; a 3-word array containing the ASCII-coded name of the file to be purged.
ISECU	If the file was created (see DCRET call description) with a non-zero security code, this parameter must be specified and it must match the original security code assigned to the file.
ICR	Cartridge reference label (optional); a 2-word array which defines cartridge search and node destination conditions (see RFA Call Parameter Variations).
IERLC	Optional error condition location (see RFA Call Parameter Variations).

DREAD, DXREA

This routine reads data from your file (currently open to the Data Control Block) into your program's data buffer. DREAD is equivalent to the FMP routine READF.

CALLING SEQUENCE

```
      CALL DREAD(IDCB,IERR,IBUF,IL[,LEN[,NUM[,IERLC]]])

   for the extended format version:

      CALL DXREA(IDCB,IERR,IBUF,IL[,LEN[,NUM[,IERLC]]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).
IBUF	Data buffer; an array, the size equal to or greater than the value of the IL parameter, into which the requested data is placed by the system.
IL	Data length; a variable specifying the number of words to be read (see RFA Call Parameter Variations).
LEN	Actual length (in words) of data read (optional); a variable to which the actual count of words transferred is returned; set to -1 if end-of-file is read.
NUM	Record number (optional); a variable containing the record number from which data is to be transferred. Meaningful only for Type 1 and Type 2 files (see the appropriate RTE Programmer's Reference Manual). For DXREA, NUM is an optional 2-word variable < 32767.
IERLC	Optional error condition location (see RFA Call Parameter Variations).

DSTAT

This routine is called to obtain status information for all mounted cartridges in the cartridge directory. The information returned for each cartridge includes the cartridge logical unit number, the last FMP track, cartridge reference number, and if a program has locked the cartridge, the programs ID segment address. DSTAT is equivalent to the FMP routine FSTAT.

Under Session Monitor, ISTAT will contain information about only those discs mounted to the session's (default or specified) Session Control Block, and system discs. You may get information on all discs mounted to the system by specifying IOP non-zero. In addition to the information returned as stated above, the identification of the user who mounted the cartridge (user or group ID) is also returned.

CALLING SEQUENCE

```
CALL DSTAT(ISTAT,IERR,IDEST[,IERLC[,ILEN[,IFORM[,IOP[,IADD]]]]])
```

ISTAT	Status buffer; an array to which the cartridge directory status is returned. Default is 125 words.
IERR	Error return; a variable (see RFA Call Parameter Variations).
IDEST	Address of destination node; a variable that contains a decimal value specifying the nodal address for which the status information is to be obtained. The value of this parameter may either be positive to denote the node number or negative to denote the communications line logical unit number.
IERLC	Optional error condition location (see RFA Call Parameter Variations).
ILEN	Length in words of buffer ISTAT. Default is 125 words.
IFORM	If zero, the disc directory will be written in FORMAT I (default). If non-zero, the disc directory will be written into the buffer in FORMAT II. This is exactly as it appears on disc.

IOP An optional one-word variable specifying the type of cartridges that information is to be returned about.

If = 1, all discs mounted to the system are returned in ISTAT, whether under session control or not.

If = 0, and under session control, private and group discs mounted to the session and system discs are returned in that order.

If = 0, and not under session control, system discs and non-session discs are returned in that order.

IADD Returned non-zero by the system if not all of the cartridge list could be returned in ISTAT (ISTAT not large enough).

The two formats that can be used with the DSTAT call are shown below:

ISTAT FORMAT I

WORD	CONTENTS	CARTRIDGE
1	Logical Unit Number	First Cartridge
2	Last FMP track	
3	Cartridge Reference Number	
4	Lock Word	
5	Logical Unit Number	Second Cartridge
6	Last FMP track	
7	Cartridge Reference Number	
8	Lock Word	
9	Logical Unit Number	Third Cartridge
.	.	
.	.	
.	.	
	0 = no more discs	

where:

Lock Word is ID segment address of locking program or 0 (not locked).

ISTAT FORMAT II

WORD	CONTENTS		CARTRIDGE
1	Lock word	Logical Unit #	First Cartridge
2	Last FMP track		
3	Cartridge Reference Number		
4	ID		
5	Lock word	Logical Unit #	Second Cartridge
6	Last FMP track		
7	Cartridge Reference Number		
8	ID		
9	Lock word	Logical Unit #	Third Cartridge
.	.		
.	.		
.	.		
	0 = no more discs		

where:

Lock Word is the offset of the ID segment in the Keyword Table or 0 (not locked). ID identifies who mounted the cartridge (user or group ID).

DWIND

This call may be used to rewind (position to beginning of first record) a Type 0 file, or set disc files so that the next record in the file is the first record (positions to beginning of first record). DWIND is equivalent to the FMP routine RWNDF.

CALLING SEQUENCE

```
    CALL DWIND(IDCB,IERR[,IERLC])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).
IERLC	Optional error condition location (see RFA Call Parameter Variations).

DWRIT, DXWRI

This routine writes data from a data buffer to your file (currently open to the Data Control Block). DWRIT is equivalent to the FMP routine WRITF.

CALLING SEQUENCE

```
   CALL DWRIT(IDCB,IERR,IBUF,IL[,NUM[,IERLC]])

for the extended format version:

   CALL DXWRI(IDCB,IERR,IBUF,IL[,NUM[,IERLC]])
```

IDCB	Data Control Block; a 4-word array (see RFA Call Parameter Variations).
IERR	Error return; a variable (see RFA Call Parameter Variations).
IBUF	Data buffer; an array of a size equal to or greater than the value of the IL parameter.
IL	Data length; a variable (see RFA Call Parameter Variations).
NUM	Record number (optional); a variable containing the record number to which data is to be transferred. Meaningful only for Type 1 and Type 2 files (see the appropriate RTE Programmer's Reference Manual). For DXWRI, NUM is an optional 2-word variable up to (2**31)-1.
IERLC	Optional error condition location (see RFA Call Parameter Variations).

Chapter 5 DS/1000-IV DEXEC Calls

DS/1000-IV Remote Executive Module

DS/1000-IV Remote Executive (DEXEC) is a subroutine which supports RTE EXEC requests. It provides you with the ability to pass control of your processing to the EXEC module of the operating system. The DS/1000-IV DEXEC calls are the remote processing equivalent to the RTE EXEC calls. That is, you can pass control of your remote processing to the EXEC module of a local or remote operating system via a call to the DEXEC subroutines.

DEXEC calls must always be used for remote processing. For local processing, you may use either EXEC or DEXEC calls. However, use of DEXEC calls is recommended to facilitate the possibility of later moving your program to a remote environment.

NOTE: DEXEC calls are not supported in RTE-6/VM type 6 (Extended Background) programs.

DS/1000-IV DEXEC Call Considerations

The philosophy behind the implementation of DEXEC calls is to ensure that a parameter's position and meaning stay the same for the system on which the EXEC call is executed. Thus, it is suggested that you consult the RTE manual for the system at which you wish to execute the EXEC call. For example, some EXEC call formats on RTE-L, RTE-XL and RTE-A are different from the RTE-IVB, RTE-IVE, RTE-MIII and RTE-6/VM formats. To perform an EXEC call at an RTE-L, XL or A from an RTE-IVB, IVE, MIII or 6/VM system, use the parameter's position and meaning as described in the appropriate RTE-L, XL or A manual. Also, all EXEC calls supported on a local system are available via a local DEXEC call (specifying that local node).

Parametric compatibility is maintained between DEXEC calls and EXEC calls with one additional parameter. DEXEC calls include a "destination node" parameter, IDEST, that declares the network node at which the call to DEXEC is to be processed. The IDEST parameter may be either a positive value declaring the destination node number, or a negative value declaring the communication line logical unit number.

A value of negative one (-1) always indicates the local node.

The information returned in the A-register and B-register by DEXEC is the same as that returned by EXEC (refer to the appropriate Operating System reference manual) with additional DS/1000-IV error conditions and codes. These error conditions and codes are listed in the appendix of this manual.

For remote execution of program calls to DEXEC, it is suggested that you use the "no abort" feature provided by RTE operating systems. Without this feature, when severe errors are encountered, the operating system will automatically abort your program. You enable the no abort feature by setting bit 15 of the ICODE parameter word to 1. This feature is described in the EXEC call sections under the heading "Error Return Point" of the appropriate Operating System reference manual furnished with your system.

When accessing remote Session Monitor nodes via DEXEC calls, your program will be using either a specified or a default session. This session can be obtained by:

1. Using the programmatic logon to a remote session (DLGON) (Refer to the Utilities and Program Calls Chapter of this manual.)

2. Using REMAT and establishing an account at the remote node (using the REMAT ATtach command)

3. Using only the DEXEC call, where a default session will be established for the program at the remote Session Monitor node.

Accessing a remote Session Monitor node through a specified or default session account will restrict the program to the resources of that account. Thus, the "Session LUs" in the account's Session Switch Table (SST) are the only LUs the program can access.

Note that the SST for sessions created from another node do not include any Configuration Table LU definitions because remote sessions have no "station specific" LUs.

If access to system LUs (rather than session LUs) is desired and you know the correct password, use the utility DLGNS (or the REMAT AT command) to access the node outside of session.

The DS/1000-IV DEXEC call syntax is outlined in the following paragraphs. Refer to the EXEC calls in the RTE Programming and Operating Manuals furnished with your system for detailed information about the parameters.

The DEXEC request codes which are supported for remote execution are ICODEs:

1, 2, 3, 6, 9, 10, 11, 12, 13, 23, 24, 25, and 99

All other ICODEs will be rejected by DEXEC with error code DS06(1). All defined EXEC request codes are executable when IDEST, the destination node number, specifies the local node.

NOTE: Request codes 6 and 25 are not supported on RTE-L/XL/A. On RTE-A systems with multiuser environment, you must load programs that are controlled by DEXEC 9, 10, 23, 24 12 or 99 as system utilities.

DEXEC 1 — Remote Read

A call to DEXEC with request code 1 results in the transfer of one record from an I/O device to a buffer. Further, you can perform an "interactive write/read" operation via the control word parameter and the two optional parameters.

The interactive write/read option is useful for those programs that must communicate with an operator using a question/answer format for the exchange of information. Unlike other DEXEC calls, it is translated into two EXEC calls at the destination node, a write followed by a read. Because a write/read operation is configured as a single request, only two communication line operations (a request and a reply) are needed to process the request. Separate write and read requests would require four communication line operations to process the same information. Another advantage of a write/read request is that the write portion of the request is immediately followed by the read portion, with no intervening operations. This prevents annoying delays between questions from the system and your responses.

The remote execution of the interactive program EDITR is one example of the interactive write/read operation. The current pending line and the EDITR prompt character (/) are transmitted to your remote terminal as the write portion of the request. Once the prompt is displayed, the read portion of the request is issued to obtain your EDITR command response.

The interactive write/read operation uses the same buffer for write and read. This buffer is not initialized between the two operations; the response to a question simply overlays the question. In instances where the response has shorter length than the question, both the response and the portion of the question that has not been overlayed are returned. It is therefore suggested that the transmission log be examined to determine the number of characters returned in the response.

NOTE: HP does not support Read or Write requests that reference a disc LU.

CALLING SEQUENCE

For RTE-IVB, RTE-IVE, RTE-MIII and RTE-6/VM:

```
CALL DEXEC(IDEST,ICODE,ICNWD,IBUFR,IBUFL[,IPRM1[,IPRM2]])
```

For RTE-L, RTE-XL, and RTE-A:

```
CALL DEXEC
    (IDEST,ICODE,ICNWD,IBUFR,IBUFL[,IPRM1[,IPRM2[,0,0,KEYWD]]])
```

IDEST The destination node address where the call is executed. A value of -1 indicates the local node. Local execution will also be performed whenever IDEST equals the local node number.

ICODE Read request code = 1

ICNWD Control word. Unlike an EXEC 1 call, bit 11 of ICNWD cannot be used to specify control information to the driver in RTE-A, L and XL systems. Refer to the "Interactive Write/Read" call description for more information on this restriction. (See the EXEC call description in the appropriate Operating System and Device Driver Reference Manuals for an explanation of this parameter.)

```
+---------+
| CAUTION |
+---------+
```

Do not specify the logical unit number of any DS/1000-IV communication line in the control word. Doing so may destroy the integrity of your network.

IBUFR Read buffer address.

IBUFL Read buffer length (+ for words, - for characters). The maximum buffer size for a remote DEXEC read operation is 512 words (-1024 characters). If the Z-bit=1 (indicating double buffering), the combined buffer length, IBUFL + IPRM2, must be less than or equal to 512 words.

IPRM1 Optional parameter (see the EXEC call descriptions in the appropriate Operating System reference manual for an explanation of this parameter). If the Z-bit = 1 (indicating double buffering), this parameter specifies the address of the second buffer.

IPRM2 Optional parameter: (See the EXEC call descriptions in the appropriate Operating System reference manual for an explanation of this parameter.) If the Z-bit=1 (indicating double buffering), this parameter specifies the length of the second buffer (+ for words, - for characters). The combined buffer length, IBUFL + IPRM2, must be less than or equal to 512 words.

0,0 Formal place holders which must be included whenever KEYWD is specified.

KEYWD Optional parameter: the locked LU's keyword number.

Interactive WRITE/READ Request

CALLING SEQUENCE

For RTE-IVB, RTE-IVE, RTE-MIII, and RTE-6/VM:

```
CALL DEXEC(IDEST,ICODE,ICNWD,IBUFR,IBUFL[,IPRM1[,IPRM2]])
```

For RTE-L, RTE-XL, and RTE-A:

```
CALL DEXEC
    (IDEST,ICODE,ICNWD,IBUFR,IBUFL[,IPRM1[,IPRM2[,0,0,KEYWD]]])
```

IDEST The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.

ICODE Read request = 1

ICNWD Control word. Set bit 11 = 1 to indicate interactive write/read; define read buffer in IBUFR/IBUFL and write buffer in IPRM1/IPRM2. Bit 12 must equal zero. Interactive write/read and double buffering are mutually exclusive.

NOTE: Because bit 11 is used to indicate a DEXEC interactive write/read, this bit cannot be used to send control information to the driver in RTE-A, L and XL systems. (Bit 11 can be specified to the driver in DEXEC 2 and 3 calls.)

```
+----------+
| CAUTION |
+----------+
```

Do not specify the logical unit number of any DS/1000-IV communication line in the control word. Doing so may destroy the integrity of your network.

IBUFR	Read buffer address.
IBUFL	Read buffer length (+ for words, - for characters). The maximum buffer size for a remote DEXEC read operation is 512 words (-1024 characters).
IPRM1	Write buffer address.
IPRM2	Write buffer length (+ for words, - for characters). [write buffer < read buffer]
0,0	Formal place holders which must be included whenever KEYWD is specified.
KEYWD	Optional parameter: the locked LU's keyword number.

DEXEC 2 — Remote Write

A call to DEXEC with request code 2 results in the transfer of one record from a buffer to an I/O device.

CALLING SEQUENCE

For RTE-IVB, RTE-IVE, RTE-MIII, and RTE-6/VM:

```
CALL DEXEC(IDEST,ICODE,ICNWD,IBUFR,IBUFL[,IPRM1[,IPRM2]])
```

For RTE-L, RTE-XL, and RTE-A:

```
CALL DEXEC
     (IDEST,ICODE,ICNWD,IBUFR,IBUFL[,IPRM1[,IPRM2[,0,0,KEYWD]]])
```

IDEST	The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.
ICODE	Write request code = 2.
ICNWD	Control word.

```
+----------+
| CAUTION |
+----------+
```

Do not specify the logical unit number of any DS/1000-IV communication line in the control word. Doing so may destroy the integrity of your network.

IBUFR	Write buffer address.
IBUFL	Write buffer length (+ for words, - for characters). The maximum buffer size for a remote DEXEC write operation is 512 words (-1024 characters). If the Z-bit=1 (indicating double buffering), the combined buffer length, IBUFL + IPRM2, must be less than or equal to 512 words.

IPRM1	Optional parameter (see the EXEC call descriptions in the appropriate Operating System reference manual for an explanation of this parameter). If the Z-bit = 1 (indicating double buffering), this parameter specifies the addresss of the second buffer.
IPRM2	Optional parameter. (See the EXEC call descriptions in the appropriate Operating System reference manual for an explanation of this parameter.) this parameter specifies the length of the second buffer (+ for words, - for characters). The combined buffer length, IBUFL + IPRM2, must be less than or equal to 512 words.
0,0	Formal place holders which must be included whenever KEYWD is specified.
KEYWD	Optional parameter: the locked LU's keyword number.

DEXEC 3 — Remote I/O Device Control

DEXEC request code 3 performs remote I/O control operations such as backspace, write end-of-file, rewind, and so forth. Remember that, when using DEXEC to a Session Monitor destination node, the program has access only to the resources available to the default account at that node unless a specific logon call is made. See DEXEC Call Considerations at the beginning of this chapter.

CALLING SEQUENCE

For RTE-IVB, RTE-IVE, RTE-MIII, and RTE-6/VM

```
CALL DEXEC(IDEST,ICODE,ICNWD[,IP1])
```

For RTE-L, RTE-XL, and RTE-A:

```
CALL DEXEC
   (IDEST,ICODE,ICNWD[,IP1[,IP2[,IP3[,IP4[,0,0,KEYWD]]]]])
```

IDEST	The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.
ICODE	I/O control request code = 3.
ICNWD	Control word.
IP1-IP4	Optional parameters (required for list output line spacing and various other functions).
0,0	Formal place holders which must be included whenever KEYWD is specified.
KEYWD	Optional parameter: the locked LU's keyword number.

NOTE: DEXEC 3 calls to HP-IB devices which do double buffering are not supported. Use PTOP to implement these HP-IB calls.

DEXEC 6 — Remote Program Termination

Using this DEXEC call (request code 6) your program can inform an RTE operating system that it wants to terminate execution of itself or another program. Because the EXECM or EXECW module is actually the father program for all remote scheduling, a program scheduled via a DEXEC request can be terminated from any other program via a DEXEC termination request. (NOTE: DEXEC 6 will only terminate programs that were originally scheduled with a DEXEC 9, 10, 23 or 24 call and is not supported on RTE-L and RTE-XL.)

CALLING SEQUENCE

```
    CALL DEXEC(IDEST,ICODE,INAME,INUMB)
```

If INAME = 0, indicating a program is to terminate itself (local operation only), optional parameters can be used as shown.

```
    CALL DEXEC(IDEST,ICODE,INAME,INUMB
                    [,IOP1[,IOP2[,IOP3[,IOP4[,IOP5]]]]]
```

IDEST	The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.
ICODE	Request code = 6.
INAME	A 3-word integer array containing the program name to be terminated in the first five bytes (the sixth byte is not significant). INAME can be the value zero which causes a program to terminate itself (local operations only). Contrary to usage in an EXEC call, this parameter is not optional; it must be specified.

INUMB Completion type code. Contrary to usage in an EXEC call, this parameter is not optional; it must be specified. The completion type codes follow (note that only 0 and 1 can be executed remotely because all remotely scheduled programs are sons of either EXECM or EXECW.):

-1	serial reusability completion. When rescheduled, program is not reloaded if it remained resident in memory. If this call is a "father" program's request, it is the same as INUMB = 0.
0	normal completion.
1	place program in dormant state; save current suspension point and resources.
2	terminate and remove named program from the time list. If program is in I/O suspend state, system waits until I/O completes before terminating program; however, this call does not wait. The program's disc tracks are not released.
3	terminate named program immediately, remove it from the time list, and release program's disc tracks. If program is in I/O suspend state, a system-generated clear request is issued to the driver. An abort message is printed at the system console.

IOP1-
IOP5 Up to five optional parameters passed to the caller the next time the program executes. (INAME = 0 only.)

DEXEC 9, 10, 23, 24 - Remote Program Schedule

This call to DEXEC (request codes 9, 10, 23, or 24) schedules a dormant program for execution. For request codes 9 and 10, the program to be scheduled must be dormant. For request code 24, the calling program will suspend execution until the scheduled program becomes dormant (at which time the scheduled program is started and the calling program's execution resumes). Request codes 9 and 23 will not allow resumption until the scheduled program completes. Optionally, you may pass up to five parameters plus an optional buffer which can be an ASCII-coded data string.

NOTE: If program-to-program communication is desired, use the PTOP calls. These calls are covered in the PTOP Chapter of this manual.
On RTE-A systems with multiuser environment, you must load programs scheduled by DEXEC 9, 10, 23, or 24 as system utilities.

CALLING SEQUENCE

```
      CALL DEXEC(IDEST,ICODE,INAME[,IPRM1[,IPRM2[,IPRM3[,IPRM4
                     [,IPRM5,IBUFR,IBUFL]]]]])
```

IDEST The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.

ICODE	Schedule request code =	9	(immediate schedule, wait)
		10	(immediate schedule, no wait)
		23	(queue schedule, wait)
		24	(queue schedule, no wait)

INAME A 3-word integer array containing the program name to be scheduled in the first five bytes (the sixth byte is not significant). This parameter must reference an ASCII-coded program name. If a value of zero is referenced for a remote schedule call or if the program named does not exist, an error results.

IPRM1-
IPRM5 Up to five optional one-word parameters may be passed to the named program.

IBUFR	Optional buffer address.
IBUFL	Optional buffer length (+ for words, - for characters). The maximum length of this string buffer is 512 words (-1024 characters). If IBUFR is specified, IBUFL is NOT optional.

Network timing constraints require that the remote execution of programs scheduled with wait must be completed in less than 20 minutes. Exceeding this time limit results in the return of the request timeout error, DS05(0). If a request timeout error occurs, you will not be able to recover returned parameters and the program may be aborted at the node where it is scheduled. Further, only one program at a time can be scheduled with wait at a remote node. Attempts to schedule more than one program with wait at a remote node will result in subsequent programs waiting for the execution of the first program to be completed. The second program's timing begins when it is scheduled; the wait time is included in the 20 minute time limit.

Remote Scheduling

Programs scheduled at a remote session monitor node will run under a specific or default account, unless non-session access is used. Remote scheduling can be done with DEXEC(9) or DEXEC(23) (with wait) and DEXEC(10) or DEXEC(24) (without wait). Any program scheduled remotely will be aborted if the master session is terminated.

Scheduling a Copy of a Program

If a program is being scheduled-with-wait (and ONLY with wait) in a remote Session Monitor node, the user can specify in the DEXEC(9) or DEXEC(23) calls to have the program duplicated and renamed (cloned) for an exclusive copy. This is done by setting bit 11 of the ICODE parameter to 1. This option is ignored if the destination node is the local node.

For example, to schedule program INAME at node IDEST with the no-abort and rename options, use:

```
   CALL DEXEC(IDEST,9+104000B,INAME)
                   or
   CALL DEXEC(IDEST,23+104000B,INAME)
```

When the program is renamed, the last two characters will be replaced with ".A", ".B", ".C", etc. until a name is found to be unique.

If the program is known to the RTE system, a copy of the program is attempted. If the program cannot be duplicated, the original is scheduled. If the program is not known to the RTE system an attempt is made to open the original "SP'd" program file. If the file does not exist, an "SC05" error is returned to the caller. If the file exists, it is programmatically "RP'd" to the duplicated ID segment and scheduled. If the "SP'd" program has the "don't clone" bit set, you will get the original program when "RP'd". An error on the "RP" operation will also result in an "SC05" error. If the DEXEC call is directed to the local node, the cloning option is ignored.

Re-Directing Master Requests to Originator

Programs scheduled-with-wait in a remote slave session or non-session node may issue master calls back to the originating node. If the user at the originating node is running under a session, master calls issued by the remote program will by default (if no log-on to another user-name is supplied) be directed back to the originating session. For example, if the remote program issues a DEXEC write to session LU 1 at the originator's node, EXECM will attach to the originator's session and LU 1 will be mapped through the SST to the user's terminal LU.

Sons of the remote program will default to the originating session established for the father remote session.

DEXEC 11 — Remote Time Request

You can use this DEXEC (request code 11) to obtain the current time from the real-time clock at a specific node within your network.

CALLING SEQUENCE

```
      CALL DEXEC(IDEST,ICODE,ITIME[,IYEAR])
```

IDEST	The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.
ICODE	Remote time request code 11.
ITIME	Time value array (5 words).

```
ITIME(1) = 10s of milliseconds
ITIME(2) = seconds
ITIME(3) = minutes
ITIME(4) = hours
ITIME(5) = day of the year
```

IYEAR	Optional year value (1 word).

DEXEC 12 — Remote Timed Program Schedule

This DEXEC call (request code 12) schedules a program for execution at specific time intervals. Execution may be scheduled either at an absolute start time or following a specified initial offset value.

NOTE: On RTE-A systems with multiuser environment, you must load programs scheduled by DEXEC 12 as system utilities.

Absolute Start Time

CALLING SEQUENCE

```
    CALL DEXEC(IDEST,ICODE,INAME,IRESL,MTPLE,IHRS,MINS,ISECS,MSECS)
```

IDEST The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.

ICODE Remote timed schedule request code 12

INAME A 3-word integer array containing the program name to be added to the time list in the first five bytes (the sixth byte is not significant). This parameter must reference an ASCII-coded program name. If a value of zero is referenced for a remote schedule call, if the program named does not exist, or if the program named is not dormant, an error results.

IRESL Resolution code.

1 = 10s of milliseconds
2 = seconds
3 = minutes
4 = hours

MTPLE Execution multiple.

IHRS,MINS,ISECS,MSECS Absolute start time in hours, minutes, seconds, and tens of milliseconds on a 24-hour clock.

Initial Offset Time

CALLING SEQUENCE

```
CALL DEXEC(IDEST,ICODE,INAME,IRESL,MTPLE,IOFST)
```

IDEST	The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.
ICODE	Remote timed schedule request code 12.
INAME	A 3-word integer array containing the program name to be added to the time list in the first five bytes (the sixth byte is not significant). This parameter must reference an ASCII-coded program name. If a value of zero is referenced for a remote schedule call, an error results.
IRESL	Resolution code. 1 = 10s of milliseconds 2 = seconds 3 = minutes 4 = hours
MTPLE	Execution multiple.
IOFST	A negative value that declares the initial execution offset time based upon the content of the IRESL (resolution code) parameter.

DEXEC 13 — Remote I/O Status

You can use this DEXEC call (request code 13) to obtain the status and type of a device identified by a logical unit number.

CALLING SEQUENCE

For RTE-IVB, RTE-IVE, RTE-MIII, and RTE-6/VM:

```
CALL DEXEC(IDEST,ICODE,ICNWD,ISTA1[,ISTA2[,ISTA3]])
```

For RTE-L, RTE-XL, and RTE-A:

```
CALL DEXEC(IDEST,ICODE,ICNWD,ISTA1[,ISTA2[,ISTA3[,ISTA4]]])
```

IDEST	The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.
ICODE	Remote I/O status request code 13.
ICNWD	Control word containing the LU of the I/O device.
ISTA1	First status word. For RTE-IVB, IVE, MIII and 6/VM: word 5 of the appropriate EQT entry is returned here. For RTE-L, XL and A: Device Table word 6 is returned here.
ISTA2	Second status word (optional). For RTE-IVB, IVE, MIII, and 6/VM: word 4 of the appropriate EQT entry is returned here For RTE-L, XL, and A: Interface Table word 6 is returned here.

ISTA3 Third status word (optional).

For RTE-IVB, IVE, MIII and 6/VM: logical unit up/down flag and subchannel number are returned here.

For RTE-L, XL, and A with Z-bit = 0: driver parameter word 1 is returned here.

For RTE-L, XL, and A with Z-bit = 1: buffer address for return of driver information is specified here.

ISTA4 Fourth status word (optional).

For RTE-L, XL and A with Z-bit = 0: driver parameter word 2 is returned here.

For RTE-L, XL and A with Z-bit = 1: buffer length for return of driver information is specified here.

When the Z-bit = 1, ISTA4 length specification must be less than or equal to 512 words.

DEXEC 25 — Remote Partition Status

This call obtains the current status of a specific partition. The information returned includes the starting page number, total number of pages, and status of the partition. DEXEC 25 requests are not supported on RTE-L/XL/A.

CALLING SEQUENCE

```
      CALL DEXEC(IDEST,ICODE,IPART,IPAGE,IPNUM,ISTAT)
```

IDEST	The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.
ICODE	Remote partition status request code 25
IPART	A decimal value that declares the partition number whose status is desired.
IPAGE	The starting page number (plus 1) of the partition is returned here. This value represents the ordinal page number (i.e., first, second, etc.). A zero is returned if IPART contains an invalid partition number.
IPNUM	The size of the user area available in the partition (in pages, minus 1 for the base page) is returned here. A value of -1 is returned if IPART contains an invalid partition number.

ISTAT The status of the partition is returned here. Upon return, the parameter ISTAT has the following status format:

```
bits  15 14                       7                    0
     +---------------------------------------------------+
     |PR|RT|         0            |      ID Segment      |
     +---------------------------------------------------+
```

```
PR = 0  if partition is reserved for programs requesting
        it.
   = 1  if partition is not reserved.

RT = 0  for a real-time partition.
   = 1  for a background partition.

ID Segment = 0 if no program resides in partition.
           = an index value pointing to the keyword
             table for the program's ID segment if a
             program does reside in the partition.
```

> **NOTE**
>
> The content of ISTAT bits 7 - 0 (ID segment) is meaningful only for calls referencing the local node.

DEXEC 99 — Remote Program Status

A call to DEXEC with request code 99 results in the return of status information for a specific program. The program status codes are defined in the appropriate Operating System Reference Manual. DEXEC(99) is supported in DS/1000-IV only. There is no RTE equivalent.

NOTE: On RTE-A systems with multiuser environment, you must load programs controlled by DEXEC 99 as system utilities.

CALLING SEQUENCE

```
    CALL DEXEC(IDEST,ICODE,INAME[,ISTAT])
```

IDEST The destination node address where the call is executed. A value of -1 indicates local node. Local execution will also be performed whenever IDEST equals the local node number.

ICODE Remote program status code 99

INAME A 3-word integer array containing the program name for which status is requested in the first five bytes (the sixth byte is not significant). This parameter must reference an ASCII-coded program name. If a value of zero is referenced for a remote program status call, an error results.

ISTAT Optional return parameter for program status (see below).

The status of the program named in a call of this form is returned in the A-register. (The value returned in the B-register is zero.) If the parameter ISTAT is included in the call, the status is returned to both the A-register and ISTAT. The status word returned has the following form:

```
 15 14                                   3        0
+-----------------------------------------------+
|SG|TL|              0                  | Status  |
+-----------------------------------------------+
```

SG	=0	if program named is not a segment.
	=1	if program named is a segment.
TL	=0	if program named is not in the time list.
	=1	if program named is dormant but in the time list.

Status = the actual program status code of named program.

The status word will contain -1 if the program named does not exist.

RTE-L, XL and A expanded status codes are mapped to RTE-IVB and 6/VM equivalents as shown below:

STATUS CODES

ORIGINAL RTE-L/XL/A STATUS	MAPPED RTE-IVB AND 6/VM RETURNED EQUIVALENT
0	0
1	ERROR
2	2
3	3
4	ERROR
5	ERROR
6	6
7	6
47	0
50	3
51	3
52	3
53	3
54	3
55	3
56	2
57	1
60	1
61	4

Chapter 6
DS/1000-IV PTOP Calls

DS/1000-IV Program-to-Program Communication

The DS/1000-IV Program-to-Program (PTOP) calls provide a data exchange interface between two or more programs located at different nodes. The fundamental concept of PTOP is the master/slave relationship between programs. All PTOP are either "masters" or "slaves," or a combination of both. In all PTOP communication, the master program establishes communication with the slave and initiates each message between itself and the slave program. The master program both sends and receives a message in one call.

The PTOP calls are divided into master and slave call categories as follows:

Master Calls	Description
POPEN	Initiate PTOP communication with slave program.
PREAD	Read data from slave program.
PWRIT	Write data to slave program.
PCONT	Exchange control function data with slave program.
PCLOS	Terminate slave program and logical communication (HP 1000 slave); terminate slave program only (HP 3000 slave).
PNRPY	PTOP with no wait.

Slave Calls	Description
GET	Obtain next master program request.
ACEPT	Accept master program request.
REJCT	Reject master program request.
FINIS	End communication with all master programs.

NOTE: PTOP calls are not supported in RTE-6/VM type 6 (Extended Background) programs. On RTE-A systems with multiuser environment, you must load programs controlled by PTOP master calls as system utilities.

Sample Master-Slave Protocols

The sequence of messages exchanged between master and slave programs is called the protocol of the application. Master-slave protocols can be simple or complex depending on the complexity of the application, how often the programs are expected to be used, how many concurrent copies of the program set will be in use at any given time, and how reliable the application must be.

A master-slave protocol becomes more complicated as the number of programs is increased, especially if the programs are chained. (Programs are chained when a slave is also a master to another slave. This type of protocol is discussed later in this chapter.)

HP 1000 to HP 3000

If your slave program will be at an HP 3000 node, you must call the HELLO utlity to establish a session at the HP 3000 before issuing any PTOP calls. (A "son" program can call the PRCNM utility to communicate with an HP 3000 session established by a "father" program.) When a program has established a session with HELLO, it must terminate the session with a call to the BYE utility. (HELLO, PRCNM and BYE are described in Chapter 7, "DS/1000-IV Utility and Program Calls.")

The following diagram illustrates the PTOP calling sequence between a master program at an HP 1000 node and a slave program at an HP 3000 node. If your application will have multiple masters, an HP 3000 program will be automatically cloned for each master and your slave will never serve more than one master.

```
HP 1000 MASTER                  HP 3000 SLAVE

Establish an SMP
Call HELLO

Schedule slave
Call POPEN ---------------->    Call GET
                                Acknowledge master
               <--------------- Call ACCEPT or REJECT
                                   .
                                   .
             (information exchanged)
                                   .
                                   .
Terminate slave
Call PCLOS ---------------->    Slave terminated

End Session --------------->    Session and logical
Call BYE                        communication terminated
```

Figure 6-1. HP 1000 to HP 3000 Dialogue.

Note that the PCLOS call to an HP 3000 slave program only terminates the slave; it does not terminate logical communication. The BYE call terminates both communication and the SMP established by HELLO.

HP 1000 to HP 1000

Several protocols of increasing complexity are outlined in this section: One-to-One, Many-to-One, and Chain. While the One-to-One protocols may be followed verbatim, the other protocols are intended to be guidelines only and should be tailored to fit your particular application.

When choosing a master-slave protocol from these examples, you should balance the listed limitations of each against the complexity of your application and the number of messages involved. For example, if throughput is an important consideration and you are using full-speed HDLC, use the approximation that all messages take the same amount of time; the time to perform your application will then be proportional to the number of messages needed to do the protocol plus the application.

All of the master-slave protocols described here require that special care be taken when terminating programs. The more complicated protocols, for example, require that the slave never be PCLOSed; these programs will be active until DS/1000-IV is shut down. The Many-to-One and Multiple Server protocols assume that each master is doing one thing at a time, even when the slave is serving more than one master. If your application will have single masters starting up concurrent operations at one time, then these solutions will not be adequate and you should refer to the Chain protocol.

One-to-One

The simplest master-slave protocol involves one master communicating with one slave program. In this protocol, the master issues a POPEN call on the slave, the slave calls GET and ACEPT or REJECT, and the two programs exchange information. The master then calls PCLOS to terminate the slave and logical communication with the slave's node.

```
         MASTER                        SLAVE

     Schedule slave                   Call GET
     Call POPEN ----------------->    Acknowledge master
                    <-------------    Call ACEPT or REJCT
                                .
                                .
                 (information exchanged)
                                .
                                .
     Close slave
     Call PCLOS ------------------>    Slave and logical
                                       communication terminated
```

Figure 6-2. One-to-One Protocol.

LIMITATIONS: only one master can communicate with the slave program at a time; this protocol cannot support multiple masters.

One-to-One with Cloning

This protocol is essentially the same as the previous one except it allows the slave program to handle multiple masters. (It is also the simplest way of dealing with the complexities of multiple masters requesting access to the same slave program.) Cloning of a slave program is accomplished by setting the clone parameter ("ICLON") of the POPEN call to 1. If all masters that POPEN the slave program request cloning, each master will have its own copy of the program. This ensures that the one slave will not be shared by more than one master.

When a slave program is POPEN'd with the cloning option, the master must issue a PCLOS, rather than a FINIS, to terminate the slave when it is through. A slave FINIS call is not adequate because FINIS does not release the ID segment created for the slave when it is cloned. The greatest protection from interference between masters can be achieved by issuing an explicit DLGON call to the slave node, specifying an override parameter of 2. (For more information on DLGON, refer to Chapter 7, "DS/1000-IV Utility and Program Calls.")

LIMITATIONS: this protocol will address most two-program applications, but it works only when the slave is running in a Session Monitor node. In addition, the logon and cloning operations take some time. If these restrictions are unacceptable, refer to the Many-to-One solutions.

Many-to-One

An HP 1000 slave program can be designed to have any number of masters. Handling multiple masters without cloning, however, can be very complicated. Not only must the slave be able to keep track of which master is issuing the current request, it must also know how many are being served.

When one slave program is shared by several masters, care must be taken that none of the master programs terminate the slave with a call to PCLOS while the slave is serving another master. Instead, master programs should signal the slave with a PCONT call when they are finished. Later, when all master requests have been satisfied, the slave program can call FINIS to clean up system resources and, if desired, EXEC 6 to terminate.

One way of solving the multiple master situation is to have the slave serve one master at a time and then terminate when all the masters have been served. This is accomplished by having the slave check the function code of each master request that it receives. (The function code is contained in the "IFUNC" parameter of the slave GET call.) If the slave receives a POPEN while serving another master, it rejects it. After the slave completes serving the current master, it returns to its GET, ACEPT's the next POPEN it receives, and begins serving the next master.

On the master side of this protocol, the master program waits for some period of time after being rejected before reissuing its POPEN call. How long it waits should be determined by your application; the master should wait until the other master currently using the slave is likely to be through. This wait should be random if more than one master at a time might attempt to POPEN a slave that is already in use.

If this protocol is to run in a Session Monitor environment at the slave node, the master should issue a DLGNS (to request non-session access at the slave node) before issuing its POPEN call. The POPEN must not request cloning. (For more information on on DLGNS, refer to

Chapter 7, "DS/1000-IV Utility and Program Calls.)

LIMITATIONS: does not give access to the Session Monitor environment for file access and capability checking; forces subsequent masters to wait.

Many-to-One, Multiple Server

The next level of complexity is a slave capable of serving many masters concurrently. This protocol can be written in much the same way as the Many-to-One protocol above with the exception that subsequent POPEN calls are accepted and acted upon by the slave. This feature complicates the protocol because the slave must keep track of which master is making each request.

The slave can easily identify a master program if all the information it needs can be contained in the data or the tag field of the POPEN call. This solution is not adequate, however, if your slave program requires additional information or must keep state information (such as DCBs). If this is the case, the slave can keep tables for the state information and index them with a "master number" that it can generate as each POPEN is received. The slave can also send this number to each master in the tag field to identify subsequent calls.

LIMITATIONS: as with Many-to-One protocol, does not give access to the Session Monitor environment for file access and capability checking; requires complex bookkeeping.

Many-to-One, Possibly Cloned

If your application will communicate with both Session Monitor and non-Session Monitor nodes, you could consider writing two different slaves with the same name. In this protocol, one slave is written to run in the Session Monitor node using the One-to-One, cloned slave protocol; the other is written to run in the non-Session Monitor node using the Many-to-One, Multiple Server protocol.

On the master side of this protocol, the master program calls DLGON, saves the error message returned by the call, and checks to see if it is a -4. An error of -4 indicates that the slave node does not have Session Monitor. If the call succeeds, the slave node does have Session Monitor. The master then issues a POPEN with cloning (the cloning parameter is ignored by non-Session Monitor nodes) and proceeds. How the master terminates is dependent upon whether or not the slave node has Session Monitor. If a -4 error was returned by the DLGON call, the master program terminates. If a -4 error was not received at the DLGON call, the master program issues a PCLOS to release the ID segment created for the slave program when it was cloned.

If writing two different slave programs is not appropriate or feasible, but your application involves slave programs at both Session Monitor and non-Session Monitor nodes, your master program can employ a shut-down protocol. This protocol is a merging of two distinct protocols. Which protocol is actually in effect depends on whether the slave program is running in a Session Monitor or non-Session Monitor node. If the slave program is located at a Session Monitor node, it is cloned and the protocol in effect is fundamentally the same as One-to-One, cloned slave protocol. If the slave node does not have Session Monitor, the protocol in effect is similar to the Many-to-One, Multiple Server protocol.

On the master side of the shut-down protocol, the master program calls DLGON (but without checking the error return) and then calls POPEN with the cloning option. The master must be prepared to have its POPEN rejected by the slave. If the POPEN is rejected, the master waits a short time (one second or less) and then reissues the POPEN. This is repeated until the POPEN is ACEPTed. When finished (or if an error is detected and a premature shut-down is desired), the master requests permission from the slave program to issue a PCLOS and, if permission is granted, it PCLOSes the slave.

If the slave program is running in a Session Monitor node, the master's DLGON call is accepted and, if the subsequent POPEN call is accepted, the slave program is cloned. When the master program requests permission to PCLOS the slave, perhaps with a PCONT, the slave is PCLOSed. (The cloned slave will never reject the PCLOS request.)

If the slave program is running in a non-Session Monitor node, the master's DLGON call is rejected and the slave program is not cloned. Because the slave is not cloned, it is POPENed several times by different master programs and must be able to identify which master is making which request. When the master program requests permission to PCLOS the slave, the slave program refuses all but the last PCLOS and then terminates.

The slave program in this protocol is written as outlined above in the Many-to-One, Multiple Server protocol with tables to keep track of multiple, concurrent copies of your application. In addition, it must know how many masters are communicating with it at any one time (this information may be implicit in the way you keep your tables.) When a slave program receives a PCLOS permission request from a master program, it should grant permission only if it is serving that particular master and that master only. After granting permission, the slave should REJCT all incoming messages.

LIMITATIONS: this protocol is very complex because it must operate in both a session and non-session environment. A loss of bandwidth for data transfer occurs as a result of the additional messages required soley for the purpose of handshaking. In addition, the logon and cloning operations take some time.

Chain Protocol, Non-Session Monitor Nodes Only

A "chain" is a set of three or more programs that are written so that some serve as both masters and slaves. A "chain protocol" has an added level of complexity over the protocols previously described. This is primarily because programs closer to the original master have less direct control over programs that are further down the chain. Although these programs can control activity along the chain to some degree, the effectiveness of their control and their ability to gather information necessary for that control, are hindered by the existence of intermediary programs.

If your application requires a chain protocol, you must first select from among the master-slave protocols discussed earlier and then add the chaining. If the relationship between two of the programs in your chain requires a simple protocol such as One-to-One, but the protocol required between one of these programs and another needs to be more complex, then your entire program set must be written using the more complex protocol.

The following diagram and text describe a common chain protocol that consists of a set of three programs which transfer data between three different nodes. Using this program set, a user at NODE A, NODE B or NODE C can transfer data between the other two nodes. Moving data from the local node to a remote node, or pulling data from a remote node to the local node, are subsets of this application.

Although this example shows three programs, protocols involving four or more chained programs can also be created by replicating the protocol parts of the middle program in the three-program chain. In the following diagram, the middle program is called the "Producer."

```
        NODE A              NODE B                 NODE C

     ("Initiator")        ("Producer")           ("Consumer")
        Master <----------> Slave/Master <----------> Slave
```

Figure 6-3. Three-Program/Node Chain Example.

Each program in this set has its own unique function: NODE A's "Initiator" initiates the data transfer and is the program run by the user to start the tranfser; NODE B's "Producer" produces the data and sends it; and NODE C's "Consumer" receives the data and acts on it (perhaps storing it in a file). Because the Initiator is pure master and the Consumer is pure slave, there are few difficulties in writing these programs. The Producer, however, is both master (to the Consumer) and slave (to the Initiator), and is subsequently more complex.

Shut down in this protocol, and in chain protocols in general, is also more complex than in the previously discussed protocols. While the Initiator cannot communicate directly with the Consumer, it can "orphan" the Consumer by PCLOSing the Producer prematurely. Because of this, the Consumer must be "cleaned up" before the Producer terminates and shut down must be the reverse of start up: first the Consumer is terminated, then the Producer, and finally the Initiator.

The shut-down process is started by the Initiator program which signals the Producer to shut down the Consumer. The Producer and the Consumer then engage in a shut-down dialogue which, depending upon the requirements of your particular application, may consist of a simple PCLOS on the part of the Initiator, or may involve several messages. When the Consumer has been shut down, the Producer can ACEPT a shutdown order from the Initiator, and the Initiator and Consumer can conduct their own shut-down dialogue. Again, this dialogue may be simple or complex depending on your application.

Another complication that must be dealt with is presented by the program that acts as the initiator of the application. Although this program must be largely uninvolved during the normal operation of the protocol, it must remain active throughout. A common mistake made when designing an Initiator is to have it order each exchange between the Producer and Consumer programs. A protocol such as this one might be conceived in the following manner:

```
        NODE A              NODE B                NODE C

      ("Initiator")        ("Producer")              ("Consumer")

     Tell the Producer
     to send the first
     block
     Call PCONT ----------> Call GET
                            Send the first
                            block to the
                            Consumer
                            Call PWRIT ------------> Call GET
                                                     Act on the
                                                     first block;
                                                     tell the Producer
                            Tell the    <----------- Call ACEPT
                            Producer the
                            first block is
                            done
     Tell the <------------ Call ACEPT
     Producer to
     send the
     second block
     Call PCONT ----------->
                                      .
                                      .
                                      .
```

Figure 6-4. Chain Protocol Application Example.

This interpretation of a chain protocol is inefficient because the time needed to send one block is equal to the sum of the time taken to send two messages. (All messages over the same speed link will take about the same time.) This solution is not adequate if the performance of your program set is a high priority. If you require better performance, you may consider having the Producer and Consumer exchange multiple blocks before the Producer acknowledges the Initiator's order. In this case, the Initiator must still remain involved in each transaction, at least to the extent that it knows when activity has ceased between the two other programs.

An additional complication encountered with this version of the protocol is that the replies cannot be sent unless the Initiator's messages to the Producer are acknowledged within the TCB time-out value set at the Producer's node, and the master TCB time-out set at the Initiator's node. (TCB time-out values are set by the Network Manager during network initialization.) You will have to determine, perhaps through experimentation, how many messages the Producer and Consumer programs can exchange safely (without a time-out occurring) before the Producer must acknowledge the Initiator's message. Once the correct timing is established, your Initiator program can be coded to call PCONT to order the Producer to send multiple blocks to the Consumer. After the blocks are sent, the Producer can then acknowledge the Initiator's order and wait at its GET for the next order.

An alternative to this solution to the timing problem is to have the Producer use the system time as a basis for determining when the slave TCB time-out has almost expired. This could be done by coding the Producer to check the system time when its GET receives a request from the Initiator, add the slave TCB time-out value to the time and, after it sends each block to the Consumer, compare the sum of these two values to the current system time. In order to increase the probability that this protocol will work within the TCB time-out constraints, you should give the Producer more than enough time ACEPT the Initiator's requests. This extra time will help avoid time-outs during transient peaks in traffic or resource bottlenecks.

LIMITATIONS: This protocol is one of the most complex protocols described in this chapter. Although you can estimate the time taken to process requests, these estimates cannot be completely accurate because they can be influenced by traffic in the network, noise on the transmission lines, system loading on any of the systems involved, and many other factors that can slow message processing. Because the estimates used must include a margin for error, throughput in the protocol is further reduced. Very long chains are impossible because the time needed to perform all the tasks at the lower end of the chain, and have them reported back to the top, may grow to exceed the Initiator's master TCB time-out, even for just a single transaction.

Chain Protocol, Non-Session Monitor and Session Monitor Nodes

If a chain protocol involves both non-session and session nodes, it shares the same characteristics as the previously described chain protocol, but with the following additional ramifications:

* If the Initiator, Producer and Consumer are all running in a session environment, the Initiator has responsibility for the resources used by both the Producer and the Consumer. If the Initiator's session ends, not only will the Initiator terminate, but the Producer and Consumer will be terminated as well.

* If session and non-session nodes are mixed in the chain, a process in a non-session node can become "orphaned" due to the premature termination of its Producer or Initiator. Once this occurs, the "orphan" will wait indefinitely at its GET along with all of the other processes below its place in the chain.

LIMITATIONS & ERROR RECOVERY CONSIDERATIONS: This is the most complex protocol: sessions are created and destroyed on behalf of master programs; system resources (in addition to DS resources) are allocated to programs so that applications can accomplish their tasks; and all communication between masters and slaves must occur within the user-defined TCB time-outs. The protocols between each master and slave pair in the Initiator, Producer and Consumer scheme are so complex because these programs must be general enough to work in varied environments.

Error recovery after breaking this protocol is also the most complex. The ways the protocol can be broken include the following: a master-slave program may abort; the session under which a master or slave is running at a Session Monitor node may terminate abnormally; a TCB time-out may occur. In addition, a bug in the protocol itself, or the occurrence of a condition that was not planned for, may cause the master and slave programs to become confused.

Once a process becomes aware of an error condition it must determine what sort of error recovery is appropriate. State information (usually maintained by the Producer) may be impossible to recover and may make error recovery procedures difficult. Establishing enough resynchronization to shutdown and cleanup may also be difficult.

PTOP Calling Sequences

The PTOP calling sequence syntax described in the following paragraphs is shown in FORTRAN format. All parameters, required and optional, are shown for each PTOP call. An optional parameter is denoted in the call syntax by enclosure within brackets. The brackets are not to be specified in an actual coded call line. Parameters should be specified as integers in the sequence shown and must be included when required in the calling sequence. An optional parameter does not need to be specified unless it precedes an optional parameter that you need to specify in your particular case. In this case, you should define each intervening optional parameter as a dummy variable to maintain the proper sequential parameter order.

PTOP Call Common Parameters

Several parameters are common to more than one PTOP call. These common parameters are described in this paragraph in detail. Parameters unique to a specific call are described within the paragraph defining that call.

Common Parameter	Description
IPCB	PTOP control block; a 4-word array that serves as a control block for the data link. The PCB array resides within the calling program and must not be modified.
IERR	Error return; A variable to which an error code is returned if an error condition is encountered during execution of a PTOP call, or '1' is returned if the slave program responded with a REJCT. If no errors were encountered during the PTOP call execution, the value will be '0' and the slave program responded with an ACEPT.
ITAG	Tag field; a 20-word array in which information you define can be exchanged between the master and slave programs. ITAG data is passed from a master program via a POPEN, PREAD, PWRIT or PCONT call. A slave program may then pass ITAG data back to the master program with an ACEPT or REJCT call. Within an HP 3000 system the tag field is also a 20-word array.

Master PTOP Calls

POPEN Call

A POPEN call directed to a remote node causes the scheduling of the named slave PTOP program at that node. If the remote node is under Session Monitor control, the user can either precede the POPEN call with an explicit DLGON call, take the default user name, or run the program as a 'son' of REMAT. In any of these cases, a non-interactive session will be created at the slave node and the slave program will run under that session. If that slave program is to access files in an account at its node, the files must be on cartridges available to the session (account). Since a master can establish only one session per node, communication with multiple slaves is only possible when they all run under the same session. If a remote session has been created for a slave program to run under, that session does not get released when the master issues a PCLOS or the slave issues a call to FINIS. The master program must either terminate or make a call to DLGOF to release the session.

NOTE: If your master program is segmented, you must call POPEN from the main program. If your master is an RTE-A CDS program, you may call POPEN from any segment, but the PTOP control block ("IPCB" parameter) must be declared in the main program if it is written in PASCAL, or declared as a SAVE variable if it is in FORTRAN.

HP 1000-HP 1000 PTOP Calling Sequence

For PTOP communications between HP 1000s (HP 1000-to-HP 3000 PTOP is discussed later in this section), you call POPEN within a master program to initiate the communication link with a slave program.

CALLING SEQUENCE

```
      CALL POPEN(IPCB,IERR,NAME,NODE,ITAG[,ICLON])
```

IPCB	PTOP control block (see PTOP Common Call Parameters).
IERR	Error return (see PTOP Common Call Parameters).

NAME	Slave program name: an array of up to 14 words containing the ASCII-coded slave program name; terminated by a blank or non-ASCII character.
NODE	The number of the node where the slave program resides and where it is to be scheduled for execution (see below for additional information).
ITAG	Tag field; 20 word array (see PTOP Common Call Parameters).
ICLON	Slave cloning parameter. When equal to 1, slave program is duplicated and renamed. When zero and an ID segment exists, the original slave is scheduled. Default is 0.

The node value declared by the NODE parameter is placed into the PTOP control block (IPCB) and is used for any PREAD or PWRIT calls that reference this control block.

A value of -1 in the NODE parameter declares execution is to occur in the local node.

When ICLON is set to 1 and the remote node is under session, a slave program is duplicated and renamed under the remote session created by the master user. A master program can then have its own exclusive copy of the slave program, and eliminate the problem of coding a slave program that must process or reject several masters. Slave programs are not cloned in the local node.

When a slave program is renamed, the last two characters will be replaced with ".A",".B",".C", etc. until a unique name is found.

If the slave program cannot be found (SP'ed but clone bit not set or program does not exist), an IERR of -41 is returned. If the file is SP'ed (a type 6 file exists) and the clone bit is set, the program is RP'ed (given an ID segment) and scheduled. An error in the "RP" operation will also return an IERR of -41. When the clone bit is not set, the slave must have an existing ID segment in order to be scheduled.

If the remote node has Session Monitor, the remote session under which the slave program is running will be "owned" by the master user that issued the first POPEN request that caused the slave program to be scheduled. All subsequent masters will have their own explicit or default sessions separate from the one in which the slave program is running. (The name of the session may be the same, but the actual session is separate.) In this case, the original copy of the slave program will run under the session specified by the first master POPEN request and will have the capability level of that account. All further master POPEN requests (whether they have their own explicit session or not) will be directed towards this original copy of the slave located in the session specified by the initial POPEN request. If secondary masters log-off, the remote session under which the global slave is running will not be affected.

If a slave is going to be accessing resources unique to a slave session, you must be sure to either clone (make a copy of) the slave or make sure the other masters "know" the slave is running under the proper account. If "secondary" masters do not need a session by a different name, they need not bother with creating one - just let the system create the default session.

NOTE

The master program that issued the initial POPEN request must not log-off the remote session or terminate until all the other masters have completed accessing the slave or the slave will be aborted.

PTOP slave programs can turn around and issue master RFA, DEXEC, etc. requests back to the master program's node and session. The slave program simply issues these requests to the originating node without logging on (as described in the previous section under paragraph "Re-directing Master Requests to Originator"). In a slave handling multiple masters, these requests will be directed only to the master who issued the original POPEN request.

Refer to the master-slave protocols described earlier in this chapter for more information on POPEN.

HP 1000-HP 3000 PTOP Calling Sequence

CALLING SEQUENCE

```
    CALL POPEN(IPCB,IERR,NAME,NODE,ITAG[,IENAM[,IPRAM[,IFLAG
              [,IBFSZ]]]])
```

IPCB	PTOP control block (see PTOP Common Call Parameters).
IERR	Error return (see PTOP Common Call Parameters).
NAME	Slave program name; an array of up to 28 bytes containing the ASCII-coded slave program name. The array is terminated by a blank or non-ASCII character.
NODE	The number of the node where the slave program resides (see below for additional information).
ITAG	Tag field 20 word array (see PTOP Common Call Parameters).
IENAM	DS/3000 program ASCII entry point name (optional); refer to the HP Distributed Systems Network DS/3000 Reference Manual.
IPRAM	DS/3000 program control information (optional); refer to the HP Distributed Systems Network DS/3000 Reference Manual.
IFLAG	DS/3000 program loading options (optional); see below for additional information. Also, refer to the HP Distributed Systems Network DS/3000 Reference Manual.
IBFSZ	DS/3000 program communications buffer size (optional); refer to the HP Distributed Systems Network DS/3000 Reference Manual.

For data communications with an HP 3000 node, the NODE parameter must contain the negative value of the logical unit number associated with the 1000/3000 link. Remember that you must open the line using the program DSLIN (from the HP 3000 side you must also use the MPE DSLINE command to open the line) and call the subroutine HELLO (or enter the HELLO command from RMOTE) before issuing PTOP calls to an HP 3000. For information on DSLIN, see chapter 3 of this manual.

If you use PTOP calls over an X.25 link to an HP 3000, you must know the lu of the virtual circuit assigned to you by the X.25 network. The subroutine LU3K provides this information. Refer to chapter 7 for more information on LU3K.

The IFLAG parameter passes loading option information to the scheduled slave program at the HP 3000 node. The loading options are declared by the value of the bits within the IFLAG parameter word. These bit settings are defined below.

NOTE

The HP 3000 convention for specifying computer word bits is the reverse of the HP 1000 convention. That is, the bit configuration in an HP 3000 computer word is represented:

```
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

while the bit configuration in an HP 1000 computer word is represented:

```
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
```

HP 1000 Bit	HP 3000 Bit	Description
0	15	Not used.
1	14	LOADMAP bit. If one, a load map is produced. If zero, no map is produced. The default setting is zero.
2	13	DEBUG bit. If one, a breakpoint is set at the first executable instruction of the new process. If zero, no breakpoint is set. The default setting is zero. The value of this bit is ignored if the user is non-privileged and the new process requires privileged mode, or if the user does not have read/write access to the program file of the new process. The default setting is zero.
3	12	NOPRIV bit. If one, the program is loaded in the non-privileged mode. If zero, the program is loaded in the mode specified when the program file was prepared. The default setting is zero.
4 and 5	11 and 10	LIBSEARCH bits. These bits denote the order in which libraries are searched for the program: 00=System Library; 01=Account Public Library, then System Library; 10=Group Library, then Account Public Library, followed by System Library. The default setting is 00.
6	9	NOCB bit. If one, file system control blocks are established in an extra segment. If zero, control blocks may be established in the PCBX area. The default setting is zero.
7 and 8	8 and 7	Reserved for MPE; should be set to 00.
9 and 10	6 and 5	STACKDUMP bits. These bits control the arming/disarming (enable/disable) of the mechanism by which the stack is dumped in case of an abort operation: 00=arm stack dump only if armed at father level; 01=arm stack dump unconditionally; 10=same as 00; 11=disarm stack dump unconditionally for new process. The default setting is 00.
11	4	Reserved for MPE; should be set to zero.

NOTE

The following bits are used only if the STACKDUMP bits are 01; otherwise, the following bits are ignored.

12	3	DL to QI bit. If one, the portion of the stack from DL to QI is dumped. If zero, this portion is not dumped. The default setting is zero.
13	2	QI to S bit. If one, the portion of the stack from QI to S is dumped. If zero, this portion is not dumped. The default setting is zero.
14	1	Q-63 to S bit. If one, the portion of the stack from Q-63 to S is dumped. If zero, this portion is not dumped. The default setting is zero.
15	0	ASCII dump bit. If one, the stack dump is interpreted in ASCII code, as well as in octal notation. If zero, no ASCII interpretation is performed. The default setting is zero.

If the IFLAG parameter is omitted, the default values are implied.

PREAD Call

A master program PREAD call is issued to read data from a slave program. This routine obtains a data buffer from a slave program executing at the node specified in the master POPEN call which last referenced the indicated PTOP Control Block (IPCB). The data and tag field are read into the master program buffers.

CALLING SEQUENCE

```
    CALL PREAD (IPCB,IERR,IBUF,IL,ITAG)
```

IPCB	PTOP control block (see PTOP Common Call Parameters).
IERR	Error return (see PTOP Common Call Parameters).
IBUF	Data buffer; an array of a size equal to or greater than the value of the IL parameter.
IL	Data length in words; a positive decimal value that declares the length of data available for the PREAD request. This value is passed to the IL parameter in the slave's GET call as an indication of the maximum amount of data to send to the master. A maximum of 4096 words may be transferred between an HP 1000 and another HP 1000, or between an HP 1000 master program and an HP 3000 slave program. The maximum number of words transferred between an HP 3000 master program and an HP 1000 slave program depend on which size buffer module was appended to the request and reply converters (RQCNV and RPCNV). Maximum is 4096 words. The 3000/1000 buffer size defaults to the buffer size specified when the 3000/1000 link was configured. This default is overridden by supplying the optional buffer size parameter in the POPEN call at the 3000. Between the 1000 and 3000, a negative length may be used to indicate the length is in bytes. When an odd number of bytes are to be transferred, an extra byte is appended to the end of the data.
ITAG	Tag field (see PTOP Common Call Parameters).

PWRIT Call

This call is used to transfer data from a master program to a slave program. A data record and the tag field are written from buffers in the master program to a slave program executing at the node specified in the POPEN call which last referenced the indicated PTOP Control Block (IPCB).

CALLING SEQUENCE

```
CALL PWRIT(IPCB,IERR,IBUF,IL,ITAG)
```

IPCB	PTOP control block (see PTOP Common Call Parameters).
IERR	Error return (see PTOP Common Call Parameters).
IBUF	Data buffer; an array of a size equal to or greater than the value of the IL parameter.
IL	Data length in words; a positive decimal value that declares the actual length of data being transferred to the slave for the PWRIT request. This value is passed to the IL parameter in the slave's GET call as an indication of how much data to expect from the master. A maximum of 4096 words may be transferred between an HP 1000 and another HP 1000, or between an HP 1000 master program and an HP 3000 slave program. The maximum number of words transferred between an HP 3000 master program and an HP 1000 slave program depend on which size buffer module was appended to the request and reply converters (RQCNV and RPCNV). Maximum is 4096 words. Between the 1000 and 3000, a negative length may be used to indicate the length is in bytes. When an odd number of bytes are to be transfered, an extra byte is appended to the end of the data.
ITAG	Tag field (see PTOP Common Call Parameters).

PCONT Call

A call to PCONT provides for the exchange of a tag field between the master and slave programs. The call to PCONT transmits a tag field to the slave program. The slave program must issue a GET call to obtain the tag field data. Then, the slave program must issue a call to either ACEPT or REJCT and return a tag field to the waiting master program.

CALLING SEQUENCE

```
      CALL PCONT(IPCB,IERR,ITAG)
```

IPCB	PTOP control block (see PTOP Common Call Parameters).
IERR	Error return (see PTOP Common Call Parameters).
ITAG	Tag field (see PTOP Common Call Parameters).

PCLOS Call

A PCLOS call is issued from a master program to terminate the communication link established by the call to POPEN and to terminate the execution of the slave program associated with the communication link. The communication link between the slave program and all programs associated with it is terminated immediately. Note that the communication link mentioned here refers to the logical data transfer path between programs and does not refer to a physical communication device.

If the slave has multiple masters, a call to PCLOS terminates all other pending master requests, if they exist, and therefore may cause some serious problems. In these cases, it is safer to make a call to PCONT with a flag set in one of the tag field words indicating the master is finished communicating with the slave. The slave may then call FINIS when all masters have indicated they are done and then terminate.

CALLING SEQUENCE

```
      CALL PCLOS(IPCB,IERR)
```

IPCB	PTOP control block (see PTOP Common Call Parameters).
IERR	Error return (see PTOP Common Call Parameters).

PNRPY Call

The PNRPY call is issued from the master program to eliminate the need for the master to wait for a reply from the slave (an ACEPT or a REJCT) before continuing processing or issuing another call. This call is honored only for PWRIT, PCONT, and PCLOS. POPEN and PREAD calls will be forced to wait for the reply. PTOP communication with no reply is available only between HP 1000s and no Message Accounting is provided on PTOP calls with no reply.

CALLING SEQUENCE

```
     CALL PNRPY([IMODE][,ITTO])
```

IMODE Defines the scope of the call. The options are:

0	No-reply only applies to the next PTOP call (default).
<0	No-reply applies to all PTOP calls that follow, until explicitly turned off.
>0	Turns off the no-reply option.

ITTO Transaction Time-Out override value. If specified, it must be positive and less than 256. If not specified, the transaction time-out value which is used for all other requests for that node, is used. Negative values or values greater than 256 are truncated to the value of the lower eight bits of the stated value. ITTO is specified in five second intervals (i.e., ITTO=2=10 secs).

The basic purpose of this call is to improve network performance by cutting down on unnecessary traffic. Using this call, however, sacrifices the guarantee that messages will be delivered and whether they will be delivered in the proper sequence. The no-reply option will only attempt to send the message through the DS/1000-IV network, but confirmation as to whether the message was delivered successfully will not be generated.

The Transaction Time Out value is a sensitive parameter and requires some backround information on networks in general.

Flow control is a very important consideration in any network design. A network has the capacity to handle only so many messages at any one time. In DS/1000-IV, flow control is handled by Transaction Control Blocks (TCBs). No message can be generated without first allocating a TCB to monitor whether a reply is received within a specified number of seconds, and what master should receive the reply. Since there is a finite number of TCBs (established when the nodes are first initialized by the Network Manager) there is a fixed limit on the number of messages allowed in the network at any one time.

Even with the no-reply option, a TCB is allocated for each request. In normal cases, the TCB are deallocated once a reply is returned or when a time-out occurs. With the no-reply option however, TCBs are deallocated only when a time-out occurs. This consequently means that you must not generate transactions at a rate faster than the TCBs can time-out. Doing so would tie up all the available TCBs and any further requests would return a -58 or a DS08(0) error (no TCBs available).

The parameter ITTO allows you to specify a Transaction Time-Out value, to be used optionally in place of the default value specified when the node was initialized. You would certainly wish to use a very small time-out value here in order to obtain a very high transaction rate (i.e., the smaller the time-out value, the less time it takes for a TCB to be freed for the next request). However, too small of a value allows more messages to be injected into the network than it can handle causing other nodes to suffer since they must compete for resources with this flood of extra messages.

On the other hand, a value that is set too high causes programs to consume TCBs faster than they become available, consequently generating a -58 error to all PTOP calls, until a TCB becomes available. A delicate balance must be made here to insure the Transaction Time-Out value is not set too low so that messages flood the network, but, at the same time, not set too high so that programs eventually consume all available TCBs.

The basic rule to follow in determining this balance is:

If we let N be the number of TCBs in the system, and T be the Transaction Time-Out value, the rate of transactions may not exceed N messages per T seconds:

$$\text{Rate of Transactions} < N/T$$

If an attempt is made to increase the rate of transaction, this will, of course, cause an error of -58 since no TCBs are available. In this case, it would actually be faster to use the standard PTOP calls with reply since the TCB are released as soon as the reply arrives.

NOTE

Misuse of the no-reply option can undermine the flow control of DS/1000-IV. This will occur when too many messages flood the system because the transaction time-out values are set far smaller than the amount of time taken to transmit a given message. Also no Message Accounting is performed on messages sent with no reply. Use this option carefully, and only if the success of your application depends on the function of this call. Also, a no-reply PTOP call issued to a 91740 DS/1000 node will be rejected.

Slave PTOP Calls

GET Call

The GET call is issued from a slave program to obtain the next outstanding master program request call destined for this slave program. If no master program requests are outstanding, the slave program execution is suspended. The call to GET must be followed either by a call to ACEPT or a call to REJCT (see ACEPT and REJCT call descriptions).

CALLING SEQUENCE

```
CALL GET(ICLAS,IERR,IFUNC,ITAG,IL[,IBUFR,IBUFZ])
```

ICLAS	Slave PTOP class number: a value for ICLAS is assigned to the slave program as the first parameter when the slave program is scheduled. The first thing every slave program should do is issue a call to RMPAR to obtain the value of ICLAS. Only one value for ICLAS is created by the initial POPEN call. All further POPEN calls to the same slave are assigned the same ICLAS value. The value for ICLAS must be saved and referenced in each call to GET. Be careful that you do not alter the content of ICLAS.
IERR	Error return (see PTOP Common Call Parameters).
IFUNC	Functional return; a function value is returned to this word upon completion of a call to GET. This value indicates the type of master program request obtained via GET. Request types are: 1 = POPEN 2 = PREAD 3 = PWRIT 4 = PCONT
ITAG	Tag field received from master program (see PTOP Common Call Parameters).

IL A value is returned here at the completion of the GET call when IFUNC=2 or 3 (PREAD or PWRIT). IL indicates the actual number of words to be transferred if the master request is accepted. See the IBUFR parameter.

IBUFR Data buffer (optional); an array the size equal to or greater than the value of the IBUFZ parameter. This parameter is only required when the master request is a PWRIT. Data from the master is transferred to this buffer (see Table 6-1).

For POPEN, PREAD, and PCONT requests, the buffer contents are unmodified. On PWRIT requests the data is copied to IBUFR, up to the limit specified in IBUFZ. IL contains the actual size of the data buffer, as sent by the master. Among its other error checking, the slave should check to see whether IL is larger than IBUFZ, i.e., if the master sent too much data.

IBUFZ Defines buffer size of IBUFR (optional).

The request is released after the GET call. The reply is not sent until ACEPT or REJCT is called.

Table 6-1. GET Calls With PWRIT and PREAD.

Master Call	IL (GET call)	IBUFR (GET call)	IBUF (ACEPT call)	IBUFZ (GET call)
PWRIT	Actual number of words transferred here.	Data from slave is transferred to this buffer.	Data from master may be optionally transferred to this buffer.	Size of IBUFR in GET call. If IL > BUFZ, an error will occur.
		One or the other buffer may be used. If IBUFR (GET call) is used, IBUF (ACEPT call) should not be used.		
PREAD	Size of master buffer specified in master.	-----------	IL words from this buffer are transmitted to master.	-----------

ACEPT Call

A slave program making a call to this routine accepts (and completes) the master program request obtained via the GET call. A tag field and optionally a data buffer are then returned to the master program. The master program is suspended until this reply is received. If the reply is delayed to long, a master program time-out condition results. If the reply transmission fails, error code -47 is returned to the ACEPT call IERR parameter. In this case, the slave program should return to its GET call to allow any retry attempt to be initiated from the master program.

CALLING SEQUENCE

```
      CALL ACEPT(ITAG,IERR[,IBUF])
```

ITAG	Tag field to be returned to master program (see PTOP Common Call Parameters). In a PREAD request, the actual length of the buffer being transferred should be passed in this parameter.
IERR	Error return (see PTOP Common Call Parameters).
IBUF	Optional data buffer; this buffer contains the data to be transferred in a PREAD request and optionally in a PWRIT request.

REJCT Call

A slave program call to this routine rejects the master program request obtained via the GET call. A tag field is returned to the master program. Similar to the results of an ACEPT call, if the reply to the master program is delayed, it is possible that an error code -47 will be returned to the REJCT call IERR parameter. In this case, the slave program should return to its GET call to allow any retry attempt to be initiated from the master program.

CALLING SEQUENCE

```
      CALL REJCT(ITAG,IERR)
```

ITAG   Tag field (see PTOP Common Call Parameters).

IERR   Error return (see PTOP Common Call Parameters).

A call to REJCT may be used when a slave receives a second master request and the slave is not designed to handle multiple masters. It may also indicate an error occurred in the slave and the master request could not be completed. For the latter case, it is a good idea to send the master, by way of the tag field, any error values returned and the function value representing the pending master request. If you have an error in the GET call, there may be a problem in the call sequencing or there may be a problem that will prevent any message from getting to the master. If an error occurs elsewhere in the slave processing, this value should also be returned.

FINIS Call

A call to FINIS from the slave program terminates communication with master program requests. This call does not cause termination of the slave program. Any master program requests destined for this slave program are rejected until a communications link is established via another POPEN call from a master program. In a multiple master environment, you may want to set up a counter in the slave program to keep track of how many master requests are still active. Only when the counter is 0 (no more master requests) should a call to FINIS be issued.

CALLING SEQUENCE

```
      CALL FINIS
```

PTOP Sample Programs

An example of a master program and an example of its associated slave program follows.

```
FTN4,L
      PROGRAM FCPY (3,50), PTOP File Copy (Master Program)
C****************************************************************
C This PTOP Master/Slave program pair provides a high speed file*
C transfer capability, utilizing highly efficient buffered data *
C transfers.                                                    *
C                                                               *
C The data (records) of the source file are read into a large   *
C buffer then transferred all at once with one PWRITE request   *
C across the comm link. This technique dramatically increases   *
C the link throughput over the normal RFA techniques of         *
C one-record-at-a-time transfer.                                *
C                                                               *
C****************************************************************
C  RU,FCPY,Source namr,Destination namr,Destination Node Number *
C****************************************************************

      IMPLICIT INTEGER(A-Z)

      DIMENSION DCB(144),BUFFER(1024),IPBUF(10),PCB(4),TAG(20)
      DATA BFSZ/1024/,DMY/0/
      LU=LOGLU(I)

C Get Run String.  B reg will contain length of string on return.
      CALL EXEC (14,1,BUFFER,-80)
      CALL ABREG (I,LEN)

C Start parsing after "RU,FCPY,".
      PTR=9

C Get NMAR1 and NAMR2. NAMR2 goes to Tag buffer for slave program.
      I=NAMR (IPBUF,BUFFER,LEN,PTR)
      I=NAMR (TAG,BUFFER,LEN,PTR)

C If parameter 2 is defaulted, copy IPBUF to TAG.
      IF(TAG .NE. 0)GOTO 16

      DO 15 J=1,10
      TAG(J)=IPBUF(J)
15    CONTINUE
16    CONTINUE
```

```
C Get security code and cartridge reference number.
      SC  =IPBUF(5)
      CR  =IPBUF(6)

C Open the source file.
      CALL OPEN (DCB,ERR,IPBUF,1,SC,CR)
      IF(ERR .GE. 0) GO TO 5
      WRITE(LU,100)ERR
100   FORMAT(/"OPEN ERROR ",I3)
      STOP 1

C Get the size and type of source file.
5     CALL LOCF (DCB,ERR,REC,DMY,DMY,SZSEC,DMY,TYPE)
      SIZE  =SZSEC/2
      TAG(7)=TYPE
      TAG(8)=SIZE

C Get node number.
      I=NAMR (IPBUF,BUFFER,LEN,PTR)
      NODE=IPBUF(1)

C Now schedule the slave, tag field contains destination file namr
      CALL POPEN(PCB,ERR,6HFCPYS ,NODE,TAG)
      IF(ERR .EQ. 0) GO TO 10
      WRITE(LU,110)ERR
110   FORMAT(/"POPEN ERROR "I3)
      STOP 2

C Initialize record count and buffer index
10    BUFFER(1)=0
      IX=3

C Now loop to read and block records from the file
C into the BUFSZ buffer.

20    CALL READF (DCB,ERR,BUFFER(IX),80,LEN)
      IF(ERR .GE. 0) GO TO 25
      WRITE(LU,120)ERR
120   FORMAT(/"READF ERROR",I3)
      STOP 3

C Advance record count kept in BUFFER (1).
C Store length of current record in BUFFER (IX-1).
C Advance IX to next record position.

25    BUFFER(1)=BUFFER(1)+1
      BUFFER(IX-1)=LEN
      IX=IX+LEN+1
```

```
C Now, if we haven't exceeded the buffer length or encountered
C an end-of-file, go back and read another record.
      IF (IX .LT. BFSZ-80 .AND. LEN .NE. -1) GO TO 20

C If it was an EOF, backup the record counter.
      IF (LEN .EQ. -1) BUFFER(1)=BUFFER(1)-1

C Write the buffer to the slave program.
      CALL PWRIT (PCB,ERR,BUFFER,IX,TAG)

C If we don't get an error or an eof, go read some more records.
      IF(ERR .EQ. 0  .AND.  LEN .NE. -1) GO TO 10

C Was it error or eof?

C If PWRIT error, print message and quit.  The user will have to
C purge the destination file and try again or make
C this program more elegant.  You might indicate to the
C slave an error occurred by setting
C the tag field to -1 or something letting the slave
C purge the output file.
      IF(ERR .NE. 0)WRITE(LU,130)ERR
130   FORMAT("/FCPY: PWRIT ERROR = "I3)


C We are now finished. Tell the slave we are finished
C by sending PCONT.
      CALL PCONT (PCB,ERR,TAG)

C Close the input file.
      CALL CLOSE (DCB)
      END
```

```
FTN4,L
      PROGRAM FCPYS (3,50), PTOP File Copy (Slave Program)

C*********************************************************************
C
C   This program receives data from the master program, FCPY,
C   deblocks the data in the incoming buffers and stores the data
C   in a file.
C
C*********************************************************************

C The following example program has minimum error checking and
C processing code to keep this example relatively simple. More
C useful error checking and error messages should be written for
C a production program.

      IMPLICIT INTEGER (A-Z)
      DIMENSION BUF(1024),TAG(20),P(5),DCB(144)

C Call RMPAR to get class number for class get issued
C   by (CALL GET).

      CALL RMPAR (P)

C I/O CLASS number passed in parameter 1.
      CLASS=P(1)

C Now sit and wait for some action.

      CALL GET (CLASS,ERR,FCN,TAG,LEN)
      IF (ERR .NE. 0) STOP 1

C FCN= 1 POPEN
C      2 PREAD
C      3 PWRIT
C      4 PCONT

C On POPEN, the tag field contains the destination file NAMR.

      GO TO (10,25,25,25)FCN

C Create the destination file.

C Information in the tag field is the result of a call to NAMR
C by the Master program. The routine NAMR and its output is
C described in the Relocatable Library Manual for your system.

C                         Name, Len   , Type , SC    , CRN
10    CALL CREAT(DCB,ERR,TAG,TAG(8),TAG(7),TAG(5),TAG(6))
      IF(ERR .GE. 0) GO TO 30
```

```
C Process error and send to master program

25    CALL REJCT (TAG,ERR)
      CALL FINIS
      STOP 3

30    CALL ACEPT (TAG,ERR,BUF)

C   At this point we don't want to accept any more POPEN's
C   from any other master programs until we complete this transfer.

40    CALL GET (CLASS,ERR,FCN,TAG,LEN)
      GO TO (50,50,60,70)FCN

C Reject all POPEN attempts until we exit.

50    CALL REJCT (TAG,ERR)
      GO TO 40

C Process PWRIT.

60    CALL ACEPT (TAG,ERR,BUF)

C Unpack the buffer and WRITE records to destination file.
C BUF(1) is the first word in the buffer and equals the
C number of records in the buffer. LEN is the second word
C in the buffer and equals the record length. IX is set to
C 3 to begin reading the third word in the buffer. After a
C record is read, IX is reset to equal its original value plus
C the length of the record just read, plus 1 to pass the word
C containing the length of the next record.

      IX=3
      DO 65 I=1,BUF(1)
      LEN= BUF (IX-1)
      CALL WRITF (DCB,ERR,BUF(IX),LEN)

65    IX=IX+LEN+1

C Go back to the GET statement and wait for the next buffer.

      GO TO 40

C Handle PCONT and close file when all buffers have been passed.

70    CALL ACEPT (TAG,ERR,BUF)
      CALL CLOSE (DCB)
      CALL FINIS
      END
```

Chapter 7
DS/1000-IV Utility and Program Calls

Utility Calls

Utility routines are available which allow you to perform special functional operations prior to calling RFA or DEXEC routines. In addition to the special functions, the utility routines provide you with capabilities that are not available through the conventional program communication calls. The calls to the utility routines are described in the following paragraphs. Included are descriptions of the DS/1000-IV utility routines for:

1) Gaining non-interactive access to a remote account.

2) For sending a message to a remote system console or to a remote message processor.

3) Obtaining your local system's node number.

4) Down-loading program files into an RTE-MIII, RTE-IVE or RTE-L/XL/A.

5) Accessing a DS/1000-IV version of the RTE Interactive Editor.

6) Obtaining extended error information.

See the Remote File Access Chapter in this manual for additional information on syntax conventions used in the following program calling sequences.

The utilities used for accessing an HP 3000 are HELLO, PRCNM, and BYE. These HP 3000 access utilities are described at the end of this chapter.

NOTE

Program calls to DS/1000-IV utility routines are not supported in RTE-6/VM type 6 (Extended Background) programs.

DLGON Utility

DLGON provides non-interactive access to a specific remote account at another HP 1000 node. DLGON creates a session at the requested node. The created session has the capability, cartridge list and Session Switch Table (SST) defined for the specified account, but no configuration LUs are mapped in. DLGON causes the same log-on message at the requested node as a local session log-on. Session identifiers returned by DLGON are LU numbers not otherwise in use at the requested node. A subsequent RFA, DEXEC, or PTOP call from this program to the requested node causes a temporary attach to this remote account for the duration of the call.

A user program may have active sessions at a maximum of 16 nodes at a time and one active session per node. A subsequent DLGON or DLGNS call logs off a previous remote session owned by the program. (See the section "Network Account Table" in the Network Manager's Manual, Vol. II, Appendix C for information on increasing the number of nodes that a program can have access to.)

If the specified node does not have session monitor, -4 is returned in IERR (same as RS04). DLGON calls to the local node are not allowed (IERR= -7; same as RS07).

NOTE: HP does not support DLGON calls to non-Session Monitor systems.

CALLING SEQUENCE

```
            CALL DLGON (IERR,NODE,IACCT,LEN[,IORIDE])

   OR

            INTEGER DLGON
               :
               :
            ISESS = DLGON (IERR,NODE,IACCT,LEN[,IORIDE])
```

ISESS The session identifier of the remote session created, or the negative of the session identifier of the existing remote session that is being used. (See the following section on Session Sharing.)

IERR A variable to which logon error codes are returned. Zero is returned if there were no errors. See Appendix B for a description of DLGON error codes.

NODE	The positive node number or negative LU number of the remote node at which the account is located. An error is reported if the local node number is entered here.
IACCT	The name and optional password of the remote account in the form "user.group/password". "User", "group", and "password" can each be up to ten characters in length.
LEN	The length of the account name in positive (+) words or negative (-) bytes.
IORIDE	An integer specifying the degree to which you want to override session sharing. For most applications, the default value (0) is sufficient. If your program must own a session (making it impossible for other programs to terminate that session), set IORIDE to 2. See the following section on Session Sharing for more information.

Session Sharing

A program's DLGON request is normally refused if a session that is owned by another program within the calling program's process group exists at the requested node. Refusal is signified by -6 returned in IERR. The calling program then shares the session that caused the request to be rejected.

The following terms are used in this section:

ANCESTOR	The program that scheduled the calling program (its parent) or that parent's parent, etc.
PROCESS GROUP	Programs are in the same process group if they are scheduled from the same session; for non-session systems, if they are scheduled from the same terminal. On systems with session capability installed, each program that runs outside of session is in its own process group, but can still be tied to ancestors.
OWNER	The program that creates a session (via DLGON or implicitly by any other DS request) is considered the owner of that session. Only the owner is allowed to log off that session (using DLGOF). If the owner terminates, that session is also logged off. If the program that owns the session terminates or logs it off, all programs running under that session are aborted, and the session's SST is destroyed.

SLAVE BACK	If the program that created the remote session (the owner) that the calling program is running under exists at the requested node, DS traffic may "slave back" to the session that the owner is running under.

Your program can share an already-established remote session or can override session sharing with the IORIDE parameter. If a session which you are willing to share already exists at the requested node, a -6 is returned in IERR and you then share the session that already exists. IORIDE values indicate:

2 Absolute session sharing override. Do not try to share any existing session.

1 Override session sharing with non-ancestors. Share with ancestors. Request a new session even if a non-ancestor program within the calling program's process group already owns a session at the requested node.

0 (Default for DLGON/DLGNS) No override. Share with ancestors or other programs within the calling progam's process group.

-1 (Default for all non-DLGON/DLGNS DS access) "Slave ability." No override. The log-on request is refused as above. However, slave back, if possible.

For example, suppose program A, running under session 10 at node X, creates (with DLGON) session 15 at node Y and then schedules program B. B then issues a DLGON call to node X, with IORIDE= -1. Its DLGON call is refused, and -10 (the negative of the session identifier of the session that created B) is returned in ISESS. Subsequent DS traffic from program B to node X slaves back to session 10. Any programs scheduled by B could also slave back to session 10.

NOTE

If the requested node's sub-level is 0, IORIDE is forced to 0, as these nodes do not support session sharing override. If the log-on request is refused, the error qualifier is set to 1 (DSERR returns RS06(1)). See the Network Manager's Manual, Vol. II, Appendix A for more information on sub-levels.

The DLGON IORIDE parameter only controls access to a session at the time the session is created. Thus, a session created with IORIDE= 2 could still be shared by other programs, if they connect to it (by rejected DLGON or other DS requests) after it is created. However, the program that creates the session is still the only one that can terminate the session. An IORIDE value of 2 ensures that the calling program will not have its session terminated by another program.

DLGOF Utility

The DLGOF routine releases the session at the specified remote HP 1000 node. Private and group cartridges, if any, are saved and any active programs in the remote session are aborted. This call is ignored if a session does not exist at the specified node.

CALLING SEQUENCE

```
CALL DLGOF (IERR,NODE)
```

IERR	A variable to which log-off error codes are returned. Zero is returned if no errors occurred. See the appendix on error codes for a description of DLGOF error codes.
NODE	The positive node number or negative communication link LU number of the remote node at which the session is located. An error is reported if the local node number is entered here.

The DLGOF call results in the same system console messages at the destination node that are displayed for local terminal log-offs.

If your program needs to access more than 16 nodes during a single execution, issue a DLGOF call to the nodes no longer required, whether session or not. This releases the table entries.

DLGNS Utility

The DLGNS routine allows non-session access to a specific remote HP 1000 node with Session Monitor. The user must specify the password defined by the system manager of the remote node. If the specified node does not have session monitor, -4 is returned in IERR (same as RS04).

NOTE: HP does not support DLGNS calls to non-Session Monitor nodes.

CALLING SEQUENCE

```
        CALL DLGNS (IERR,NODE,IACCT,LEN[,IORIDE])

  OR

        INTEGER DLGNS
           :
           :
        ISESS = DLGNS (IERR,NODE,IACCT,LEN[,IORIDE])
```

ISESS	The session identifier of the remote session created, or the negative of the session identifier of the existing remote session that is being used. (See the previous section on Session Sharing.)
IERR	A variable to which error codes are returned. See Appendix B for a description of DLGNS error codes.
NODE	The positive node number or negative communication link LU number of the remote session monitor node. An error is reported if the local node number is entered here.
IACCT	The password for non-session access at the target node. The password can be up to ten characters in length and can optionally be preceeded by a slash.
LEN	The length of the password in + words or - bytes, including the optional leading slash.

IORIDE An integer specifying the degree to which you want to override session sharing.

Non-session access to a remote node is generally reserved for network and/or system managers. A password is required to gain this type of access. The password is set during DINIT initialization and can be changed by running DSMOD (/P command). In non-session mode, only system global cartridges can be accessed. There is no access to session cartridges. There are no capability checks on RTE operator commands.

The IORIDE parameter for DLGNS determines which programs the calling program is willing to share DS access with, whether the existing access is session or non-session. These programs are the same for the IORIDE parameter values set for DLGON. If, however, the request for log-on is not refused, non-session access is given, rather than session access.

DMESG Utility

You can send an ASCII-coded message of up to 72 characters from your program to a remote HP 1000 node's system console. Because the destination nodes have non-interactive sessions, session LU 1 in the SST (Session Switch Table) will map to system LU 1. When the message arrives at the destination device, it is displayed together with a prefix message that identifies the sending node. The message format is:

(FROM NODExxxxx): message text

xxxxx	is the node number where the message originated.
message text	is up to 72 characters of ASCII-coded text.

CALLING SEQUENCE

```
        CALL DMESG(IDEST,MSGAD,MSGL)
```

IDEST	Destination node address where the message is to be sent.
MSGAD	ASCII-coded message; less than or equal to 72 characters.
MSGL	Message length; a positive value to indicate number of words, or a negative value to indicate number of characters in the message.

If an error condition is encountered, a 4-character error message is returned to the A-register and B-register (for example, DS05) upon return to your program. These error messages are defined in the error code appendix. Extended error information can be obtained by a call to DSERR which is described in this chapter.

DMESS Utility

This utility is used to transmit a system command to the message processor at a remote HP 1000 node. The buffer defined in the call to DMESS is also used to receive any reply from the remote node. The message buffer contents may be overwritten following execution of this utility, whether the reply is returned or not. If there is a remote session, the system command will be executed under that session and be subject to capability restrictions of that session.

```
***********
* WARNING *
***********
```

Issuing the system command "OF,<program name>" via DMESS, where <program name> is the name of any DS/1000-IV program may result in the failure of a requested transaction and/or close files or terminate programs. DMESS is the remote processing version of the RTE utility, MESSS. Refer to the appropriate Operating System reference manual for more information regarding the operation of this utility.

CALLING SEQUENCE

```
CALL DMESS(IDEST,IBUFR,IBUFL)
```

IDEST	Destination node address where the message is to be sent.
IBUFR	Message buffer; less than or equal to 40 characters.
IBUFL	Message buffer length in bytes; a positive value indicating the number of bytes in the buffer (40 bytes maximum).

NOTE: Many RTE-A system messages exceed the IBUFR size limit and are truncated.

Upon return to your program, the A-register and B-register contain status information:

A = 0	Indicates that there is no return message from the remote node.
A < 0	Indicates the negative value of the byte count for a return message from the remote node. The return message is contained in the message buffer (IBUFR).
B = -1	Indicates an error condition. In this case, the A-register contains -4 and IBUFR contains a 4-character error message (for example, DS03).

Extended error information can be obtained by a call to DSERR which is described in this chapter.

DNODE Utility (RMTIO)

You may use the DNODE utility to cause FORTRAN READs and WRITEs to be executed at a remote HP 1000 node. The DNODE subroutine is contained in $FDSLB. DNODE is not supported by FORTRAN 4X or 77. Consult the appropriate FORTRAN manual to accomplish remote reads and writes for these versions of FORTRAN.

CALLING SEQUENCE

```
            CALL DNODE(NODE)
```

NODE	The destination node number for the remote I/O operation. The default setting (before calling DNODE) is -1 (local node). To return to local use, CALL DNODE(-1).

If the node number specified in the DNODE call is a Session Monitor node, then a prior call to DLGON is required to establish a specific session at that node, or the user can elect to take the default account.

Upon return from the call to DNODE, the A-register contains the original destination node number, and the B-register contains the new destination node number. The content of these registers may be obtained by referring to DNODE in a FORTRAN function reference statement such as:

```
    REG=DNODE(NODE)
```

where:

REG is a real variable

Once set, the destination node number remains in effect until explicitly changed by another call to DNODE or until execution of the calling program terminates.

Because the value of the NODE parameter initially defaults to -1, it is not necessary to call DNODE to perform I/O processing at the local node.

NOTE

In RTE-MIII and RTE-IVE, the DNODE subroutine is contained in the module %RMTIO. When loading programs making calls to DNODE use the following RTMLG command file.

```
LOAD
TR,<snap'file>
ECHO ON <list file>
MAP MODULES ON <list file>
OUTPUT ON <output file>
REL %PROG
SEARCH %RMTIO
SEARCH <other libraries>
   .
   .
   .
END
```

DSERR Utility

You may use the DSERR utility to make an error check and/or determine complete information about a reported error in the most recent HP 1000 request. It is a good idea to call DSERR if an error occurs on any DS subroutine call, then print the returned buffer rather than the original IERR value. This method will provide more information. DSERR can be used to report errors in HP 1000 to HP 1000 communication only; it returns erroneous results if used to report errors in calls to an HP 3000.

CALLING SEQUENCE

```
CALL DSERR(IERBF[,NODER[,LQLFR]])
```

IERBF (Returned). ASCII error message return buffer. Must be an integer array of at least 24 words. The message is of the form:

```
DS ERROR: XXXX(QQ), REPORTING NODE NNNNN
```

where: XXXX field may contain either the ASCII or numeric (converted to ASCII) error code.

QQ = Error Code Qualifier.

NNNNN = Node number reporting the error.

For example:

```
DS ERROR: DS04(1), REPORTING NODE 500
DS ERROR: FM-32(0), REPORTING NODE 1
```

Error codes and qualifiers can be found in the error appendix of this manual.

NODER (Returned). A positive integer to which the reporting node number is returned. (Optional).

LQLFR (Returned). A positive integer to which the qualifier code is returned. (Optional).

Following any DS operation in which an error has occurred (regardless of the level at which the error was detected), information describing the cause of the error is placed in a portion of the request buffer.

Upon being called, DSERR obtains this information, converts it into various fields of a local message buffer, then moves this buffer into the message buffer (IERBF).

A recommended procedure when making DS/1000-IV requests of any type is to make the request, check for an error, and continue if there is none. Otherwise, call DSERR to obtain complete information about the error and print the returned message buffer along with some message identifying what was being attempted. Since DSERR returns a superset of the error information returned in the previous DS routine (DEXEC, DOPEN, etc.), there is no need to print the value of IERR or the A and B registers on return.

Remote Interactive Editor Utility

The DS/1000-IV remote editors are available only on M/E/F-Series machines. "EDITR" is used at all M/E/F-Series machines except those with RTE-MIII and RTE-IVE operating systems. The remote editor "REDIT" is used at RTE-MIII and RTE-IVE systems. Refer to the RTE Interactive Editor Reference Manual for a description of the EDITR commands.

DS/1000-IV provides several methods of editing files residing on remote nodes. The recommended method is to use Remote I/O Mapping. See your Network Manager to determine if this feature is available in your network. Remote I/O Mapping enables you to access and interact with a remote node as if your terminal were connected directly to it. If both your local node and the remote node have DS transparency software, you may run the editor at the remote node by specifying the remote node number in the editor program's file namr. There may be occasions, however, to use the remote EDITR utility. The remote version of the editor may be scheduled in four similar ways depending on the type of system located at the local and remote node (see Table 7-1).

RTE-IVB or RTE-6/VM Session Monitor Target Node

When the target node (the node at which you wish to have the remote editor scheduled) is a RTE-IVB or RTE-6/VM session node, you should schedule the editor from REMAT. Since the utility program EDITR cannot by itself create a session to gain access to files on private or group cartridges, you should use REMAT to create the remote session for you and then use the REMAT "RW" (run with wait) command to cause the EDITR to run in the remote session.

Besides creating a remote session, this method also obtains the reverse linking to the originating session, if any, required for interactive I/O. (Refer to the chapter on DEXEC calls.) Another advantage to using the REMAT "RW" command is that a copy of the remote EDITR will be automatically created and scheduled.

To schedule a remote editor located at an RTE-IVB or RTE-6/VM node with Session Monitor, use the following sequence of commands:

```
:RU,REMAT

$SW,destination node[:user.group/password]
    [,local node],security code
```

```
#RW,EDITR,[lu],[line length],destination node,
          local node[,flag]
```

lu	The session logical unit number of the I/O device from which you enter EDITR commands. If omitted, logical unit 1 or, the system console LU is assumed.
line length	Maximum output record length in characters; specification is optional. If omitted, a length of 150 characters is assumed.
destination node	Network node at which EDITR is to be scheduled and at which the file to be edited resides. If the "destination node" parameter is -1 or 0 or null and the "flag" parameter is 0 or null, the local node is assumed to be the destination node ("destination node" = -1 always indicates local node). If the "destination node" parameter is 0 or null and the "flag" parameter contains a non-zero value, node number 0 is declared as the destination node (see the "flag" parameter description below).
local node	The local node at which REMAT has been run.
flag	This parameter should be used only if you want to schedule EDITR in node number 0 and node number 0 is not the local node. If this is the case simply specify the destination node as 0 or null and specify any non-zero value in the "flag" parameter.

The remote Interactive Editor allows access to the logical source (LS) area only for edit operations with the local node. In response to the Interactive Editor prompt "SOURCE FILE," the following replies and results are defined:

Response	Result
file name	Declares an existing file to be edited. The file must reside in the same node as the scheduled Interactive Editor.
blank character	Assigns a null file for remote edit; or assigns the LS area for local edit.
0 (zero)	Assigns a null file unconditionally.
: (colon)	Aborts the remote Interactive Editor immediately.

Remember when scheduling any program with the REMAT "RW" command, the program is subject to a 20-minute execution time limit. After 20-minutes, the remote editor will abort and any edits made will be lost. Editing via Remote I/O Mapping is not subject to this limitation. If you plan to do any major editing of a remote file, and cannot use Remote I/O Mapping, and your local node supports an editor and file system, transfer the file to your local node and perform the editing there.

Refer to the appropriate manual for information describing the EDITR commands.

RTE-IVB, 6/VM Multi-Terminal Monitor Node

Scheduling the remote editor at an RTE-IVB or RTE-6/VM MTM node is similar to scheduling the remote editor at RTE-IVB and RTE-6/VM nodes with Session Monitor (see above). The only difference is MTM nodes will not automatically schedule a copy of EDITR. Instead, the original version will be used and this may prevent anyone at the local node (the node where the editor is being scheduled) from scheduling the editor until you are finished.

For this reason, the Network Manager should make several copies of EDITR available at the MTM node. Each copy may be specified as EDIxx, where xx is a two-character suffix distinguishing the different copies of the program. See your Network Manager regarding what copies of EDITR are available to you at the MTM target node.

Scheduling From REMAT at any Node

One method of scheduling a copy of the remote editor at an RTE-IVB or RTE-6/VM Multi-Terminal Monitor node from an RTE-IVB or RTE-6/VM Session Monitor node, or from an RTE-L/XL/A node, is to use the following sequence of commands:

```
:RU,REMAT

$SW,destination node,local node,security code

#RW,EDIxx,lu[,line length],destination node,
          local node[,flag]
```

The parameters are the same as described under RTE-IVB and RTE-6/VM Session Monitor Target Node except for the appropriate suffix (xx) required to schedule the correct copy of the remote editor and you must specify the system LU rather than the session LU.

Note that you have a 20-minute time limit to complete your editing and exit. This time limit is imposed whenever you schedule a program from the REMAT "RW" command (as described in the previous section).

Scheduling From RTE-IVB or RTE-6/VM

Due to the 20-minute execution time limit imposed by scheduling from REMAT, an alternate method exists to schedule the remote editor at a RTE-IVB or RTE-6/VM Multi-Terminal Monitor node from a RTE-IVB or RTE-6/VM node running under Session Monitor. Scheduling the remote editor from an RTE-L/XL/A, RTE-MIII or RTE-IVE using this method cannot be used.

From the system prompt, use the following run string:

```
RU,EDITR,lu[,line length],destination node[,flag],suffix
```

where:

suffix	The last two characters in the name of an alternate copy of EDITR located at the destination node. For example, if the destination node contains a copy of EDITR named EDIT9, you must specify "T9" as the "suffix" parameter. This suffix has the same purpose as described in the Scheduling from REMAT section.

Scheduling From RTE-MIII and RTE-IVE

When scheduling a remote editor at an RTE-IVB or RTE-6/VM Multi-Terminal Monitor node from an RTE-MIII or RTE-IVE node, an alternate method exists similar to the method mentioned in the previous section.

From the system prompt, use the following run string:

```
RU,REDIT,lu[,line length],destination node,,suffix
```

Note that there is no "flag" parameter in this run string for REDIT. The two commas, however, are required when specifying the "suffix" parameter to be used as parameter place holders.

The "suffix" parameter is defined in the same manner as described in the previous section.

Remote WHZAT Utility

The RTE system status utility WHZAT has been modified for DS/1000-IV. You can schedule it for remote execution from any RTE node to obtain program status or other system information provided by WHZAT. It can be scheduled from your local node to display the program status of a remote node to your terminal screen. Note that the DS version of WHZAT must be available for execution at the remote node. The DS version of WHZAT is %WHZDS for RTE-IVB and %WHZ6D for RTE-6/VM. For RTE-L/XL/A systems, the program status can be obtained with the PL command.

The remote WHZAT runstring is:

```
:RU,WHZAT,LU,option,DS node #
```

where:		
	LU	is the output device LU. Default is your terminal.
	option	is the WHZAT program option, AL, PT, PL, etc.
	DS node #	is the node where the output is to be returned. This must be your local node to display the WHZAT output on your terminal.

EXAMPLE:

```
:RU,REMAT
$SW,111,,DS                (Node 111 is the system in question.)
#RU,WHZAT,,AL,106          (Node 106 is the local system where
    :                      the program status of node 111 is to
    :                      be displayed.)
    :
 (WHZAT output)
    :
#
```

Table 7-1. What Editor To Use

You are at	You want to run EDITR at	Command string
RTE-IVB, 6/VM MTM; RTE-L, XL, A; RTE-MIII, RTE-IVE	RTE-IVB, 6/VM SM	:RU,REMAT $SW,destination[:user.grp/password], [local],sc #RW,EDITR,[syslu],[line length], destination,local[,flag]
RTE-IVB, 6/VM SM	RTE-IVB, 6/VM SM	:RU,REMAT $SW,destination[:user.grp/password], [local],sc #RW,EDITR,[seslu],[line length], destination,local[,flag]
RTE-IVB, 6/VM SM; RTE-IVB, 6/VM MTM	RTE-IVB, 6/VM MTM	:RU,REMAT $SW,destination,[local],sc #RW,EDIxx,[syslu],[line length], destination,local[,flag] or RU,EDITR,[syslu],[line length], destination,local[,[flag][,suffix]]
RTE-L, XL, A	RTE-IVB, 6/VM MTM	:RU,REMAT $SW,destination,[local],sc #RW,EDIxx,[syslu],[line length], destination,local[,flag]
RTE-MIII, RTE-IVE	RTE-IVB, 6/VM MTM	:RU,REMAT $SW,destination,[local],sc #RW,EDIxx,[syslu],[line length], destination,local[,flag] or RU,REDIT,[syslu],[line length], destination,,suffix

where:

destination	Destination node where editor will execute.
user.grp/password	Account and password in the specified node to which subsequent REMAT commands will attach.
local	Local node at which REMAT has been run.
sc	The local node's network management security code.
flag	Used to schedule the editor in remote node 0 (zero).
seslu	Session logical unit to which output will be directed.
syslu	System logical unit to which output will be directed.
line length	Maximum output record length in characters.
suffix	The last two characters in the name of an alternate copy of the editor.

Remote Interactive Query Utility

For those experienced with using QUERY/IMAGE DATABASE MANAGEMENT SYSTEMS, a remote version of QUERY exists as part of the 92069A IMAGE/1000 product. The remote database may be accessed either through REMAT or programmatically.

The utility program QUERY cannot by itself create a session to gain access to files on private or group cartridges. When scheduling QUERY at a remote Session Monitor node, the user should use REMAT to create the remote session, then use the REMAT "RW" (run with wait) command to cause remote QUERY to run in the remote session.

Besides creating a remote session, this method also obtains the reverse linking to the originating session required for interactive I/O.

Another advantage to using the REMAT "RW" command is that a copy of the remote utility will be scheduled.

If remote QUERY is scheduled programatically, a call to DLGON should be made to create the desired remote session, or create the remote session with REMAT using the "AT" command (see the ATTACH command under Remat Operator Commands) and then SWitching to your local node before issuing the REMAT "RW" command.

Refer to the "IMAGE/1000 92069A and 92073A Database Management System Reference Manual" (part number 92069-90001) or the "IMAGE/1000-II 92081 Database Management System Management Reference Manual" (part number 92081-90001) for a detailed discussion on accessing remote QUERY.

FCOPY Utility

The FCOPY utility may be used to copy a file from one node to another node. Upon completion of FCOPY execution, the copy will be the same security type and size as the original file unless you specifically declare a different security type or size for the copy within your call to FCOPY.

If you use the RTE command BR to terminate execution of FCOPY, the error code -100 (programmatic break detected) is returned.

CALLING SEQUENCE

```
CALL FCOPY(IFIL1,ICR1,IFIL2,ICR2,IERR[,ISEC1[,ITYP2
          [,ISIZ2[,IREC2[,IMODE[,ISEC2]]]]]])
```

IFIL1	Origin file name; a 3-word array containing the ASCII-coded name of the file to be copied.
ICR1	Two word integer array. Word one is the cartridge identifier, where a positive value indicates cartridge reference number, negative value indicates LU, and zero indicates that the file search is not restricted to any particular cartridge. Word 2 indicates the node of the origin file. (See RFA Calls Chapter of this manual.)
IFIL2	Destination file name; a 3-word array containing the ASCII-coded name of the file copy.
ICR2	A two word error containing the destination file cartridge identifier and node number (see ICR1, above).
IERR	Error return variable.
ISEC1	Origin file security code (optional); specify only if a security code exists for the origin file.

ITYP2	Destination file type (optional); specify only if you want the destination file type to differ from that of the origin file.
ISIZ2	Destination file size in blocks (optional); specify only if you want the destination file size to differ from that of the origin file. If specified, this value must be greater than the block size of the origin file. The value specified here may not be negative because a line failure that occurs before the file is truncated and closed would result in all of the available remote disc space being allocated to the destination file. If you attempt to specify a negative value, error code -6 results.
IREC2	Destination file record size in words - Type 2 files only (optional); specify only if you want the record size of the destination file to differ from that of the origin file. It may not be a negative value.
IMODE	Transfer mode (optional); specify a non-zero value in this parameter to cause both the origin file and the destination file to be opened as Type 1. It is usually desirable to enable this mode because variable record length files are then transferred in blocks of 128 words which is faster than record-by-record transfers.

```
***********
* WARNING *
***********
```

Do not declare the IMODE parameter if the origin file has file extents. Extents are not copied if a file is opened as Type 1. Failure to observe this caution will result in a corrupt copy of the file although no error return is generated.

ISEC2	Destination file security code (optional); specify only if a different security code is desired on the destination file.

When the remote node is Session Monitor and the destination file resides in a private or group cartridge mounted under a specific account, you must make a prior call to DLGON or take the default account. If the origin and destination files are at two different Session Monitor nodes other than the local node, then two prior DLGON calls are required, one for each node (or you may take the defaults).

Another way to access a specific account is to run REMAT, ATach to a specific account, SWitch to your local node and then run with wait (RW) your local program. All remote requests will be directed to the remote session account specified in the REMAT ATach command.

If a file of the same name as IFIL2 (the destination file) exists in the destination file's cartridge, an attempt is made to create a unique file name by replacing the first two characters of the destination file name with period characters (..). If this fails to generate a unique file name, control returns to the calling program with an indication that the FCOPY operation was not successful. If the substitute file name is unique, IERR = 1 will be returned.

If you do not specify the IMODE parameter, or if you specify IMODE as zero, the source file record length cannot exceed 128 words or error code -104 (record size error) occurs. You may avoid this record length limitation by specifying a non-zero value in IMODE. However, the CAUTION concerning source file extents in the parameter descriptions is applicable in any case.

NOTE When copying files with odd byte-length records from an RTE-A system to a non RTE-A system, you can not use block transfer; do not set IMODE to a non-zero value.

FLOAD Utility

You may use this utility to down-load an absolute or memory-image program file into an RTE-MIII system, or a memory-based RTE-A, L or XL system. This utility may not be used for RTE-IVE, IVB or 6/VM.

The FLOAD utility generates a request to DS/1000-IV APLDR at the RTE-MIII or memory-based RTE-L/XL/A node for initiation of a down-load operation. The absolute program file must have been previously produced by the RTE-MIII Loader/Generator or the RTE-L/XL/A Loader. (Refer to the chapter on On Line Loaders.) Files for RTE-L/XL/A are standard RTE-L/XL/A format. The proper version of APLDR must exist at the remote node. The technique described here is not applicable if the "mini-APLDR" is used.

CALLING SEQUENCE

```
CALL FLOAD(NAME,ICR,NODE1,NODE2,IERR[,ISECU[,IPRTN[,IPSIZ
                      [,IERBF]]]])
```

NAME	Absolute program file name; a 3-word array containing the ASCII-coded name of the program file to be down-loaded. The first five characters of the file name are used to fill in the ID segment.
ICR	Program file's cartridge reference number.
NODE1	Node address at which the program file exists; this must be a non-negative node number.
NODE2	Destination node address at which the program file is to be down-loaded.
IERR	Error return variable. Refer to the error code appendix under FLOAD Error Codes.
ISECU	Program file security code (optional).
IPRTN	Destination memory partition number (optional for RTE-MIII, not used in RTE-L/XL/A.) If not specified, APLDR uses the first available partition, regardless

of its size.

IPSIZ	Destination partition size in pages (optional for RTE-MIII, not used in RTE-L/XL/A.) If not specified, APLDR may attempt to load the program into a partition that is too small to contain the program.
IERBF	Error buffer (optional); a 3-word array used for the return of an ASCII-coded program name under certain error conditions (see below).

A call to FLOAD can be executed within an RTE-MIII or RTE-L/XL/A node to down-load a program file into itself. In this case, specify the local node number as the NODE2 parameter.

The array defined by the IERBF parameter may be used to receive return information upon completion of a call to FLOAD. For example:

IERR Content	IERBF Content	Meaning
0	-	Successful down-load operation completed.
-60	PROGA	A program named PROGA occupies memory space needed by the program being down-loaded.
-61	..OGA	Duplicate program name at destination node. The program has been renamed by replacement of the first two characters of its name.

The program file to be down-loaded must reside on a system cartridge.

GNODE Utility

You may call this utility to obtain your local node number.

CALLING SEQUENCE

```
          CALL GNODE(NODE)
```

NODE	Node number return variable. Will be -1 if DS is not initialized.

HELLO (HP 3000 Logon Utility)

You may use the HELLO utility to establish communication between your RTE system and an HP 3000. This utility creates a remote Session Main Process (SMP) at the HP 3000. This SMP acts as a logical extension to the local process. You must execute the HELLO utility before any programmatic interfacing, such as RFA or PTOP calls, is initiated. The result of calling this utility is equivalent to the RMOTE HELLO command. See the discussion of RMOTE in this manual for the procedure required to execute your programs within the SMP established by issuing the HELLO command.

NOTE: If you are using the HELLO utility in a segmented PTOP master program it must be called from the main program.

CALLING SEQUENCE

```
CALL  HELLO(IERR,LDEV,LSTDV,NMSMP,LOGR,LOGRL[,LUX.25])
```

IERR	An error code is returned here if an error condition is encountered when issuing the HELLO command. Upon successful completion of the call to HELLO, the value of IERR is zero. See the appendix on errors for a list of HELLO utility error codes and definitions.
LDEV	The logical unit number of an HP 3000 (an integer <256) or an X.25 address (up to 15 ASCII-coded digits).
LSTDV	The logical unit number of the desired list device. The "logon" response generated at the HP 3000 as a result of a successful HELLO operation is transmitted to this device.
NMSMP	The Session Main Process (SMP) number is returned here from the HP 3000.
LOGR	An integer array that contains the HELLO command logon parameters in the form of a message. The first six characters of this logon message must be the characters HELLO followed with a blank. The entire message string also must be followed (terminated) with a blank.
LOGRL	The length (in characters) of the logon message contained in the LOGR array.
LUX.25	The LU associated with an X.25 network. If the LUX.25

parameter is not passed and an X.25 address is passed in LDEV, HELLO passes zero in LUX.25. This zero is used by the X.25 virtual circuit allocation routine which indicates that X.25 will use the first network in SAM (i.e., the last network entered by the user from XINIT).

For a complete description of the syntax of the HELLO command, refer to the HP 3000 MPE Commands Reference Manual.

If you are using a PSI BISYNC connection and the line is not up when you call HELLO, HELLO automatically schedules DSLIN with default parameters. (DSLIN must be RP'ed.) If you want non-default DSLIN parameters, run DSLIN before calling HELLO.

BYE (HP 3000 Logoff Utility)

Following completion of your operations at the HP 3000, you may terminate the Session Main Process via a call to the BYE utility. This utility issues a BYE command which terminates communication between your RTE program and the HP 3000.

The BYE utility call should be issued by the same program that issued the HELLO utility call for the current SMP.

If your program aborts abnormally or terminates without issuing a BYE command, the UPLIN module automatically issues a kill job command. In this case, the logoff response from the HP 3000 is transmitted to your system console (logical unit 1).

CALLING SEQUENCE

```
        CALL BYE(IERR,LDEV,LSTDV,NMSMP)
```

IERR	An error code is returned here if an error condition is encountered when issuing the BYE command. Upon successful completion of the call to BYE, the value of IERR is zero. See the appendix on errors for a list of BYE utility error codes and definitions.
LDEV	The logical unit number of an HP 3000 (an integer <256>) or an X.25 address (up to 15 ASCII-coded digits).
LSTDV	The logical unit number of the desired local list device. The "logoff" message generated by the HP 3000 as a result of a successful BYE operation is transmitted to this device.
NMSMP	The Session Main Process (SMP) number obtained via this sessions corresponding HELLO command.

PRCNM (HP 3000 Process Number Utility)

This utility is used to establish communication between a program and a Session Main Process (SMP) created by a "father" program. For example, if you establish a session on an HP 3000 by issuing the "HELLO" command from RMOTE, any program you run from RMOTE may issue a call to PRCNM to establish communication with the same session. PRCNM will fail if there is no father program, or if the father has not established a session on the HP 3000.

CALLING SEQUENCE

```
            CALL PRCNM(IPARM)
```

IPARM May be any non-zero value to establish communication with the HP 3000 session. If the node on which the calling program is running has an RTE-MIII, RTE-IVE or memory-based RTE-L, XL or A operating system, "IPARM" can be set to zero to clear the communication values set by a previous run of the program.

The program that initially issues the HELLO call (the "father") must not terminate before the program that calls PRCNM (the "son") completes execution. If this occurs, the UPLIN module will issue a kill job command to terminate the HP 3000 session. The "father" program should also issue the BYE call.

If PRCNM fails because there is no father program, or the father has not established a session on the HP 3000, a -1 is returned in the B-register. If the call completes successfully, a zero is returned in the B-register. If PRCNM is called with a zero value in "IPARM," zero will always be returned in the B-register.

For an example of of using the HELLO , BYE and PRCNM, see the program at the end of this chapter.

LU3K (for X.25 Links)

If you use (pooled) X.25 virtual circuits for HP 1000 - HP 3000 connections, you can obtain the actual lu number of the virtual circuit connection by calling LU3K. This lu number is required by the PTOP "POPEN" call.

A program or one of its ancestors must call HELLO before calling LU3K. If the program's ancestor called HELLO, the program must call PRCNM before calling LU3K.

CALLING SEQUENCE

```
   I3KLU = LU3K(IDUMY)
```

I3KLU	The lu number of the X.25 virtual circuit is returned to this integer variable.
IDUMY	Dummy parameter required for Fortran programs. IDUMY tells the Fortran compiler that LU3K is an external function, not a variable.

```
FTN4,L
      PROGRAM SAMPLE (19)
C  SAMPLE RTE DS/1000/3000 PROGRAM THAT CAN OPTIONALLY EXECUTE
C  UNDER THE RMOTE ISSUED "HELLO" (LOGON).
      DIMENSION IP(5),LGN(40)
C  FIRST 6 CHARS. OF LOGON MUST BE "HELLO ".
      DATA LGN/2HHE,2HLL,2HO ,37*2H  /
C  GET SCHEDULE PARAMETERS.
      CALL RMPAR(IP)
C  I/O LU IS FIRST PARAMETER OR LOGLU
      LU=IP(1)
      IF (LU.LT.1) LU=LOGLU(IDUMMY)
C  CHECK FOR RMOTE'S SESSION.  IF PROGRAM SCHEDULED BY RMOTE,
C  FIFTH PARAMETER RETURNED BY RMPAR WILL BE NEGATIVE.  IF
C  NOT SCHEDULED BY RMOTE, FIFTH PARAMETER WILL BE ZERO OR
C  POSITIVE.  IF SCHEDULED BY RMOTE, RUN UNDER RMOTE "HELLO".
C  SET IPARM TO ANY NON-ZERO VALUE AND PASS IN PRCNM CALL.
      IPARM=1
      IF (IP(5).GE.0) GO TO 10
      CALL PRCNM(IPARM)
      GO TO 20
C  NOT SCHEDULED BY RMOTE.  ISSUE "HELLO" TO THE HP 3000.
   10 WRITE(LU,101)
  101 FORMAT(/"ENTER LU OF HP 3000:")
      READ(LU,*) ILU3K
      IF (ILU3K.LE.0) GO TO 999
   14 WRITE(LU,102)
  102 FORMAT(/"ENTER ACCOUNT NAME")
      READ(LU,103)(LGN(I),I=4,40)
  103 FORMAT(37A2)
C  OBTAIN LENGTH OF LAST FORMATTER I/O REQUEST VIA
C  RELOCATABLE LIBRARY SUBROUTINE, ITLOG.
      CALL ITLOG(IL)
C  REMEMBER FOR LENGTH PARAMETER (IL) THE FIRST SIX
C  CHARACTERS ARE "HELLO ".
      CALL HELLO(IERR,ILU3K,LU,ISMP,LGN,IL+6)
      IF (IERR.EQ.0) GO TO 20
      WRITE(LU,106) IERR
  106 FORMAT("HELLO FAILED WITH ERROR ",I6)
      GO TO 10
   20 WRITE(LU,107)
  107 FORMAT("<BODY OF PROGRAM>")
C  PROGRAM TERMINATION: ISSUE "BYE" IF NOT UNDER RMOTE.
      IF (IP(5).GE.0) CALL BYE(IERR,ILU3K,LU,ISMP)
  999 STOP
      END
```

Figure 7-1. Sample Program (HELLO, BYE, PRCNM).

Chapter 8 DS/1000-IV On-Line Loaders

The DS/1000-IV On-Line Loader, APLDR, allows you to load programs into RTE-MIII, RTE-IVE and RTE-L/XL/A nodes from remote or local files.

There are three versions of APLDR. On RTE-MIII systems, APLDR is loaded from %3APLD; on RTE-IVE systems, APLDR is loaded freom %APL4D. For RTE-L/XL/A systems there is a full-function version of APLDR, loaded from %APLDL (referred to as APLDR-L). However, if you use APLDR-L in disc-based RTE-L/XL/A systems, the REMAT "LO" command and the utility FLOAD are not available. For memory-based (unmapped) RTE-L systems, there is a reduced version of APLDR, loaded from %APLDX (referred to as APLDX). APLDX is recommended for unmapped RTE-L systems and only such systems.

RTE-MIII On-Line Loader

The normal steps taken in loading programs into an RTE-MIII node from either remote or local files is as follows:

1. Run RTMLG with the loader operation. This program prepares relocatable modules for subsequent loading into an RTE-MIII by converting relocatable code to absolute. If the generator operation is used, loading of the generation is done via the Communication Bootstrap Loader.

2. If the program to be loaded is segmented, run SGPRP on the loader's absolute binary output to set up the maximum address bounds used by the main program together with its largest segment. In addition to running SGPRP, make sure your program is coded to include the call SEGLD. This call is used when the main program has completed, and a segment needs to be located, loaded, and executed from a disc file.

3. To bring a copy into memory from your local node, issue the standard RTE-MIII "LO" command, or in a remote RTE node, use the REMAT "LO" command. Specify the file name, security code, cartridge identifier, node number, etc. according to the requirements of the command. The program is loaded into the RTE-MIII system. If the program is segmented, only the main is transferred to memory until a SEGLD call is executed to bring in a segment.

The RTE-MIII Segmented Loader/Generator

The RTE-MIII segmented loader/generator (RTMLG) operates only in a disc-based node. RTMLG can be used to prepare relocatable modules for subsequent loading (via the LO command) into an RTE-MIII node. RTMLG also can be used to generate an RTE-MIII operating system for subsequent loading (via the Communications Bootstrap Loader) into an RTE-MIII node.

Before using RTMLG, it must be loaded into your system. To load RTMLG, issue the following sequence of commands from the loader program:

```
SZ,16
LIB,%LGLIB
RE,%RTMLG
SEA
END
```

To schedule RTMLG for execution of the loader operation, use the RTE program scheduling command in the form:

```
         ,fi,le,nm
RU,RTMLG          [,error lu[,options]]
         ,lu[,,]
```

fi,le,nm	The name of a file that contains input commands. This file must contain a reference to the snapshot created when the particular RTE-MIII node was generated.
lu	The logical unit number of the device from which commands will be entered. If the <error lu> and/or <loader options> parameters are to be specified, the <lu> parameter must be followed by the appropriate number of commas used as parameter position placeholders.
error lu	The logical unit number of the device to which the error log is directed.
options	(Used in loader operations only). The loader options available are defined in the appropriate System Generation Manual.

Once RTMLG is executed interactively without an answer file, an initial query is issued as follows:

```
GEN OR LOAD?
```

You respond either "LO" for loader operation, or "GE" for generator operation. The queries that follow your response are the same as those described for the loader in the appropriate Programmer's Reference Manual, and for the generator in the appropriate System Generation Reference Manual.

RTMLG Considerations

You should be aware of the following considerations when using RTMLG:

1. Unlike the RTE-MIII loader (RTMLG), if echo and map output are directed to the same file, you may specify ECHO OFF and/or MAP OFF at any time.
2. The Mount Cartridge and Dismount Cartridge commands (MC and DC) do not execute in a disc-based node.
3. The Transfer command (TR) can be used only for Loader operations. TR will not execute for Generator operations.

SGPRP — Segmented Program Preparation

Following relocation using RTMLG, segmented programs need to be prepared for execution in RTE-MIII nodes using the program, SGPRP. This preparation process is simply the setting up of the maximum address bounds used by the main program together with its largest segment. These bounds are used by the RTE-MIII Absolute Program Loader (APLDR) to delimit the space that is not available for loading other programs. Further, these bounds are used by LIMEM to determine the amount of free memory above the highest segment.

SGPRP reads the absolute code of the main program and each of the segments to find the highest address used in base page and upper program memory. The high address words in the special records of the main program are then rewritten to reflect the larger bounds.

SGPRP can be scheduled for execution only in a local RTE-IVB, RTE-6/VM or RTE-MIII node to prepare a program for subsequent execution in an RTE-MIII node. SGPRP cannot be scheduled for execution in a remote node.

Within RTE-MIII nodes, SGPRP operates only in the flexible disc environment.

Scheduling SGPRP Execution

SGPRP may be scheduled from a command file or interactively with the following command sequence:

```
                [,fi,le,nm]
        RU,SGPRP
                [,lu]
```

fi,le,nm	The name of the command file that contains the SGPRP input directives.
lu	The logical unit number of the device from which SGPRP input directives will be entered.

For RTE-MIII the <lu> parameter defaults to the operator console.

For hard-disc based systems the <lu> parameter defaults to the interactive device from which SGPRP was scheduled for execution.

SGPRP Input, Non-Interactive Mode

From a file or from a non-interactive device, the SGPRP input directives consist of a list beginning with the main program file name and followed by each segment's name in the form:

```
filename[:sc[:crn]]
```

filename	The name of the file containing the main program.
sc	The main program file's security code (optional)
crn	The main program file's cartridge reference number (optional).

Terminate the program name list with the characters, /E.

Example of a SGPRP input directive list:

```
MAINP:CJ:32767
SEGM1:CJ:-15
SEGM2:CJ:-15
SEGM3:CJ:-15
/E
```

SGPRP Input, Interactive Mode

From an interactive device, the SGPRP program issues requests for input directives.

When SGPRP begins execution, the following messages are displayed:

```
     SGPRP STARTED
     "MAIN PROGRAM NAME?"
```

In response to the second message, respond with a main program name in the form:

```
     filename[:sc[:crn]]
```

where <filename[:sc[:crn]]> is exactly as defined for input from a file or non-interactive device (see Non-Interactive Mode above).

Next, SGPRP displays the following message:

```
     "/E OR SEGMENT NAME?"
```

For each segment to be prepared, enter its file name in the form:

```
     filename[:sc[:cr]]
```

where <filename[:sc[:cr]]> is exactly as defined for input from a file or non-interactive device.

Terminate the list by entering the characters, /E.

Following your entry of /E, SGPRP displays a message to indicate that it has completed execution:

```
     SGPRP DONE
```

SGPRP Error Reporting

Any error conditions encountered during execution of SGPRP result in the display of an error message. For interactive input, these error messages will be sent to the input device. For non-interactive input, the error messages will be sent to the system console (LU1).

The RTE-MIII Absolute Program Loader

The RTE-MIII Absolute Program Loader (APLDR) operates only within an RTE-MIII node. APLDR has been modified for DS/1000-IV so that you can load absolute programs into an RTE-MIII node from remote as well as local files. APLDR can be scheduled either in your local node using the standard RTE-MIII command "LO", or in a remote RTE-MIII node using the REMAT command, "LO". If remote REMAT "LO" commands are to be directed to the RTE-MIII node, then both EXECW and RFAM must be present and DS must be initialized. EXECW must reside in the RTE-MIII node and RFAM must reside in the node where the program file exists. (See DS/1000-IV Network Manager's Manual)

If the APLDR program is executed via the RTE-MIII "LO" command, then prior to performing the program load, APLDR displays the query:

```
LOAD FILE'S NODE?
```

In response, you enter the node number where the file resides.

To load programs from local files at an RTE-MIII node using this version of APLDR, the RFAM module must be present and enabled (via DINIT or DSMOD) because the program file is accessed with RFA calls. Because this method is used, you do not have to append the RTE-MIII file management subroutines to APLDR. If the RFAM module has not been scheduled via DINIT or DSMOD, programs cannot be loaded from local files, but can be loaded from local input devices. In this case, specify the logical unit number associated with the input device in your entry of the "LO" command (if not specified, the input device parameter defaults to LU 4).

RTE-MIII On-Line Loader Example

The following listing illustrates a possible answer file, *TEST1, for the loader/generator, RTMLG, to be used when loading a program, TEST1. In this example, *TEST1 is accessed as the loader command file using the following command:

```
RU,RTMLG,*T,ES,T1,,,,,SS
```

RTMLG then uses the following file to load the program TEST1:

```
LOAD
TR,SNAP20::DS
ECHO ON 'TEST1::SP
MAP MODULES ON MTEST1::SP
LINKS IN CURRENT
OUTPUT ON TEST1::SP
REL %TEST1::SP
SEARCH $DSLB3::MM
SEARCH $DSLB2::MM
SEARCH $DSLB1::MM
SEARCH $DSNMA::MM
SEARCH $DSRR ::MM
SEARCH $DSLSM::MM
SEARCH $DSMXL::MM
SEARCH %FF4.N::SM
SEARCH %MSYLB::23456
SEARCH %FF4.N::SM
SEARCH %RLIB1::SM
SEARCH %RLIB2::SM
SEARCH %RLIB3::SM
END
END
END
```

The libraries searched in this example are not an inclusive list of libraries that may be necessary to load your program successfully.

Once the program has successfully loaded, it may then be down-loaded into the RTE-MIII system using the REMAT "LO" command:

```
:RU,REMAT
$SW,,4,DS
#LO,TEST1,,2
```

RTE-L On-Line Loader

The steps taken to load programs into an RTE-L/XL are similar to those taken in an RTE-MIII except that segmented programs are prepared by the RTE-L/XL loader and the program SGPRP is not needed.

The following steps may be taken using either local or remote files:

1. Run the RTE-L/XL version of LOADR to prepare relocatable modules for subsequent loading into an RTE-L/XL node by converting relocatable code to absolute. Since the RTE-L/XL version of LOADR has the same name as the RTE-IVB's, IVE's and 6/VM's LOADR, the RTE-L/XL version will most likely be renamed to something else when running in an RTE-IVB or 6/VM node. See your Network Manager to determine the name of the RTE-L/XL LOADR located in the RTE-IVB, IVE or 6/VM node.

   For more information on the RTE-L/XL version of LOADR, see the appropriate Loader Reference Manual.

2. Depending on which version of the RTE-L/XL absolute program loader you have installed, perform one of the following steps:

   a) If APLDR-L (which is designed for memory-based RTE-XL/A systems) exists in your system, either issue the "LO" command locally or in a remote RTE node, use the REMAT "LO" command.

   b) If APLDX exists (the mini-APLDR, designed for unmapped RTE-Ls) in your system, run the program RPRTL from a hard-disc based node.

3. If your program is segmented, be sure your program is coded to include the call SEGLD. This call is used when the main program has completed, and a segment needs to be located, loaded, and executed from a disc file.

   If the segmented program is to reside on disc at the local system, you must relocate SEGLD from the RTE-L/XL system library, $SYSLB. If the segmented program is to be loaded into memory over a DS communication link you must relocate %SGXL for RTE-A/XL, or %SGLDL for RTE-L, from the DS/1000-IV modules before searching system libraries.

RTE-L Loader Considerations

When running the RTE-L/XL version of LOADR, the same "LO" and "BL" commands must be added to the loader answer file, but you must determine these values from the generation listing created when the RTE-L/XL was generated.

Any values will do, provided that the program falls completely within an area of memory not being used by any other program which will be in memory at the same time (see next section on Loading Restrictions). You may actually run more programs than will fit in 32K of memory, provided you first delete one or more programs which would conflict before loading another program into memory.

To take a simple example, suppose a memory-based RTE-L/XL is generated, with DINIT starting at location 60000 (octal), base page at 650 (octal). You can relocate any number of programs by placing in the RTE-L/XL loader answer file the commands:

```
BL,650B
LO,60000B
```

Of course, no two such programs may be in memory at the same time. If you need several of them in memory at the same time, you will have to determine where LO and BL need to be set for each program from the load map of the previous program.

Loading Restrictions

Before issuing the REMAT "LO" command or running RPRTL to "down-load" an absolute program into an RTE-L/XL, a few restrictions need to be recognized. Any combination of programs may be in memory at any one time, subject to the following restrictions:

- APLDR-L or the mini version, APLDX, will not accept a request to load any program into memory which would overlay any part of any other program in memory. This includes main memory, as well as the base page area. To use an area of memory formerly occupied by a program, the command "OF, program name, ID" must first be given to remove the program causing the conflict before REMAT's "LO" command or RPRTL can be used to put a new program in that area.

- The memory formerly occupied by such a purged program may be used by one or more programs, but a sufficient number of ID segments must have been allocated during the system generation phase for all the programs which may exist in memory at any one time.

- The file loaded must have been created by the RTE-L/XL loader for the particular RTE-L/XL system being used, using the snapshot created when the RTE-L/XL system was generated.

- If two masters should attempt to load programs into the same RTE-L/XL node at the same time, the first request to reach the RTE-L/XL will result in that area of memory being "reserved" for that program. If the second program does not conflict with this reservation, both loads can proceed. However, if there is a conflict, the second user will receive a program conflict error, and will be told to remove the first program. There of course must be some coordination between the two users whose programs are in conflict, i.e., the second user should not summarily remove the first program, without first assuring that no one else will need it.

- A DS/1000-IV program cannot be loaded as a type 6 program. Also, you cannot call a DS/1000-IV subroutine from a type 6 program.

Additional Loading Restrictions for APLDX

If the program is relocated in the background area, then it will only be accepted if the background area is empty (no current ID segment for any background program) and the load/swap module is not generated in.

Only one segmented program may be in the real time area at a time.

The RTE-L Mini-APLDR (APLDX)

The RTE-L Absolute Program Mini-Loader is an alternate version of APLDR-L intended for use in unmapped RTE-L nodes only. It is NOT RECOMMENDED for RTE-XL or A. The sole purpose of this loader is to provide more user-available memory space in an RTE-L which is a DS/1000-IV node. This is done by allowing that area of memory occupied by programs which are not currently needed, to be overlaid by other programs. For example, the DS/1000-IV initialization program DINIT is not required after it has initialized DS. If PASS (the "start-up program") is used, it also is not required once the "start-up" program has completed.

This capability extends to user programs also and thus greatly extends the usefulness of an RTE-L node by allowing a remote, disc-based node to be the storage location of a number of program files which may be copied into an RTE-L's memory as required.

The mini-version of APLDR-L provides additional user address space. However, some APLDR-L features are lost:

REMAT "LO" command

"LO" command entered locally at the RTE-L

FLOAD subroutine

"IO" command entered remotely	(This command can only be entered locally, and then only if the RTE-L version of program COMND is used.)
"PL" command entered remotely	(This command can only be entered locally, and then only if the RTE-L version of program COMND is used.)

RPRTL

To initiate a "load" of a program into an operating RTE-L node (as opposed to "cold" loading, where the entire operating system, and real time programs are loaded) which has the "mini-APLDR" installed, run the program RPRTL.

RPRTL may be run by an operator, or scheduled by a program. If run by an operator, it is desirable that the "error printout LU" parameter be specified in the run string so that, if an error occurs, the operator can be notified (if no LU is specified, no printout occurs). If scheduled by a program, this parameter is optional; if an error occurs, this information is always passed back to the "father" program, if one exists.

To schedule RPRTL, use the following command sequence:

```
:RU,RPRTL,[error printout LU],remote node,program file <namr>
```

error printout	LU (optional) used to print errors. If not specified or zero, errors are not printed. If this parameter is not supplied, the program assumes it is being scheduled by another program and errors are then returned to the "father". For an explanation of the errors, see the error appendix.
remote node	RTE-L system node number, into which the program file specified is to be loaded.
program file <namr>	FMP <namr>, including name, optional security code and cartridge reference number of the program file. This file must be located in the same node as this program. The first five characters of this file name become the program name in the RTE-L. This allows easy program re-naming.

L-Series On-Line Loader Example

The following listing shows an example of an answer file used with the L-Series loader:

```
OUTPUT,LDEMO  <---------Executable Binary File (Type 6)
SNAP,SNAPL    <---------System Symbol Table Snapshot File
LCOM          <---------Labelled Common desired
REL,%LDEMO::DS
END
```

DOWN-LOADING AN OPERATING SYSTEM

Before loading the absolute binary version of this program into an L-Series, you may optionally want to down-load a different Operating System into the L-Series. For more information on down-loading, see Chapter 4 of the DS/1000-IV Network Manager's Manual, Vol. II.

If you do not wish to change your operating system, skip to DOWN-LOADING A PROGRAM.

To down-load an operating system, you might use a sequence of commands similar to the following:

```
:RU,DSVCP,3
```

Run DSVCP, the Remote Virtual Control Panel and enter the node number of the RTE-L/XL.

```
/DSVCP: NODE 3 AND LU 13

/DSVCP:   \B
PASSWORD?: DS
```

Break into front panel mode.

The following listing is the contents of the memory registers on the RTE-L/XL when the break mode was issued. The command, %BDS7 is a boot-up command. In this example, file P00007 is the absolute binary operating system file which is sent over the DS link to to RTE-L/XL system.

```
P 003046 A 000000 B 000000 M 003045 T 100020 /DSVCP: %BDS7
```

`S=26 COMMAND ?BR`	Since the DS link is initialized during boot-up, the RTE-L/XL system has no idea the program DSVCP was communicating with it. For this reason, the system from which DSVCP was scheduled is waiting for a reply from the RTE-L/XL which it will never get. The solution is to enter a break command so that the DSVCP prompt will return.
`/DSVCP:  \E`	Terminate DSVCP.
`/DSVCP: END`	

```
***********
* WARNING *
***********
```

The RTE-L/XL node will halt all activity when the BREAK command is sent to it. (It must also be set for remote virtual control panel and enabled for forced cold load.) You must be sure that nothing critical is in progress before taking this action.

DOWN-LOADING A PROGRAM

When the desired Operating System is loaded, the absolute binary version of the program, LDEMO, may be down-loaded into the RTE-L/XL. To do this, you could use a sequence of commands similar to the following:

```
:RU,REMAT                            Run REMAT.

$SW,3,,DS                            SWitch to the RTE-L/XL system.

#OF,DINIT,ID                         In this example, DINIT was removed
                                     to provide more room in the real
                                     time partition (since DINIT is no
                                     longer needed at this point).

#LO,LDEMO                            Load LDEMO.

#EX                                  End REMAT.

 $END REMAT
```

For down-loading a program with APLDX, enter the above set of commands except - LO,LDEMO. Then, after ending REMAT, type:

```
:RU,RPRTL,1,3,LDEMO                  Run RPRTL to down-load the
                                     program, LDEMO, into the RTE-L/XL
                                     system located at node 3. Error
                                     messages are logged to LU 1.

PROGRAM < LDEMO> DOWN-LOAD COMPLETE
```

Once the program has loaded successfully into the RTE-L/XL, you may run the program locally at the RTE-L/XL or remotely from REMAT:

```
:RU,REMAT

$SW,3,,DS

#RW,LDEMO

#EX                                  Once your program has finished
                                     executing, terminate REMAT.   $END
REMAT

:                                    Return to RTE processing.
```

RTE-A APLDR

APLDR in RTE-A is similar to APLDR in RTE-L/XL. The subroutine %$MWB1 must be relocated in the system area during system generation. The REMAT "LO" command and the utility FLOAD are only available in memory-based systems. Another requirement is that the system contains only reserved partitions. APLDR does not work with the dynamic memory partition. The following enhancements have been added to APLDR for RTE-A:

1) Loads sharable EMA programs. This requires that the sharable label be set up. This is usually done with the program BOOTX or BUILD.

2) A maximum of fifteen sharable EMA labels can be set up.

3) Each sharable area can be shared by sixty-three different programs (see RTE-A Programmer's Manual for additional information on shared EMA).

4) APLDR requires that a big enough free partition is available to load the desired program over the DS links.

5) APLDR will not load a program into a partition in which a sharable EMA program has been assigned. If the sharable EMA program is not running, such a partition appears to be free when using the PL,PT command, but in reality, it is not truly free; therefore APLDR will not load a program into such a partition.

SEGLD — Segmented Program Loading

Large programs can be divided into a main program and one or more segments to save memory space. When you schedule the main program for execution, it is loaded and executed exactly like any other program. When execution of the main program is completed, you issue a call to SEGLD to locate, load, and execute the segment from a disc file. At the end of the segment's execution, you may terminate the program or again issue a call to SEGLD to load another segment. When the second (and any subsequent) segment is loaded it will overlay the previous segment in memory. If you have adequate memory space, the main program will not be overlaid. If the main program is not overlayed, you can return to it via a JMP (or GO TO <label>) instruction where the instruction label is an entry point within the main program.

Although a call to SEGLD can only be issued within an RTE-MIII, RTE-IVE or memory-based RTE-L, XL or A node, the file containing the program segment may exist in any hard-disc based system. In Session Monitor nodes, the file must exist on a system disc. This is because SEGLD can not log on to accounts.

If the program is real-time and segmented, and the DS version of SEGLD is used to load the segments, then the program must be the last one in the real-time area. The reason for this is if there are segments, the memory-conflict check is made from the start of segmented program to the end of the real-time area.

CALLING SEQUENCE:

For RTE-MIII/IVE:

```
CALL SEGLD(NAME,IERR[,IP1[,IP2[,IP3[,IP4[,IP5[,NODE]]]]]]
```

For RTE-L/XL/A:

```
CALL SEGLD(NAME,IERR[,IP1[,IP2[,IP3[,IP4[,IP5]]]]]
```

NAME	A 3-word array that contains the ASCII-coded name of the program segment file to be loaded.
IERR	Error return; a variable to which an integer value representing an error code is returned if an error condition is encountered during execution of this call.
IP1-IP5	Up to five integer values that may be passed from the calling program to the called segment (optional).
NODE	The node number or the negative logical unit number of the communication link where the program segment file specified in the NAME parameter resides. Specification of the NODE parameter is optional. If omitted, its value defaults to -1 (local node). In RTE-L/XL/A, the node number is assumed to be the same as the location of the main since the main and all segments are in the same file.

Chapter 9
HP 3000 Intrinsics

HP 3000 Remote File Access (RFA) Intrinsics

You can issue calls from your RTE application program to a set of compatible HP 3000 RFA intrinsics. These intrinsics provide file access at an HP 3000 node. Only a brief description of the HP 3000 RFA intrinsics together with the DS/1000-IV syntax used to call the HP 3000 intrinsics is shown in the following paragraphs. Refer to detailed descriptions of the parameters passed in the calling sequences in the HP 3000 MPE Intrinsics Reference Manual. Your calling program must reside at an RTE node that is directly linked to the HP 3000 node.

Unlike the equivalent HP 3000 intrinsics, the following calls have parameters of integer and integer array type only.

NOTE: The HP 3000 convention for specifying computer word bits is the reverse of the HP 1000 convention. In addition, the internal representation of floating point data differs. Refer to the appropriate HP 1000 and HP 3000 manuals for a description of the floating point format.

HP 3000 File Intrinsic Condition Codes

The HP 3000 File System intrinsics return condition codes to your program to indicate the result of the operation. These condition codes can be retrieved for examination via a call to a function named ICC. The function ICC returns the following condition code values:

ICC Value	HP 3000 Condition Code
-1	CCL
0	CCE
+1	CCG

The following is an example of a call to ICC both in FORTRAN and HP Assembler languages:

```
   FORTRAN                               |   HP Assembler
-----------------------------------------|-------------------------------
       IF (ICC(M)) 10,20,30              |        JSB ICC
 10    <condition code CCL>              |        DEF *+1
         .                               |        SSA
         .                               |        JMP LBL10
         .                               |        SZA,RSS
 20    <condition code CCE>              |        JMP LBL20
         .                               |        <condition code CCG>
         .                               |          .
         .                               |          .
 30    <condition code CCG>              |          .
         .                               | LBL10  <condition code CCL>
         .                               |          .
         .                               |          .
         .                               |          .
                                         | LBL20  <condition code CCE>
                                         |          .
Because of FORTRAN syntax                |          .
conventions, you must include            |          .
a dummy parameter within a call          |          .
to ICC from a FORTRAN program.           |
                                         |
-------------------------------------------------------------------------
```

HP 3000 Intrinsic Call Requirements

Program communication with an HP 3000 node is accomplished following the establishment of a session at the HP 3000 node. A session is established by execution of the HELLO command or call. HELLO can be issued by a program, or by the program that scheduled it (its "father"). The program that issues a HELLO should also issue a BYE before terminating execution. If a program is scheduled by a father program, it need not call BYE because the father program will. (If the father program is RMOTE, the HELLO and BYE commands can be used to create and terminate an HP 3000 session.)

One program (the "father") can establish a session at an HP 3000 and then schedule another program (the "son") to share the session. The son program may then establish contact with the session created by the father via a call to PRCNM. (PRCNM is described in Chapter 7, "DS/1000 Utility and Program Calls.")

If RMOTE is the father program, a HELLO command can be issued to create the session and the son program can be scheduled by using the ON, RW or RU command. RMOTE will schedule the son program with wait. If you are scheduling the son program from RMOTE, do not use the NOW option. If the father program is RMOTE or another program, do not use the fifth scheduling parameter.

The following example shows a son program called "PROG" being scheduled from RMOTE.

```
:RU,RMOTE
$SW
#HELLO <user.accountname>
#SW
$RW,PROG                   PROG is scheduled with wait.
$SW                        Issued when prompt returns after PROG
                           terminates.
#BYE
```

A program can determine whether or not it was scheduled by a father program by calling RMPAR to check the value of the fifth scheduling parameter. If the value is zero or positive, the program was not scheduled by a father and must issue its own HELLO and BYE calls. If the value is negative, the program was scheduled by a father and must call PRCNM to establish communication with the HP 3000 session created by the father.

A sample program using HELLO, BYE and PRCNM is included in Chapter 7, "DS/1000-IV Utility and Program Calls."

Many of the parameters used in the HP 3000 File System calls are optional. In FORTRAN, unlike SPL/3000, you cannot default an optional parameter imbedded in the parameter string by omitting it. Instead, you can fill in the parameter using a dummy variable. For parameters that do not return a value, use the constant zero. For parameters that return a value, use a dummy variable or a dummy array large enough to contain the returned values.

NOTE

FORTRAN requires a dummy variable name (do not specify a constant) for any non-optional but undesired return parameter. Be sure that the dummy variable name specified has adequate dimensions for the value to be returned.

Those intrinsics that return functional values (FOPEN, FREAD, and FRLAT) must be declared as INTEGER in FORTRAN programs.

FCHEK

This intrinsic obtains file I/O error information. It can be used when a call to a file intrinsic returns a condition code value that indicates an I/O error. FCHEK is equivalent to the FS/3000 intrinsic FCHECK.

CALLING SEQUENCE

```
      CALL FCHEK(filenum,errorcode,tlog,blknum,numrecs)
```

FCLOS

This intrinsic terminates access to a file by your program. When FCLOS is executed, buffers and control blocks through which you accessed the file are deleted and the device on which the file resides is deallocated. If you do not issue a call to the FCLOS intrinsic for each file opened during your session, MPE/3000 will issue FCLOS calls automatically when your session is terminated. FCLOS is equivalent to the FS/3000 intrinsic FCLOSE.

CALLING SEQUENCE

```
      CALL FCLOS(filenum,disposition,seccode)
```

FCNTL

The FCNTL intrinsic performs specified control operations on a file or device. FCNTL is equivalent to the FS/3000 intrinsic FCONTROL.

CALLING SEQUENCE

```
      CALL FCNTL(filenum,controlcode,param)
```

FINFO

This intrinsic obtains file access and status information. Once a file is opened, a call to FINFO can be issued to obtain this information. FINFO is equivalent to the FS/3000 intrinsic FGETINFO.

CALLING SEQUENCE

```
     CALL FINFO(filenum,filenam,foptions,aoptions,recsize,devtype,
               ldnum,hdaddr,filecode,recpt,eof,flimit,logcount,
               physcount,blksize,extsize,numextents,userlabels,
               creatorid,labaddr)
```

FLOCK

This intrinsic dynamically locks a file for exclusive access by your program. FLOCK is equivalent to the FS/3000 intrinsic FLOCK.

CALLING SEQUENCE

```
     CALL FLOCK(filenum,lockcond)
```

FOPEN

The FOPEN intrinsic opens an HP 3000 file for access by your program. When FOPEN is executed, a file number ("filenum") is returned. This file number must be used by your program in all subsequent file references in order to access the proper file. FOPEN is equivalent to the FS/3000 intrinsic FOPEN. You must explicitly declare FOPEN as an integer function in your FORTRAN program.

CALLING SEQUENCE

```
    INTEGER FOPEN
    filenum=FOPEN(formaldesignator,foptions,aoptions,recsize,
                  device,formmsg,userlabels,blockfactor,
                  numbuffers,filesize,numextents,initialloc,
                  filecode)
```

FPOIN

The FPOIN intrinsic sets the logical record pointer to any record within a disc file. The file must have only fixed-length records. FPOIN is equivalent to the FS/3000 intrinsic FPOINT.

CALLING SEQUENCE

```
      CALL FPOIN(filenum,recnum)
```

FRDIR

The FRDIR intrinsic performs a read operation on a specified logical record from a disc file to the user's buffer. The file must have only fixed-length or undefined-length records. FRDIR is equivalent to the FS/3000 intrinsic FREADDIR.

CALLING SEQUENCE

```
      CALL FRDIR(filenum,target,tcount,recnum)
```

FRDSK

The FRDSK intrinsic can be used to seek out and perform a transfer of a specific logical record from a disk file to a buffer prior to a call to the FRDIR intrinsic. The file referenced must allow I/O buffering and have fixed-length or undefined-length records. FRDSK is equivalent to the FS/3000 intrinsic FREADSEEK.

CALLING SEQUENCE

```
      CALL FRDSK(filenum,recnum)
```

FREAD

The FREAD intrinsic performs a read operation of a logical record from a file on any device to the user's buffer. The record read is determined by the current position of the logical record pointer. This intrinsic returns an integer value representing the number of words or bytes read. FREAD is equivalent to the FS/3000 intrinsic FREAD. You must explicitly declare FREAD as an integer function in your FORTRAN program.

CALLING SEQUENCE

```
      INTEGER FREAD
      lgth=FREAD(filenum,target,tcount)
```

FRLAB

The FRLAB intrinsic performs a read operation on your disc file label. FRLAB is equivalent to the FS/3000 intrinsic FREADLABEL.

CALLING SEQUENCE

```
      CALL FRLAB(filenum,target,tcount,labelid)
```

FRLAT

This intrinsic is used to obtain information about the interactive/duplicative attributes of a specified file pair. A value is returned that represents the current attributes of the files. FRLAT is equivalent to the FS/3000 intrinsic FRELATE. You must explicitly declare FRLAT as an integer function in your FORTRAN program.

CALLING SEQUENCE

```
      INTEGER FRLAT
      intordup=FRLAT(infilenum,listfilenum)
```

FRNAM

This intrinsic renames a disc file. FRNAM is equivalent to the FS/3000 intrinsic FRENAME.

CALLING SEQUENCE

```
      CALL FRNAM(filenum,newfilereference)
```

FSPAC

This intrinsic performs forward or backward spacing over a specified number of logical records on a disc file, or physical records on a magnetic tape file. The file must have fixed-length or undefined-length records. FSPAC is equivalent to the FS/3000 intrinsic FSPACE.

CALLING SEQUENCE

```
      CALL FSPAC(filenum,displacement)
```

FSTMD

The FSTMD call sets or resets file access modes such as automatic error recovery, critical output verification, and user terminal control. Any file access mode set by a call to FSTMD remains in effect until either reset by another call to FSTMD or until the file is closed. FSTMD is equivalent to the FS/3000 intrinsic FSETMODE.

CALLING SEQUENCE

```
      CALL FSTMD(filenum,modeflags)
```

FUNLK

The FUNLK intrinsic dynamically unlocks a file that was previously locked via a call to FLOCK. FUNLK is equivalent to the FS/3000 intrinsic FUNLOCK.

CALLING SEQUENCE

```
      CALL FUNLK(filenum)
```

FUPDT

This intrinsic performs a write update operation of a logical record to a disc file. FUPDT is equivalent to the FS/3000 intrinsic FUPDATE.

CALLING SEQUENCE

```
      CALL FUPDT(filenum,target,tcount)
```

FWDIR

The FWDIR intrinsic performs a write operation of a specified logical record to a disc file from a user's buffer. The file must have only fixed-length or undefined-length records. FWDIR is equivalent to the FS/3000 intrinsic FWRITEDIR.

CALLING SEQUENCE

```
      CALL FWDIR(filenum,target,tcount,recnum)
```

FWLAB

This intrinsic performs a write operation of your label to a disc file. FWLAB overwrites any existing label. FWLAB is equivalent to the FS/3000 intrinsic FWRITELABEL.

CALLING SEQUENCE

```
      CALL FWLAB(filenum,target,tcount,labelid)
```

FWRIT

The FWRIT intrinsic performs a write operation of a logical record from a user's buffer to a file on any device. Following FWRIT execution, the logical record pointer is set to the record immediately following the record written. FWRIT is equivalent to the FS/3000 intrinsic FWRITE.

CALLING SEQUENCE

```
      CALL FWRIT(filenum,target,tcount,control)
```

The following table shows the relationship of DS/1000-IV calls to FS/3000 file system calls.

Table 9-1. DS/1000-IV - FS/3000 Call Equivalence.

DS/1000-IV NAME	FS/3000 NAME
FCHEK	FCHECK
FCLOS	FCLOSE
FCNTL	FCONTROL
FINFO	FGETINFO
FLOCK	FLOCK
FOPEN	FOPEN
FPOIN	FPOINT
FRDIR	FREADDIR
FRDSK	FREADSEEK
FREAD	FREAD
FRLAB	FREADLABE
FRLAT	FRELATE
FRNAM	FRENAME
FSPAC	FSPACE
FSTMD	FSETMODE
FUNLK	FUNLOCK
FUPDT	FUPDATE
FWDIR	FWRITEDIR
FWLAB	FWRITELABEL
FWRIT	FWRITE

Appendix A HP Character Set

Effect of Control key *

					000-037B		040-077B		100-137B		140-177B	
b_7 b_6 b_5					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
BITS				COLUMN →	0	1	2	3	4	5	6	7
b_4	b_3	b_2	b_1	ROW ↓								
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	¦
1	1	0	1	13	CR	GS	—	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

32 CONTROL CODES

Upshifted Lower Case

← 64 CHARACTER SET →|

← 96 CHARACTER SET →

← 128 CHARACTER SET →

EXAMPLE: The representation for the character "K" (column 4, row 11) is.

```
         b7 b6 b5 b4 b3 b2 b1
BINARY   1  0  0  1  0  1  1
OCTAL    1     1        3
```

* Depressing the Control key while typing an upper case letter produces the corresponding control code on most terminals. For example, Control-H is a backspace.

9206- 1A

Appendix B Error Messages

This error section is divided into 3 major areas:

The first group of errors consist of numeric error codes and are organized in numeric order. These errors may be generated by REMAT, RMOTE or by one of the various DS/1000-IV programmatic calls. Numeric error codes may be returned to the 'IERR' parameter or by making a call to DSERR.

The second group of errors consist of alphanumeric error messages. These errors are organized in alphabetical order and consist of errors generated from numerous possibilities.

The last group of errors are organized by the specific program generating the error. Many of these errors are generated by programs described in the Network Manager's Manual but are listed here to aid in centralizing all DS/1000-IV errors. References are made where applicable.

```
********
* NOTE *
********
```

User application programs written for DS/1000 may need to be modified in order to obtain the error message qualifier available with DS/1000-IV. Otherwise, they will not be able to distinquish between the various meanings of the same error code. DSERR cannot report errors from DS/1000-to DS/3000 RFA calls.

Numeric Error Codes

HELLO/BYE Error Codes

The Hello/Bye Error Codes are HP 3000 error codes and messages reported on the RTE side.

CODE	DESCRIPTION
0	No error
1	HELLO failure or line disconnected. Recovery: try again or report error to the Network Manager.
4	Invalid LU: The LU specified is not a communications line to an HP3000, or the DS/3000 modules are not initialized. Recovery: check parameters and try again.
5	Timeout
6	Illegal (rejected) request: Request is not recognized. Recovery: check syntax and try again.
7	Transaction Table access error (not enough Transaction Control Blocks for HELLO. See DSINF and DSMOD.)
8	Non-DS error (e.g., input-only device specified as list LU)

IERR Error Codes

Numeric error codes may be returned to user PTOP, RFA and utility calls which have the "IERR" parameter. In addition, numeric error codes may be generated by REMAT and DEXEC/EXECM. REMAT displays all numeric error codes received on the operator's console using the DSERR routine provided with DS/1000-IV.

Since negative error codes may be ambiguous and the same numeric errors may have different meanings it is suggested that the user call DSERR when an error occurs in a program. The format for the DSERR buffer is as follows:

```
     DS ERROR: SSEE(QQ), REPORTING NODE NNNNN
```

where

```
     SS = subsystem prefix (e.g. SC, IO, FM for FMP, DS, etc.)

     EE = error code

     QQ = error code qualifier

  NNNNN = node reporting the error
```

If the "no-abort" option is used and an error occurs, ASCII error codes are returned in the A- and B- registers of user programs calling DEXEC or certain utilities (DMESS, etc.). Otherwise, the message is printed on the initiating console and the user program is aborted. DS/1000 system programs receiving these error codes generally display them on the operator's console or log device.

REMAT calls the error utility DSERR to print error data returned in reply messages. Several categories of symbolic error messages are returned including FM-XX codes for File Manager errors, SM-XX for Session Monitor errors, RS-XX for Remote Session Monitor errors, and of course the user may see the IOXX and SCXX errors from RTE itself.

CODE	DESCRIPTION
56	Illegal parameter type
55	Missing parameter
45	Input device must be interactive for REMAT 'FL' command
13	REMAT transfer file stack overflow
10	Input error (illegal REMAT command, etc.)
1	a) (FCOPY) Duplicate file name: the first two characters have been replaced by ".." (warning only) or b) (PTOP) REJCT received
0	a) No error or b) (PTOP) ACEPT received
-1	# a) (RFA) Disc down or b) Non-existent program (DEXEC 99, Program Status Request)
-2 #	Duplicate file name.
-3 #	Illegal backspace.
-4 #	File too long or record size error.
-5 #	Attempt to read or position a record not written or (on update) to write an illegal record length.
-6	# a) File not found or b) On DCRET or FCOPY the specified file size is negative. The remote file create subroutine does not allow a negative file size, because a communication line error may occur, which would leave all available disc space on the referenced disc allocated to the created file.
-7 #	Invalid security code.
-8	File currently open exclusively or open to more than 7 programs.

CODE	DESCRIPTION
-9 #	Attempt to open type 0 file as type 1 or to use DPOSN on type 0 file.
-10 #	Not enough or error in parameters.
-11	DCB not open (on FLUSH command) or attempt to replace a memory-resident program (remote-loading).
-12 #	EOF or SOF read.
-13 #	Cartridge locked.
-14 #	Directory full.
-15 #	Illegal file name.
-16 #	Illegal type or size = 0.
-17 #	Illegal read/write on type 0 file.
-18 *+	Destination node does not have FMP. Note: RTE-MIII file systems assign a different meaning to this error code.
-25 +	Bad FCODE (internal RFAM error) or an extended RFA call was sent to a 91740 node.
-26 +	Bad entry number in RFAM. Indicates that the file was never opened correctly, or the pseudo-DCB in the user program has been destroyed, or the specified pseudo-DCB is incorrect due to a programming error, or the DCB for this file access call was released when the same file was opened to another DCB within the program.
-28 *	No internal table space in RFAM. An attempt was made to open more files than the version of RFAM used at the destination node can handle. If the single-DCB version of RFAM is used, the only possible recovery is to wait for the file to be closed or use the multiple-DCB version. If the multiple-DCB version of RFAM is used, more files can be opened if the system is re-initialized and, when DSMOD asks for the number of files which can be opened simultaneously, answer with a larger number.
-29 *	Internal RFAM tables invalid. Contact your HP field office and report this.

CODE	DESCRIPTION
-32 #	Cartridge not found. (RTE-IVB FMP)
-33 +	On DCRET, the specified file size is negative. The remote file create subroutine does not allow a negative file size, because a communication line error may occur, which would leave all available disc space on the referenced disc allocated to the created file.
-100	"Break" was entered during an FCOPY file transfer.
-999	Non-DS error. May be an IO or RN error. (Call DSERR for more information about this error.)

* These error codes have a different meaning in RTE-MIII (local) file access calls than DS/1000-IV remote file access calls. To determine the proper meaning, determine whether the call was a local file access in RTE-MIII (OPEN, READF, etc.) or a RFA call (DOPEN, DREAD, etc.).

+ These error codes have a different meaning in RTE-IVB. If a call to DSERR shows "FM-xx" instead of "DS-xx", refer to the RTE-IVB Programmer's Reference Manual, FMP error codes.

These error codes are reported by the file system at the reporting node. Refer to the appropriate Operating System Manual to determine the meaning.

Remote I/O Mapping Error Codes

0	No error.
4	DS/1000-IV quiescent resource number is corrupt.
5	Source node is quiesced.
6	Class number set up for LUMAP has been corrupted.
8	Program LUMAP is not present in system.
9	Destination node number is not in the NRV.
10	DS/1000-IV is not initialized.
11	Error on class number allocation.
12	No class numbers available.

DLGON/DLGOF/DLGNS Error Codes

See the ASCII equivalents for more information.

13	Shutdown. (Same as SM13.)
12	Account file corrupt. (Same as SM12.)
11	FMP error XXXX on disc mount attempt. (Same as SM11.)
10	No free ID segments or FMGR not found. (Same as SM10.)
9	SST overflow. (Same as SM09.)
8	Duplicate session identifier. (Same as SM08.)
7	No room for session control block. (Same as SM07).
6	Conflict in definition of session LU XX. (Same as SM06.)
5	Illegal access. (Same as SM05.)
4	No such user. (Same as SM04.)
3	Session limit exceeded. (Same as SM03.)
2	Required class number not available. (Same as SM02.)
1	FMP error -xxxxx on Account File access. (Same as SM01.)
-1	Remote session was lost or ATACH failed. (Same as RS01.)
-2	Class I/O error on response from LOGON. (Same as RS02.)
-3	Limit exceeded on # of remote nodes. (Same as RS03.)
-4	Remote session environment not initialized. (Same as RS04.)
-5	Wrong password for non-session access. (Same as RS05.)
-6	Another program owns (or has a) session. (Same as RS06.)
-7	Log-on or log-off to local node. (Same as RS07.)
-50	(Same as DS00)
-51	(Same as DS01)

-52	(Same as DS02)
-53	(Same as DS03)
-54	(Same as DS04)
-55	(Same as DS05)
-56	(Same as DS06)
-57	(Same as DS07)
-58	(Same as DS08)
-59	(Same as DS09)

PTOP Error Codes

The following errors are returned to either the master or slave program depending on the error. If the error is returned to the slave, it can pass information to the master in the tag field. However, the slave cannot initiate recovery procedures.

Error codes -50 through -59 are returned in lieu of "DS" errors 00 through 09, respectively, in P-to-P master calls. This simplifies the user's code when error-recovery is desired. All other messages (IOxx, SCxx, etc) are converted to -47 in P-to-P slave calls.

CODE	DESCRIPTION
-40	Not enough parameters
-41	Remote program not defined. If a clone of the slave is requested in the POPEN call, this error can occur if the type 6 file (if applicable) could not be opened, or there are no blank ID segments, or there are already 26 clones (.A thru .Z) for this program name.
-42	No remote system room to initiate communication. There is room for up to 20 P-to-P slave programs to be active in a single node at the same time. May also occur if no class number is available for sending data to the slave.
-44	Remote program not open correctly (PCB was destroyed). Indicates a programming error. May occur if a REMAT SO (slave off) command has been given or PTOPM was aborted and restarted. POPEN call gets this error if the slave program is not dormant and has not been scheduled by PTOPM in the slave node. This may be due to the slave program being scheduled by other means than POPEN.
-45	A PWRIT, PREAD or PCONT call has been issued to a slave program which is dormant. This indicates the slave program has been abnormally terminated, i.e. not by PCLOS or FINIS calls or not by SO command. A user should fix the code in the slave program which causes it to be aborted or should re-issue another POPEN call in the master program.

CODE	DESCRIPTION
-46	Sequence error. Slave programs must call GET routine first, then either ACEPT or REJCT, then call GET again. An exception to this is that, after an error is returned on the GET call, the slave goes back to the "GET" call again. The master program must be the one to attempt any retries. This error code indicates a programming error.
-47	May occur in P-to-P slave subroutine calls when a communications line error, system table validity check failure, request timeout error occurs, or GRPM's class is found to be bad. May occur if the NRV does not provide a path back to the master program (correct the NRV and re-initialize the node). May occur if a request's Transaction Control Block does not exist when the slave calls ACEPT or RJECT. All RTE-reported error messages (IOxx, RNxx, SCxx, etc.) are converted to -47. Since master programs are responsible for retries, it is suggested that the error code, program name, and some description of the transaction on which the error occurred be returned by the slave program. The master will receive a more descriptive error code and can determine the retry procedure, if any. The slave program should go back to the "GET" call when any error occurs.
-48	Abortive error: indicates something seriously wrong. May occur if a class number in PTOPM's table is found to be bad or 255 requests (the maximun allowed in class I/O) have occurred; this might occur if a monitor was locked out of execution for an extremely long time; for example, if swap tracks were unavailable or a program locked itself into the only available partition.
-49	A PCLOS terminated a shared slave program while one or more requests were pending from other master programs. This error will be returned to each master which had a request pending on the slave program when it was terminated.
-50	Local node not initialized or, for requests made to itself, local node is quiescent (same as DS00).

CODE	DESCRIPTION
-51	Communications line parity or other line error (same as DS01).
-52	Communications line time out error (same as DS02).
-53	Illegal record size (same as DS03).
-54	Illegal nodal address (same as DS04).
-55	Request time-out (same as DS05).
-56	Illegal request (same as DS06).
-57	System table error (same as DS07).
-58	Remote busy (same as DS08).
-59	Illegal or missing parameters (same as DS09).
-103	Illegal PCB (P-to-P)

APLDR Error Codes

CODE	ASCII APLDR	DESCRIPTION
56		LO command not available. To use LO, search $DSLDR before searching $CMDLB when loading APLDR (RTE-A)
23		Duplicate program name
19		Program is not linked for this system
14		No I.D. segments are available
0		Successful installation of I.D. segment into system
-15		Illegal name
-53		Partition doesn't exist
-54		Partition is too small
-56		Shared EMA partition doesn't exist
-57		Shared EMA partition is too small
-58		Reserved partition conflict
-59		Too many programs sharing EMA
-60	REM	Remove program to be overlaid
-61	DUP	Duplicate program name
-62	PTN	No free partition. (For RTE-L, background filled, or background load not allowed on this system.) No partition large enough (RTE-A)
-63	PTSZ	Partition specified not large enough or shared EMA label not found (RTE-A)
-64	NOID	No blank ID segments
-65	ID?	Identification records missing or wrong
-66	CKSM	Checksum error or file not loaded for current system

CODE	ASCII APLDR	DESCRIPTION
-67	COM	Common area overflow
-68	MEM	Memory overflow
-69	DISC	LOAD and SWAP are generated in. Both APLDR and LOAD and SWAP are trying to manage memory.

DS Alphanumeric Error Codes

Some of the errors listed below contain subsystem prefixes and qualifiers in addition to the error code (i.e., DS08(1)). Since only the error code is returned in the IERR parameter of DS/1000-IV programmatic calls, it is suggested whenever an error occurs in a program, that you make a call to DSERR so that the subsytem prefix, error code qualifier, and reporting node is returned in addition to the error code. DSERR is described at the beginning of this error appendix.

Miscellaneous Alphanumeric Error Codes

Other miscellaneous alphanumeric error codes can be found in alphabetical order throughout this section.

/A COMMAND READ!

> (DSLIN) /A was read as a request to terminate the program. DSLIN was terminated.

AUTO "BYE" FAILED

> (RMOTE) The "BYE" generated automatically when the EX command is entered with a HELLO outstanding has failed. May occur if the link has been disconnected since the HELLO was entered. This error is for the operator's information only; the session at the HP 3000 is closed automatically.

BAD LU

> (RMOTE) A negative LU number was specified in a MO command.

CANNOT LOCK LU xxx

> (DSLIN) DSLIN attempt to lock the lu for primary connection failed. Attempt to lock fails when someone else is running DSLIN at the same instant, or if you have incompatible software. Wait a minute and try again. If the problem persists, contact your network manager.

CONNECTING AS A SECONDARY STATION ON LU xxx

(DSLIN) HP 1000 is awaiting a call from the HP 3000 to establish the communciation link.

DINIT Error Codes

/DINIT: ANSWER YES OR NO

(DINIT) The question asked previously requires a "YES" or "NO" answer.

/DINIT: CANNOT HAVE MA TO LOCAL NODE

(DINIT) Message Accounting is only allowed (and only makes sense) between the local and remote nodes.

/DINIT: CLASS I/O ERROR

(DINIT) A required class number cannot be allocated. DINIT is aborted. This error may require re-generation with a larger allotment of class numbers.

/DINIT: COST ERROR!

(DINIT) Cost value assigned to a communication link is not within the valid range 1 to 99.

/DINIT: DINIT ABORTED

(DINIT) If initialization was in progress, all allocated resources have been returned to RTE.

/DINIT: DINIT MAY NOT BE CLONED-ABORTING!

(DINIT) Network initialization or shutdown was attempted with a cloned copy of DINIT (name other than DINIT). Retry with an un-cloned copy: "RU,DINIT: IH,..." for RTE-IVB or 6/VM, or "RP,DINIT" first, for RTE-L/XL/A.

/DINIT: END DINIT

(DINIT) Normal completion message. The ten characters comprising this message are also returned in the 5-word temporary storage area of the father's I.D. segment if scheduled by another program. They may be recovered through the use of RMPAR. If DINIT has been aborted, the five words of returned data consists of: 10000B,ER, D,IN,IT.

/DINIT: ERROR IN LINE #<nnnnn>

(DINIT) When an error occurs in DINIT's command file, the operator may either attempt to answer the pending question correctly, abort the startup, or shutdown and fix the file. If the last option is taken, DINIT reports the line number in the file where the error was detected.

/DINIT: ERROR: SC05 mmmmm

(DINIT) The specified monitor <mmmmm> is not in the system. If the monitor was requested by name, it may be loaded on-line later, and UPLIN schedules it automatically. If the message is printed in response to a "/D" command, then it is necessary to schedule the monitor with the DSMOD "/S" command.

/DINIT: ERROR: STAT: mmmmm

(DINIT) The status of monitor <mmmmm> is not 'dormant' and therefore it cannot be scheduled. Abort DINIT using the /A command and then use RTE operator commands to change the status.

/DINIT: FILE ERROR

(DINIT) Improper response to "INPUT # OF FILES". Retry. Also occurs in lieu of NODE SPEC. ERROR when input to DINIT is from a file (see also all occurences of that message).

/DINIT: INCOMPLETE INITIALIZATION DETECTED; CLEANING UP.

(DINIT) If DINIT is aborted during startup or shutdown, the next running of DINIT detects this, prints this message, and cleans up.

/DINIT: INCORRECT NUMBER OF MA NODES!

(DINIT) The number of nodes specified in the DINIT question 'NO. of MA NODES? does not match the number of declared Message Accounting nodes.

/DINIT: INVALID MA SPECIFICATION!

(DINIT) The sixth parameter of the NRV entry was input but was not equal to "MA".

/DINIT: INVALID MA RETRY LIMIT! EXCEEDS RANGE 1-15

(DINIT) The Message Accounting retry limit must be numeric and in the range 1 to 15.

/DINIT: INVALID NAME!

(DINIT) Monitor name is not recognized by DINIT. Retry. Only HP-supplied monitor names may be scheduled by DINIT.

/DINIT: INVALID RESPONSE!

(DINIT) Operator entry error. Retry (no retry allowed for quiescent or re-start mode).

/DINIT: INVALID TIMEOUT

(DINIT) When entering NRV information, the value specified in the time-out override field is invalid. Value may be defaulted or numeric in the range 0 to 1275, inclusive. All other values are invalid.

/DINIT: INVALID UPGRADE LEVEL

(DINIT) Values of 0 or 1 are valid. All other values are invalid.

/DINIT: LU ERROR

(DINIT) In response to a request for an LU, you have either:

1) entered a non-numeric entry (not /E)

2) entered an improper LU number

3) entered an LU number not linked to DVA65, DVA66, or DVG67

4) entered an LU number for which no line timeout exist. (DVA65 only)

The above errors are easy to correct, although you may have to abort DINIT to find the communication LU or assign a line timeout value.

An additional cause for this error is the failure to set up the EQT entry with the proper extent size. This requires re-generation. See the discussion of Equipment Table set-up in the GENERATION Chapter of the Network Manager's Manual.

/DINIT: (MA TIMEOUT * RTRYS) > MASTER TIMEOUT

(DINIT) The time set for Message Accounting retries times the number of retries exceeds the Master Timeout value for the node.

/DINIT: NODE SPEC. ERROR

(DINIT) Improper nodal reference value:

1) CPU node number specified is negative

2) Communication line LU reported is negative, non-numeric, not connected to DVA65/66, or EQT not set up with proper extent size and line time-out (see all the causes for the message LU ERROR).

3) Time-out specified is negative or larger than 1275.

/DINIT: NO CLASS #s

(DINIT) Startup was attempted on a system without enough class numbers. Regeneration is required to run DS.

/DINIT: NO SYSTEM MEMORY!

(DINIT) Insufficient system available memory (in RTE-A systems, insufficient system memory block area) for use by the network. DINIT cannot initialize the system, and aborts. Re-generation of RTE may be required.

/DINIT: NOT ENOUGH LINKS SPECIFIED FOR REROUTING!

(DINIT) The number of rerouting links requested exceeds the number of LUs actually enabled.

/DINIT: READ ERROR

(DINIT) End-of-file or FMGR error has been detected on the input device/file. The question is repeated on the error LU device. The user may supply the required response from this device.

/DINIT: RSM UNABLE TO LOG OFF ITS SESSIONS

(DINIT) Upon shutdown, a log-off request is sent to the local Remote Session Monitor (RSM) for every local session created from a remote node. DINIT will wait up to 15 seconds for these sessions to be logged off.

/DINIT: RN ERROR

(DINIT) A required resource number cannot be allocated. DINIT is aborted. Re-generate, with a larger allotment of resource numbers.

/DINIT: SIZE OF SAM AREA EXCEEDS 32K!

(DINIT) The table area requested exceeds 32K. This is probably due to a typographical error. In RTE-A, this refers to insufficient system memory block area.

/DINIT: THIS LU ALREADY HAS A NEIGHBOR!

(DINIT) Two nodes were assigned as neighbors on the same LU. Correct the current entry or abort DINIT and begin again.

/DINIT: TR FILE ERROR

(DINIT) The file name specified in the schedule parameters could not be opened. Possibilities include:

- No such file
- File name spelled incorrectly
- File is currently open to another program
- File has a negative security code

Correct the file problem, and reschedule DINIT.

/DINIT: WARNING! XXXXX IS A REQUIRED PROCESSOR FOR DS/1000-IV

(DINIT) The named program, <XXXXX>, is required for DS/1000-IV to function properly, but DINIT could not schedule it. DS/1000-IV will not function correctly until this program is loaded.

1) If the program is UPLIN, DINIT will abort. Load UPLIN and re-initialize DS/1000-IV.

2) All other programs can be loaded later and will be scheduled by UPLIN.

```
/DINIT:  WARNING! - SESSION MONITOR NODE HAS
/DINIT:             WRONG REMOTE-SESSION LIBRARY!
```

(DINIT) If the Remote Session Library $DSLSM was used in a Session Monitor node, this message will be displayed. DINIT will continue as if in a non-session node so that the RTE system will at least be usable to recover and re-generate.

/DINIT: WARNING! # OF SESSIONS FOR REMOTE = nn

(DINIT) If not enough undefined LUs are available, DINIT will display this warning on interactive terminals and on the system console and continue.

DS/1000 HAS NOT BEEN INITIALIZED

(DSLIN) DS/1000-IV software has not been initialized. Run DINIT, specifying links to an HP 3000 and try again.

DOWNLOAD OF FILE:<file name>::-<lu>:<type>

```
                            AM
AT DAY nnn <hr>,<min>:<sec>     <error message>
                            PM
```

A program download error occurred. Refer to the section on Program Download Error Messages in this appendix for a detailed discussion of this error and possible error messages.

DS ERROR: PROG=ppppp STREAM=yyyyyy SEQ #=ddddd
P=ppppp A=aaaaa B=bbbbb

An unexpected error has occurred. Refer to the section on DS ERROR: (UNEXPECTED ERRORS) in this appendix for a more detailed description.

DS ERROR: UP/DOWN COUNTER EXCEEDED
LINK LU # xxx IS DISABLED

Rerouting has determined that the status of a communications link has changed too frequently. To increase communications stability it has disabled the link.

DS ERROR: COMM. READ, LU = xxx I/O STATUS = yyyyy

where I/O STATUS bits are:

```
15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
      \                        / \              /  |   |   |   |
       \----------------------/   \------------/   |   |   |   v
            device type              error         |   |   | always
          (octal 65 or 66)            type         |   |   |  zero
                                                   v   |   |
                                                DS=0   |   v
Error types (octal):                          non-DS=1 |  error bit
                                                       |  1 = error
 0 = no error                                          |
 1 = line failure                                      v
 2 = time-out                                        read=1
 3 = local busy                                     write=0
 4 = message aborted                                request
 5 = remote busy
10 = not initialized
11 = wrong mode
12 = illegal request
13 = card failure
17 = bad interrupt
```

(DRV65) The communications driver has reported a line error. A message sent to this node could not be read due to parity error or line time-out. The transmitting side will attempt to re-try automatically, after a delay, up to a fixed limit, so no action is required by the operators. This could cause rerouting to adjust around this bad link if the LU goes down. If this error occurs frequently, it is an indication that the communications line is probably noisy or line time-out is too small. See the alternatives listed for error DS01 to clean up the line.

(DVA66) The receive LU/EQT does not follow transmit LU/EQT or GRPM detected an error in the read process. Read process errors could be caused by:

1. Timeout waiting for incoming frame.
2. Attempt to read data when link is not logically connected.
3. Message type and frame size do not agree.
4. Message length and read length do not agree.
5. Line fails during read process.

NOTE: For PSI BISYNC links the status is displayed in bits 0-3. All other bits are zero.

```
DS ERROR: COMM. WRITE, LU= xxx  I/O STATUS = yyyyy
```

See DS ERROR: COMM. READ for status information.

```
DS ERROR: SELF-CHECK ERROR IN MESSAGE ACCOUNTING!
          nnnnn xxxxx yyyyy zzzzz
```

nnnnn = Node number. The other numbers are special flags for HP internal use. One of the Message Accounting modules has determined that there is an internal system failure. It has disabled itself. This should never occur. Consult your Network Manager and reboot.

```
DS ERROR: SELF-CHECK ERROR IN xxxxx
           P=ppppp A=aaaaa B=bbbbb
           REROUTING IS DISABLED FROM THIS NODE
```

xxxxxx= Name of rerouting module
pppppp= Octal contents of the P-register
aaaaaa= Octal contents of the A-register
bbbbbb= Octal contents of the B-register

One of the rerouting modules XXXXX has determined that there is an internal system failure. It has disabled itself. This should never occur. Consult your Network Manager and reboot.

```
DS ERROR: SLAVE TCB NOT FOUND, POSSIBLE TIMEOUT
          STREAM=xxxxx ORG NODE=yyyyy DEST NODE=zzzzz
```

A reply was being sent for which there was no associated request. The slave monitor or PTOP slave may have taken too much time processing this reply.

```
DS ERROR: TCB NOT FOUND, POSSIBLE TIMEOUT
          STREAM=xxxxx ORG NODE=yyyyy LU=zzzzz
```

A reply arrived which could not be matched to any master TCB. Most probable reason is that the master has already timed-out or was aborted by the operator. If Message Accounting is in the system and a reply was slow in returning, this might occur due to MA's retransmission sequence.

If this error occurs frequently, increase the master timeout value.

```
DS ERROR: ttuu(v), REPORTING NODE wwwww --REPLY FLUSHED
          STREAM=xxxxx ORG NODE=yyyyy DEST NODE=zzzzz
```

This error occurs when a reply, possibly an error reply, can't be returned to the origin node.

```
DS ERROR: ttuu(v), REPORTING NODE wwwww --REPLY FLUSHED
          STREAM=xxxxx ORG NODE=yyyyy LU=zzzzz
```

This error occurs when a reply, possibly an error reply, can't be delivered back to the origin node.

```
DS ERROR: PROG=xxxxx STREAM=yyyyy SEQ =zzzzz

            P=ppppp A=aaaaa B=bbbbb
            xxxxx  = name of program reporting the error
            yyyyy  = stream code
            ddddd  = sequence number of the request or reply being
                     processed when the error occured
            pppppp = P-register address where the program detected
                     the error (you'll need listings)
            aaaaaa = contents of the A-register when the error occured
            bbbbbb = contents of the B-register when the error occured
```

DS/1000 ERROR nnn

(RMOTE) The reported numeric DS/1000-IV error occurred during a file move operation.

DS MSG: LU # xxx JUST CAME UP

Rerouting has just been informed that this link LU is now functioning. This is an informational message appearing on the system console.

DS MSG: LU # xxx JUST WENT DOWN

Rerouting has determined that this link LU is no longer functioning. This is an informational message appearing on the system console.

DS MSG: MESSAGE ACCOUNTING REMOVED FROM NODE XXXXX

This node does not have Message Accounting (or it is not initialized). Message Accounting has been disabled to this node. This is an informational message appearing on the system console. This node has been declared to have Message Accounting at DS initialization and it does not have the proper Message Accounting software generated in. Regenerate with MA software or do not indicate this node as an MA mode at initialization.

```
>> DS/3000 COMMUNICATION LINK *DOWN*
>> xxxxxxxxxxxxx @  yyyyyyyyyyyyy
```

(QUEX) This message format is displayed if initialization is not properly established or the link fails after initialization. QUEX tries to establish the link to the HP 3000 automatically.

NOTE: For abortive communication errors, the list of remote HP 3000 sessions is cleared and any program waiting on a reply from the HP 3000 is timed out. If a user makes a request after an abortive error he will no longer be logged on and will get an illegal request error.

X and Y field errors are detailed on the following pages.

Error Messages printed by HSI (hardwire) version of QUEX

xxxxxxxx FIELD	DESCRIPTION
DATA OVERRUN	The received text did not fit into QUEX's buffer. It is possible that the text-ending characters became garbled, or that, because of overloading, the remote interface started inserting Sync characters into transparent text.
DLE EOT RECEIVED	The remote station detected a protocol failure and forced a disconnect by sending DLE EOT.
EOT RECEIVED	The EOT protocol character was received in response to the local station's ENQ or ACK0.
HARDWARE FAILURE	Line error detected by local interface board. This indicates that the local and remote stations are not or never were communicating.
IMPOSSIBLE ERROR	A valid SLC event seems to have occurred but according to the SLC state transition tables, it never should have occurred in the present SLC state. Perhaps noise on the line has changed a valid message to an invalid one.
INVALID REQUEST	The function given with the request is not meaningful or the text contains control characters which are not allowed.
MAX ENQ RECEIVED	Maximum number of ENQs received in a write conversational subroutine.
MAX # ENQ SENT	The local station has been sending text, but instead of the correct acknowledgment, it received a garbled response, the wrong alternating acknowledgment, or nothing at all. Each time (up to the maximum number specified), it sent ENQ as a means of error recovery with no satisfactory response. Indicates line not open or initialized on the HP 3000 side, or connection is broken.

Error Messages printed by HSI (hardwire) Version of QUEX (cont.)

xxxxxxxx FIELD	DESCRIPTION
MAX NAKS RECV'D	The local station has been sending text but the remote device received a bad block check and so it returned the negative acknowledgment. Each time, the local station resent the text until the maximum specified number of tries was reached. Indicates a noisy communication line or faulty interface (block-check generated incorrectly or receive-side comparison faulty).
NAK RECEIVED	This is a BSC convention saying that the remote user is not ready to receive.
NO NAK TO TTD	The local user has just made a Write Delay request and the local station has sent TTD (Temporary Text Delay). This message looks like aborted text in its shortest possible form. Somehow, the remote station has misunderstood the message and has returned something other than the standard BSC reply to TTD (which is NAK).
RVI RECEIVED	Reverse Interrupt received. The local station was sending text and instead of regular positive acknowledgement, an RVI was received. This is a Bisync convention saying that the user at the remote station wants to stop receiving text and start sending text.
SENT EOT,GOT ENQ	The local station has just sent text and closed the exchange by sending EOT. The remote station has taken advantage of the three-second period to start a "write".
TIMEOUT	While awaiting a message from the remote device (when either sending or receiving), the specified number of three-second timeouts occurred.
WRONG LINE STATE	The request issued was incompatible with the current line state.

Error Messages Printed by HSI (hardwire) Version of QUEX (cont.)

yyyyy FIELD	DESCRIPTION
LINE OPEN	Readies the line for read/write operations
READ INIT	Attempt to read data when RTE has nothing to send
SEND ENQ	Bid for the line
SEND EOT	End of transmission
WRITE CON	Transmission of data from RTE and reception of data from the HP 3000

Error Messages Printed by PSI (modem) Version of QUEX

xxxxx FIELD	DESCRIPTION
CARD FAILURE	The local card has quit responding to the driver.
ILLEGAL RQST	The command is not supported by the driver.
LINE FAILURE	Link is not connected.
REMOTE BUSY	The link is connected, but the local card's buffers are full.
TIMEOUT	No message was received within the timeout period.
UNINITIALIZED	The card has not been initialized.
WRONG TYPE	The card is not set up for DS/1000-DS/3000 messages.

yyyyy FIELD	DESCRIPTION
DISCONNECT	Disconnect the link.
GET PARAMS	Read parameters from the board.
INITIALIZE	Initialize board.
PRIMARY CONCT	Connect as primary station.
READ	Read a buffer from the card.
SECNDRY CONCT	Connect as secondary station.
WRITE	Send a buffer to the card.

>>DS/3000 COMMUNICATION LINK *UP*

(QUEX) Displayed once initialization of the DS/3000 Communication Link is properly established.

DS/3000 ERROR nnn

(RMOTE) The reported numeric DS/3000 error occurred during a file move operation.

DSxx Error Codes

For the following errors, the number shown in parenthesis is the error qualifier code. It is available to the programmer via the DSERR call. REMAT always prints the error and the qualifier.

DS-1(0) - PROGRAM NOT FOUND

DS-1(1) - PROGRAM DOESN'T EXIST

(DEXEC) This error is the result of the special DEXEC request code 99 used to aquire program status. If the request is directed at the local node, DEXEC will process the request locally and return this error if necessary.

(DEXEC) The program may exist as a type 6 file, but it does not possess an ID segment. Use the FMGR 'RP' command to restore the program.

DS-1(2) - PROGRAM DOESN'T EXIST

(EXECM) This error occurs when the program issues the DEXEC call with request code 99 to aquire program status. If the request is directed to a remote node, EXECM processes the request and returns this error if necessary.

(EXECM) The program may exist as a type 6 file, but it does not possess an ID segment. Use the FMGR 'RP' command to restore the program.

DS-2(0) - DUPLICATE FILE NAME

(FCOPY) The named file and the attempted ".." retry name both exist at the destination node.

DS-4(0) - RECORD SIZE ERROR

(FCOPY) Record size too large.

DS-6(0) - ILLEGAL RECORD SIZE

(FCOPY) The destination file size is negative.

DS-10(0) - NOT ENOUGH OR ERROR IN PARAMETERS

(RFMST - Remote File Access Intrinsics.)

DS-28(0) - NO DCBS AVAILABLE

(RFAM) All the DCBs allocated are currently in use. Close files not needed programmatically or by using the REMAT FLush command. Reloading RFAM and sizing it up will provide 6 DCBs per added page.

DS-100(0) - "BREAK" ENTERED

(FCOPY) The RTE command BR was used to terminate execution of FCOPY.

DS00(0) - LOCAL NODE IS QUIESCENT

(D3KMS) This error is returned when a program tries to generate no-wait call traffic out of a node that is attempting to quiesce. This temporary condition will be corrected when the restart command, /R, is issued using DINIT.

DS00(1) - LOCAL NODE NOT INITIALIZED WITH DINIT (#GRPM = 0)

(#MAST) This error can occur when only a 1000-3000 link has been defined and REMAT or any other master routine intended for 1000-1000 communication is called to attempt access to another 1000. If other 1000s exist in your network, shutdown DS and reinitialize with a correct network description.

DS00(2) - LOCAL NODE IS QUIESCENT (#QRN IS LOCKED GLOBALLY)

(#MAST) This error is returned when a program tries to generate traffic out of the local node while the local node is quiesced. This temporary condition will be corrected when the restart command, /R, is issued using DINIT.

DS00(3) - LOCAL SYSTEM IS QUIESCENT OR IN PROGRESS OF QUIESCING

(#MAST) Returned when a program tries to generate no-wait call traffic out of a node that is attempting to quiesce. This temporary condition will be corrected when the restart command, /R, is issued using DINIT.

DS01(0) - COMMUNICATION LINE PARITY, PROTOCOL FAILURE, 'STOP' RECEIVED, CABLE DISCONNECTED, OR OTHER HARDWARE ERROR

(DVA66 or DVA65) On DVA66 links, this error means that the link is not connected. For DVA65 links, the computer on the receiving side is responding but not receiving the data properly. This may occur if the interface cards on each side are not jumpered identically (see the appropriate interface manual for the proper settings).

This message is reported at both master and slave sides when all driver retry attempts fail.

Suggested recovery:

If the above checks have been made and the installation found OK, try any or all of the following:

- Use better line conditioning (leased lines).
- Reduce data buffer size in application programs. Large buffers have a higher probability of incurring a line "hit".
- On Direct-Dial lines, hang up and re-dial. This will usually provide a different connection. You may need to repeat this once or twice in order to obtain a low error-rate connection.
- Consider obtaining modems with adaptive conditioning. Refer to the DS/1000-IV hardware manuals or the references in the Network Manager's Manual Bibliography.
- Improve the shielding in the cables (modem cables, as well as hardwire). Route all cables well away from sources of electromagnetic noise, such as power transformers, flourescent lights, electrical motors, etc.
- In particularly noisy environments, when the remedies above are insufficient, programs should provide a limited number of retries.

DS02(0) - COMMUNICATION LINE TIMEOUT ERROR

(DVA65) For DVA65 links only. Line timeout is too small or the computer on the receiving side is not responding at all (i.e., DS is shutdown or was never initialized or that link was never enabled at the remote end). The computer may have halted for some reason, failed to respond to an interrupt, its power may have failed, or the communication cable is not connected to the interfaces on both sides.

DS03(0) - ILLEGAL RECORD SIZE.

Indicates programming error. The inbound data buffer size exceeds the maximum space allocated by the user buffer.

DS03(1) - INBOUND DATA LENGTH TOO LARGE FOR INPUT CONVERTER

(INCNV) INCNV is loaded with a fixed buffer-size module. This buffer must be capable of receiving the entire incoming message. If a larger buffer is required, consult the Network Managers Manual (#CVBF) for information on expanding this buffer.

DS03(2) - OUTBOUND REQUEST HEADER LENGTH TOO SMALL

(#MAST) The message header must be at least 13 words long. This error indicates a DS internal error has occurred. Consult your Network Manager.

DS03(3) - OUTBOUND REQUEST HEADER LENGTH TOO LARGE

(#MAST) The request header must not be larger than 63 words. If this error occurs it indicates a DS internal error. Consult your Network Manager.

DS03(4) - INBOUND REQUEST HEADER LENGTH TOO LARGE

(#GET) Request header must be greater than or equal to 13 words and not greater than 63 words. This error indicates a DS internal error. Consult your Network Manager.

DS03(5) - INBOUND DATA LENGTH TOO LARGE FOR USER BUFFER

(#GET) The data received is larger than the user buffer provided to receive the reply. This happens when a record from a file is read that turns out to be larger than the buffer provided to read the record or when in Program to Program Communication the incoming buffer is larger than the buffer space allocated by the user.

DS04(0) - ILLEGAL NODAL ADDRESS, NODE ADDRESS NOT IN NODAL ROUTING VECTOR (NRV) TABLE

NRV at node reporting error does not contain an entry for the destination node. Possibilities are:

- If local processing is requested (node "talks to itself"), NRV set up via DINIT did not contain an entry for the local node. Shutdown, edit the answer file and re-initialize.

- NRV built improperly, either at this node or at one of the intervening nodes. Use DSMOD at each node to display the NRVs, and compare this to your network topology. Correct the DINIT initialization files (nodal addresses) and re-initialize (quiesce, shutdown and restart or reboot) all nodes that had incorrect NRVs.

- May also occur if a remote file is opened or a slave program is scheduled without going through proper initializations such as DOPEN and POPEN calls.

- Programming error. Node number specified is not the proper node.

DS04(1) - ALL ROUTES TO THIS NODE ARE DOWN (REROUTING)

The rerouting software has determined that all routes to the destination node are down and the message cannot be transmitted. (Note that if 12665 (DVA65) links are included in the network they are not automatically restored to service.) It may be possible to restore links with the /L command and change the NRV with the CN command of DSMOD. DVA66 links are automatically enabled when the link problem is corrected or goes away.

This error can occur when the links at a store and forward node are down, not necessarily just links at the local node. The reporting node number can be used to determine how far into the network the message traveled.

DS04(2) - HOP COUNT EXCEEDED

Message has exceeded the maximum number of store-and-forwards specified at DS/1000-IV initialization time (or as modified by the DSMOD '/T' command). This may indicate that the NRV at this node or other nodes is specified incorrectly.

Check NRVs and use DSMOD 'CN' or '/T' to correct this error.

DS05(0) - REQUEST TIMEOUT

(#MAST) This error can occur for a number of reasons:

1) A slave monitor may be "tied up" for too long and there is no message accounting in the system (message accounting will cause a DS05(1). For example, assume we run REMAT and execute the TEllop command, sending a message to the remote node's system console. If one of the keys has been depressed at the remote node's system console, RTE will be waiting for an input, so a remote TEllop command will time out, although the message will be displayed as soon as the operator enters a command or the terminal's time-out elapses.

   The same thing can happen for DMESS, DMESG, and DEXEC calls. It may also occur when using the REMAT RW (run with wait) command, where someone else may be running a program with wait, preventing EXECW from servicing the other requests until the initial request is done. In this case, use the RU command in REMAT.

2) The master timeout may be too small or the destination node may be loaded heavily or generated improperly (with not enough resources such as disc swap tracks, available partition space, and SAM). Note that unavailable SAM is only a problem in generating a reply message. If insufficient SAM is a problem at the destination node, a DS08 error is returned to the master.

3) The reply could not be sent back due to an irrecoverable transmission error in the reply. In this case, an error message will be printed on the system console of the node which could not successfully transmit the reply.

4) The response may fail to be returned due to an error in the NRV in some node involved in the reply path which could not access the origin.

Note: Retry on a DS05 error is not always a good practice for the following two reasons:

- The request may have been correctly delivered to the destination node and have been executed, but the destination node may have a problem in sending back the reply or the reply just takes too long to reach the origin node. Retry under this situation may have adverse effect. For example, if the request was to credit or debit an account in a central bank node, this retry may cause a double-entry error.

  If receiving the same buffer twice creates a possible disastrous effect, as in the preceding example, the buffers should be assigned a sequence number passed either in the tag field or as part of the data buffer. This allows the receiving (Slave) program to detect and reject duplicate buffers.

- Although it is timed out at the origin node, the request may still exist elsewhere in the network, e.g. in a forwarding node. Retrying without limit may possibly "flood" the network with copies of the same request. This will add unnecessary burden to the network, e.g. additional usage of SAM to contain the requests.

A DS05(0) error can occur if Message Accounting does not exist at this node. The most straight-forward solution would be to generate Message Accounting in. If this is not possible or feasible, message sequencing is available using PTOP calls (not with DEXEC or RFA). Another solution to avoid DS05(0) error may be to increase the master time-out value (through the use of DSMOD). If it is absolutely necessary to have retries in an application, then a limited number of retries should be specified.

DS05(1) - REQUEST WAS RECEIVED AT THE DESTINATION NODE, BUT THE REPLY WAS NOT RECEIVED IN TIME.

The Message Accouting software confirmed deliver of the request at the destination node but the reply had not arrived by the time the Master Time-out expired.

DS05(2) - TOO MANY RETRIES BY MESSAGE ACCOUNTING

The maximum number of Message Accounting retries have been attempted and the message has not been successfully received by the destination node. This should rarely occur unless all routes to the node are down. However, if it does occur, increasing the number of retries in the DINIT answer file may solve the problem. The Message Accounting retry count is specified with the response to "# OF MA NODES? [,RETRY COUNT]" during the DS initialization.

DS05(3) - TIMEOUT SIGNALED BY MESSAGE ACCOUNTING

An acknowledge has not arrived in time and Message Accounting has declared a timeout error. If this error occurs frequently, increase the timeout values and retry count.

Note that this error differs from the DS05(0) error in that Message Accounting has signalled the timeout (i.e., the Message Accounting timeout multiplied by the number of retries has been exceeded).

DS06(0) - ILLEGAL REQUEST, OR MONITOR NOT ACTIVE

When the DS06(0) error occurs the error is being reported by programs in the destination node. The error can be caused by any of the following reasons:

1) Either the DS Monitor which services this type of request is not active or the request has been determined to be illegal by the Monitor program. Load monitor on-line, use DSMOD's /S command to schedule it, and re-try.

2) Unacceptable DEXEC call: ICODE parameter value not defined for remote-EXEC calls.

3) When the line to the HP 3000 goes down, the HP 3000 shuts down all sessions associated with the failed link. When the line returns to normal, programs using the previously valid process numbers receive the DS06 error indicating the process number is now invalid.

4) In segmented programs accessing the HP 3000, if D3KMS is appended to the segment rather than the main, the process number will always be overlaid when the segments swap, causing the use of an invalid process number. A dummy call to D3KMS in the main will cause this collection of routines to be appended to the main and eliminate the problem of overwritten process numbers.

DS06(1) - THE APPENDED ROUTINE "DEXEC" HAS DETECTED AN ILLEGAL OR MISSING PARAMETER IN THE CALLING SEQUENCE

(DEXEC) Examine the CALL statement and make appropriate corrections.

DS06(2) - ILLEGAL REQUEST CODE OR FUNCTION NOT SUPPORTED

(EXECM) Examine the CALL statement and make appropriate corrections.

DS06(3) - ILLEGAL REQUEST CODE OR FUNCTION NOT SUPPORTED

(EXECW) Examine the CALL statement and make appropriate corrections.

DS07(0) - SYSTEM TABLE ERROR

Possible malfunction in some software which caused DS/1000-IV tables to be corrupted or destroyed. At this point rebooting or shutdown/restart are probably the only viable solutions.

DS07(1) - INPUT/OUTPUT CONVERTERS (INCNV/OTCNV) REQUIRED BUT NOT SCHEDULED

(#MAST) An attempt has been made to access a node of a different DS software upgrade level but the proper input and output message converters are not present in the system to perform the necessary message conversion.

This error occurs when the NRV is corrected online to indicate the presence of different upgrade level nodes in the network. If no nodes of other upgrade levels existed during DS initialization the level converters are not scheduled. To get them scheduled, correct the DINIT answer file, shut down DS and reinitialize or reboot.

DS07(2) - INCONSISTENT LEVEL NUMBER BETWEEN LOCAL AND REMOTE NODES

Your NRV indicates another node in the network is of a different software upgrade level than it actually is. Check NRVs and eliminate the inconsistancy.

DS07(3) - TCB LIST IN ERROR

(#RSAX) DS internal tables located in System Available Memory have become corrupt. This error is reported by #RSAX, the Resident Table Access routine. Rebooting or shutdown/reinitialize are probably the only solutions.

DS07(4) - EXPECTED POOL OF "REMOTE SESSION" IDENTIFIERS (#POOL) DOES NOT EXIST.

This error can happen if DINIT is rebooted with the library $DSNSM (no session monitor node anywhere in the network) and other modules are relocated with $DSSM or $DSLSM. These other modules include REMAT, DS user programs, and DS slave monitors.

Re-load DINIT with the correct library and re-initialize DS/1000-IV.

DS08(0) - REMOTE BUSY OR RESOURCE UNAVAILABLE

This error is returned to master program if any of the following conditions exist:

- Insufficient SAM exists at the destination node to contain the request
- QUEUE is busy
- Remote system is quiescent

NOTE: This may occur if an operator quiesced the system and then left it quiesced for a long enough time that another system's re-try attempts could not succeed. For example, if a slave monitor is being replaced, the system must be quiescent. This error might then occur.

- No available TCBs in destination node. This may be a temporary condition, in which case the request should be re-tried a limited number of times. However, if there exists a serious shortage of TCBs, this error will occur often and the retries will not always succeed. In this case, re-boot the destination node and answer DINIT's question " TRANSACTIONS?" with a larger number.

- Monitor servicing this request is in "unavailable memory suspend" (state 4 on an RTE-L) or simply not available to process your request (operator suspend, etc). If unavailable memory suspend is the problem, this indicates a serious shortage of SAM. Consult your System Manager.

- Interrupt Table entry at remote node is not correct. Check the generation listing and refer to the Generation Chapter of the DS 1000-IV Network Manager's Manual. If the entry is incorrect, re-generation or a "patch" put in by your HP field support engineer is required.

- The logical unit in a DEXEC call is "down". Check the LU status with remote operator LU and EQ commands: the first to check the status of the logical unit, the second to check the status of the EQT.

- The logical unit in a DEXEC call has been locked by a program in the destination node. This may be a temporary condition if the locking program is using the LU for a limited period of time.

- The error may be in the destination node or intervening nodes. If possible, your programs should print the number of the node reporting the error, particularly when using store-and-forward.

- The system attempts the number of retries given by the "Remote Busy Retries" parameter (see RMOTE chapter) with a delay of one second between each. If none succeed, the system will perpetually re-try if the Remote Quiet Retry parameter is non-zero. If the value is zero, this error is returned to the user.

Most of the above possibilities are indications of an error in system generation if seen frequently. You should check for the following:

- Too few swap tracks.
- Too little System Available Memory.
- QUEUE made disc-resident in node with heavy data traffic.
- Too few TCBs allocated.
- The number of requests queued on a slave program's class has exceeded the limit specified in #QCNT. This situation arises when the slave program cannot keep up with the rate of the incoming requests.

  It may be "hung up" in some non-dormant state, but unable to execute nor to be swapped into memory, or waiting for resources. In the case where a slave serves simultaneously a number of masters which exceeds the queue limit, #QLIM should be overridden with a new value, [(maximum allowable blocks in queue)+1].

If seen infrequently, this error merely indicates a temporary overload and should cause little concern.

DS08(1) - TOO MANY UNACKNOWLEDGED MESSAGES BETWEEN SOURCE AND DESTINATION NODE FOR MESSAGE ACCOUNTING

The maximum number of unacknowledged messages has been exceeded. Retry the request.

DS08(2) - INSUFFICIENT RESOURCES

(EXECM) a) Class I/O error or insufficient SAM for class I/O Read, Write, or Control request.
b) Class I/O error or insufficient SAM for EXECM's internal usage.
c) Attempted I/O request to a disc LU.

DS08(3) - INSUFFICIENT RESOURCES

(EXECW) a) String could not be passed to scheduled program.

b) APLDR could not be scheduled (RTE-L, RTE-MIII).

DS08(4) - INSUFFICIENT RESOURCES

(#RQUE) a) The number of requests queued on a slave program's class has exceeded the limit specified in #QCNT. This situation arises when the slave program cannot keep up with the rate of the incoming requests. It may be "hung up" in some non-dormant state, but unable to execute nor to be swapped into memory, or waiting for resources. In the case where a slave serves simultaneously a number of masters which exceeds the queue limit, #QLIM should be overridden with a new value, [(maximum allowable blocks in queue)+1].

b) The LU, onto which the class I/O request is to be requeued, is down.

c) The EQT, onto which the class I/O request is to be requeued, is down.

d) The destination class queue cannot contain another outstanding request.

e) RTE-L rejected the Rethread request.

DS08(5) - NO AVAILABLE CLASS NUMBERS IN SYSTEM

(#MAST) All class numbers are in use. DS monitors use class numbers which may require more class numbers to be allocated at generation.

DS09(0) - ILLEGAL OR MISSING PARAMETERS

Indicates programming error. Correct program calling parameters and re-run.

DS09(1) - NOT ENOUGH PARAMETERS IN #MAST CALL

(#MAST) This is a DS internal error. One of the DS intrinsics has called with insufficient parameters. Consult your Network Manager.

DS09(2) - NEGATIVE WRITE DATA LENGTH IN #MAST CALL

(#MAST) #MAST requires positive word length specifications. DEXEC does not support negative character length specifications.

DS09(3) - NEGATIVE READ DATA LENGTH IN #MAST CALL

(#MAST) #MAST requires positive word length specifications. DEXEC does not support negative character length specifications.

DS09(4) - NEGATIVE MAXIMUM EXPECTED REPLY LENGTH

(#MAST) On a read request the maximum expected reply length can't be specifi as a negative character length. Change to positive word length and try again.

DS09(5) - NEGATIVE PARAMETER IN #GET CALL

(#GET) This error is a DS internal error. One of the DS intrinsics is call #GET with negative parameter values. Inform your Network Manager.

DS09(6) - MISSING PARAMETER IN #GET CALL

(#GET) This error is a DS internal error. One of the DS intrinsics is calling #GET without all the necessary parameters.

DSLIN IS ONLY USED FOR PSI BISYNC LINKS.

(DSLIN) You do not need to run DSLIN with HSI links. The low-level connection is already established. Continue establishing your remote session.

This message is also displayed if DS is not enabled for the HP 3000. Check you DS configuration.

DSMOD Error Codes

/DSMOD: CLASS I/O ERROR

(DSMOD) A required class number cannot be allocated. DSMOD is aborted. This error may require re-generation with a larger allotment of class numbers.

/DSMOD: DSMOD ABORTED

(DSMOD) /A command entered or irrecoverable error occurred. Explanation will have been printed prior to this message. If input from a file, line number of last error also printed.

/DSMOD: END DSMOD

(DSMOD) Normal completion message. The ten characters comprising the message are also returned in the 5-word temporary storage area of a scheduler's ID segment. They may be recovered through the use of RMPAR. If DSMOD has been aborted, the five words returned are: 100000B,ER, D,SM,OD

/DSMOD: ERROR: MON?: XXXXX

(DSMOD) The specified monitor XXXXX is not in the system. Load the monitor, then use DSMOD to schedule it.

/DSMOD: ERROR: STAT: XXXXX

(DSMOD) The monitor's status is not 'dormant' and therefore it cannot be scheduled. Abort DSMOD using the /A command and then use RTE operator commands to change the status.

/DSMOD: FILE ERROR

(DSMOD) Improper response to "INPUT # OF FILES". Retry.

/DSMOD: INVALID NAME!

(DSMOD) Monitor name is not recognized by DSMOD. Retry.

/DSMOD: INVALID RESPONSE!

(DSMOD) Operator entry error. Retry. (No retry allowed for quiescent of re-start mode.)

/DSMOD: LU ERROR

(DSMOD) Improper LU number specified or LU number does not point to the communication link driver. Retry.

/DSMOD: LU ERROR: "/L" NOT VALID FOR X.25 POOL LUS. USE DSMOD "DI" TO DISABLE THE LU.

(DSMOD) The line re-enable command ("/L") cannot be used to re-enable an X.25 pool LU. You must use the "DI" command to disable the LU and return it to the pool.

/DSMOD: NODE SPEC. ERROR

(DSMOD) Improper nodal reference value. DSMOD aborted. Correct initialization answers and restart DSMOD.

/DSMOD: NOT AVAILABLE IN NON-SESSION VERSION

(DSMOD) If the wrong Remote Session Library was used ($DSLSM instead of $DSSM) in a Session Monitor node, this message will be displayed when the /U and /P commands are attempted.

/DSMOD: READ ERROR

(DSMOD) End-of-file or FMGR error has been detected on the input device/file. The question is repeated on the (error LU) device. The user may supply the required response from this device.

/DSMOD: RN ERROR

(DSMOD) A required resource number cannot be allocated. DSMOD is aborted. Re-generation with a larger allotment of resource numbers may be required.

/DSMOD: TR FILE ERROR

(DSMOD) The file manager cannot process the file which was specified in the scheduling parameters. Correct the file problem and re-schedule DSMOD.

EDITR WAITING FOR LIST DEVICE

(EDITR) When listing to a local lineprinter, the EDITR attempts to lock the device to prevent other programs from printing to the lineprinter at the same time. If this "lock" is denied because the device is in use, the EDITR prints this message and then suspends, waiting for the device to become free. This message is for information only; no action is required.

ERR-xx

(RTMLG) Error xx occurred while attempting to run RTMLG. Refer to the section on RTMLG Generation Errors in this appendix for a more detailed description on the different types of errors possible and suggested corrective action.

ERROR OPENING aaaaa

(DSLIN) DSLIN could not open the command file aaaaa. Check syntax.

FM-XX(0) - SEE FMGR-XXX FOR DESCRIPTION

File manager errors are documented in the help file under the FMGR-XXX or FMGR XXX string format.

HELLO FAILED OR LINE DOWN

(RMOTE) The HELLO command was not correct or could not be transmitted due to a line error.

>>HP 3000: BAD BUFFER OUTGOING

(QUEX) A transmission was attempted which did not pass a verification test in QUEX. REPORT THIS MESSAGE TO YOUR HP CUSTOMER ENGINEER.

>>HP 3000: BAD BUFFER RECEIVED

(QUEX) A message was received which did not pass a verification test in QUEX. REPORT THIS MESSAGE TO YOUR HP CUSTOMER ENGINEER.

>>HP 3000 LINK READY FOR DIALING

(QUEX, modem version) QUEX has caused the node to be in the primary (calling) state. QUEX tries to write the DS/3000 initialization request to MPE every five seconds until it is successful or a time-out (255 seconds) is reported. To move to the secondary state (receive) enter the RTE command "BR,QUEX".

>>HP 3000 LINK DIALING TIMEOUT. NOW AWAITING CALL

(QUEX, modem version) A time-out has occurred and QUEX has caused the node to be in the secondary state (receive). In this state, RTE waits indefinitely for a DS/3000 initialization request from MPE. The 1000 will accept incoming calls but cannot call other nodes in the network. If you wish to go from primary to secondary mode without waiting for the 255 second time-out, enter the RTE command "BR,QUEX". Within ten seconds the AWAITING CALL message will appear on the system console. To go from secondary (receive) to primary (calling) mode, re-enable the 3000 LU via DSMOD.

ILLEGAL INTERRUPT FROM SC XX OCTAL

Reported by ID.66 when QUEUE is absent from the system. Can be corrected by either loading QUEUE or using the RP command to restore it if it was not purged.

ILLEGAL STATUS

(RMOTE) RTE returned an SC03 scheduling error for an RU, ON, or RW command.

INITIALIZATION FAILED ON LU XXX

(DINIT) Tried to initialize non-DS LU.

INVALID INPUT

(RMOTE) Wrong or missing parameter or wrong prompt on transfer file input.

INVALID REMOTE LU

(RMOTE) From SW command: Either the LU is not in the 3000 LU table, built when DINIT was executed, or it is not the LU of the currently active session opened on the 3000.

I/O ERROR AT xxx. STATUS = yyyyy.

See DS ERROR: COMM. READ for description.

IOxx Error Codes

IOXX(0) - See RTE IOXX errors for description.

IO00(4) - INVALID CLASS SPECIFICATION

(#RQUE) Source class parameter is zero or greater than maximum class number.

IO01(1) - IMPROPER, MISSING, OR TOO MANY PARAMETERS

(DEXEC) A parameter error has been detected by the DEXEC intrinsic. Correct your DEXEC call parameters and try again.

IO01(2) - IMPROPER OR MISSING PARAMETERS

(EXECM) A parameter error has been detected by EXECM. Correct your DEXEC call parameters and try again.

IO01(4) - PARAMETER MISSING, INVALID, OR CLASS SEARCH FAILED

(#RQUE) Call parameters are missing or invalid or the search for the class header failed.

IO02(4) - INVALID LOGICAL UNIT

(#RQUE) LU is not in range (1 < LU < LUMAX) or LU is associated with EQT #0 or LU is associated with a disc.

IO04(1) - IMPROPER BUFFER SPECIFICATION

(DEXEC)
a) User buffer length specification > 512 words.
b) In a write/read request, the write length is greater than the read length.

IO04(4) - INVALID BUFFER SPECIFICATION

(#RQUE) An illegal user buffer was specified. Extends beyond RT/BG area or not enough SAM to buffer the request.

I007(2) - DRIVER REJECTED REQUEST

(EXECM) A request issued by EXECM in response to a user call to the DEXEC intrinsic has resulted in a rejection by the driver. Check your parameters and try again. This error is returned by EXECM.

I010(4) - PROGRAM WAITING ON SOURCE CLASS

(#RQUE) Another program is waiting on the source class number and requeueing is not allowed.

I012(2) - LU NOT DEFINED FOR THIS SESSION

(EXECM) The session you are executing under at the remote node does not have the specified LU defined. This may be the result of a logon failure or inadequate resources of the default session account. Either logon to an account with the necessary resources or have them added to the default account definition.

LINE IS UP BUT HP3000 IS NOT REPLYING

(DSLIN) The PSI card was initialized but the HP 3000 is not replying. HP 1000 is forced to secondary mode.

LINE IS UP WITH BUFFER SIZE xxx

(DSLIN) The link to the HP 3000 has been established. Proceed as usual.

LINK IS DISCONNECTED

(RMOTE) The link to the HP 3000 is not functioning (physically disconnected or an electrically "open" circuit).

/LOGOF:DSERR SSEE(QQ), REPORTING NODE NNNNNN

(LOGOF) Check under error code (EE) for specific subsystem (SS). This message can appear if the node being accessed by REMAT is a session monitor node.

/LOGON:DSERR SSEE(QQ), REPORTING NODE NNNNNN

(LOGON) Check under error code (EE) for specific subsystem (SS). This message can appear if the node being accessed by REMAT is a session monitor node.

LU xxx IS LOCKED

(DSLIN) DSLIN attempts to lock the LU for primary connections. The LU is already locked if another user is also running DSLIN. Wait a minute and try again.

LU xxx IS NOT IN THE 3000 LU TABLE.

(DSLIN) DINIT did not set up table space for this HP 3000 LU. Shut down DS and bring it back up, specifying LU xxx for HP 3000 communication.

LU03(4) - LOGICAL UNIT LOCKED, OR INVALID PASSWORD

(#RQUE) Via EXECM and DEXEC, the specified LU is locked to another program and/or the password is incorrect or not specified; the I/O operation cannot be performed.

START HERE

LU ERROR <4-character RTE error code> <LU>

Occurs if an LU which does not exist or is not a DS/1000-IV driver but has the same device type code is specified during initialization.

Specify an LU which is associated with the correct DS/1000-IV driver.

This may also occur if NRV or the Rerouting Tables are corrupt. Check NRV and make the corrections with DSMOD or shutdown DS using DINIT and reinitialize.

LU LOCK ERROR xxxx

(EDITR) This message may be printed when a "lock" is attempted on the local systems's lineprinter (see the error message "EDITR WAITING FOR LIST DEVICE"). The RTE system error message LU01, LU02, LU03 may appear in the xxxx field above. The most likely cause of this error is insufficient resource numbers assigned at system generation time.

MPE FILE ERROR nnn

(RMOTE) The reported FS/3000 error occurred during a file move operation.

NEED "HELLO"

(RMOTE) Attempt to send a command to the HP 3000 before issuing "HELLO".

NEED TO RUN "DINIT"

(RMOTE) Attempt to switch to remote node before the RTE node has been initialized for communications to the HP 3000.

NO BUFFER SPACE

(RMOTE) Less than 256 words of memory are available for the PTOP file move buffer used with the "MO"ve command. Terminate RMOTE and assign it more pages.

NO SLAVE AT 3000

(RMOTE) The slave program which performs the slave side processing for the "MO"ve command, does not exist as COPY3K.PUB.SYS. The "MO" command will not work until the program is prepared.

NO SUCH PROGRAM

(RMOTE) RTE returned an SC05 scheduling error for an RU, ON, or RW command.

NOT ENOUGH SAM

(RMOTE) RTE returned an SC10 scheduling error for an RU, ON, or RW command.

NOT LOCAL COMMAND

(RMOTE) HELLO or BYE under the $ prompt from RMOTE. Use SW to go remote.

NRV POSSIBLY CORRUPT

Nodal Routing Vector contains erroneous information. Use DSMOD to correct the NRV or shutdown DS/1000-IV using DINIT and reinitialize.

OLD COPY3K VERSION ON 3000 MOVE FAILED. LOAD NEW VERSION.

(RMOTE) Slave program on HP 3000 for MO command is not most recent version. Have your system manager load the new version using MVCP3.

OLD RMOTE VERSION ON 1000 MOVE FAILED. LOAD NEW VERSION.

(RMOTE) Version of RMOTE is incompatible with slave program on HP 3000 for RMOTE MO command. Have your system manager load the latest version of RMOTE on your system.

OVERWRITE?

(RMOTE) Asked when the "to" file in a file move already exists. Respond YES to overwrite the data.

PRIMARY CONNECT TIMED OUT

(DSLIN) The connect timer expired before the HP 3000 replied. Check that the HP 3000 is initialized for HP 1000 communication and try again.

```
                OPEN
/PROGL:FILE          ERROR -nnnnn
                READ
                                                      AM
DOWN LOAD OF FILE <namr> AT DAY nnn, <hr>:<min>           HAS FAILED
                                                      PM
```

(PROGL) A file-open or file-read error occurred while attempting a program download. Refer to the section on Program Download Error Messages at the end of this chapter for more detailed information.

PROGRAM BUSY

(RMOTE) An 'RU' or 'ON' command specified a non-dormant program.

QUIESCENT RN LK ERROR <4-character RTE error code>

Resource number in RES has been corrupted. Reinitialize DS.

QUIESCENT RN UNLK ERROR <4-character RTE error code>

Resource number in RES has been corrupted. Reinitialize DS.

QUEX and QUEZ Error Codes

/QUEX: INSUFFICIENT S.A.M.

(QUEX) Could not deliver an incoming DS/3000 message because there was not enough System Available Memory.

/QUEX: CLASS ERROR aaaa

(QUEX) Got the indicated ASCII error message (aaaa) when a class I/O operation was performed.

/QUEX: TRACING ERROR aaaa

(QUEX) Got the indicated ASCII error message (aaaa) when an attempt to write a trace record was made. The status of tracing is set to "down". (See LOG3K.)

>> QUEX EXPECTS HSI LINK

(QUEX) The HSI version of QUEX is loaded, but the link utilizes PSI cards.

>> QUEX EXPECTS PSI LINK

(QUEX) The PSI version of QUEX is loaded, but the link is HSI.

/QUEZ: INSUFFICIENT S.A.M.

(QUEZ) Could not deliver an incoming DS/3000 message because there was not enough System Available Memory.

/QUEZ: CLASS ERROR aaaa

(QUEZ) Got the indicated ASCII error message (aaaa) when a class I/O operation was performed.

/QUEZ: TRACING ERROR aaaa

(QUEZ) Got the indicated ASCII error message (aaaa) when an attempt to write a trace record was made. The status of tracing is set to "down". (See LOG3K.)

RECV'D LEVEL yy MSG FROM NODE xxxxx

(INCNV) Node received message of higher format level than level of the node.

/REMAT: DSERR SSEE(QQ), REPORTING NODE NNNNNN

(REMAT) See error code (EE) in specific subsystem (SS).

/REMAT: xxx

(REMAT) Numerical error message equivalent to FMGR errors.

REMOTE EDITR UNAVAILABLE

(EDITR) An attempt was made to access the remote editor in a node in which it could not be scheduled because it was already in use or does not exist. If other copies exist at that node, try again, specifying a different name suffix. Otherwise, wait for the remote EDITR to be free.

REQUEST FAILED

(RMOTE) The HP 3000 rejected the last request.

REROUTING TABLE POSSIBLY CORRUPT

Table containing the link vector information has become corrupt. Shutdown DS/1000-IV using DINIT and reinitialize.

RMOTE IOxx (RMOTE) RTE-reported I/O errors.

RMOTE SCxx

(RMOTE) Indicates bad parameters (see RTE-reported errors earlier in this section) etc.

RQ(4) - REQUEST CODE IS INVALID

(#RQUE) ICODE does not equal 17, 18, 19, or 20 or the no abort bit is not set and the request code is an inappropriate negative value.

RQCNV: BAD BUFFER. MSG FLUSHED!!

(RQCNV) The driver passed RQCNV a bad message. The message was flushed.

RSxx Error Codes

RS01(0) - SLAVE MONITOR ATTACH FAILED

(#MAST) The session at the remote node is no longer accessible to this master. Programs running under this session have been aborted.

After an error of this type, future requests will cause a new session to be established. If a specific session is required rather than the default session, you must issue a specific attach (from REMAT) or logon request.

RS01(1) - SLAVE MONITOR ATTACH FAILED

(RFAM) Same as RS01(0) above, but reported by RFAM.

RS01(2) - SLAVE MONITOR ATTACH FAILED

(EXECM) Same as RS01(0) above, but reported by EXECM.

RS01(3) - SLAVE MONITOR ATTACH FAILED

(EXECW) Same as RS01(0) above, but reported by EXECW.

RS01(4) - SLAVE MONTIOR ATTACH FAILED

(PTOPM) Same as RS01(0) above, but reported by PTOPM.

RS02(0) - CLASS I/O ERROR ON RESPONSE FROM LOGON

An error occurred during class I/O communication with the LOGON program. Try again, and if the error persists, consult your System Manager.

RS03(0) - EXCEEDED LOCAL LIMIT ON NUMBER OF REMOTE SESSIONS

a) No room in local table to catalog a remote session. The local node might need a larger value for DINIT's "# ACTIVE TRANSACTIONS?"

NOTE: PNLs for remote 1000/3000 sessions and master TCBs for remote requests are taken from the same block of SAM.

b) Local limit reached on number of remote nodes accessed by this program (limit is currently 16).

RS03(1) - NO AVAILABLE SESSION IDENTIFIERS AT DESTINATION NODE

Destination node has reached its limit on number of sessions created from other nodes. The destination node might need a larger value for DINIT's "MAX # LOCAL SESSIONS FOR REMOTE NODES?" See your Network Manager.

RS04(0) - REMOTE SESSION ENVIRONMENT NOT INITIALIZED

When session is shut down at a remote node it looks to DS as though the node has been generated without Session Monitor or Session Monitor has not been initialized. Any attempt to attach to specific accounts at the node result in the RS04 error reply.

RS05(0) - WRONG PASSWORD FOR NON-SESSION ACCESS TO A REMOTE NODE

Enter the correct password. Also, non-session access is not permitted at the local node when operating under control of REMAT or by a call to DLGNS.

RS06(0) - ANOTHER PROGRAM OWNS A SESSION

DLGOF - The session that the calling program is using was created by another program.

DLGON, DLGNS - A session which the calling program is willing to share (as specified by the IORIDE parameter) exists at the requested node.

RS06(1) - ANOTHER PROGRAM OWNS A SESSION

DLGON - A session which the calling program is willing to share exists at the requested node, and the remote node's sub-level is 0.

RS07(0) - LOGON OR LOGOFF ATTEMPTED TO LOCAL NODE

No REMAT logons are permitted to the local node.

RS09(0) - RSM received an illegal request.

RTE FILE ERROR nnn

(RMOTE) The reported FMP error occurred during a file access.

SCxx Error Codes

SCXX(0) - Refer to RTE SCXX error descriptions.

SC01(1) - MISSING SCHEDULING PARAMETER

(DEXEC) Not enough scheduling parameters have been provided in the DEXEC call.

SC01(2) - MISSING SCHEDULING PARAMETER

(EXECM) The scheduling EXEC call has made it to the DS monitor EXECM where an error has been detected. Provide additional parameters and try again.

SC01(3) - MISSING SCHEDULING PARAMETER

(EXECW) The scheduling EXEC call has made it to the DS monitor EXECW where an error has been detected. Provide additional parameters and try again.

SC02(2) - ILLEGAL SCHEDULING PARAMETER

(EXECM) The scheduling EXEC call has made it to the DS monitor EXECM where one of the provided parameters has been found to be in error. Correct the error and try again. (The improper parameter is related to the STRING buffer specification.)

SC02(3) - ILLEGAL SCHEDULING PARAMETER

(EXECW) The scheduling EXEC call has made it to the DS monitor EXECW where one of the provided parameters has been found to be in error. Correct the error and try again. (Internal: The request length is incorrect.)

SC05(1) - PROGRAM NOT LOADED

(DEXEC) DEXEC has found the caller's program name specification to be in error. Correct the program name in the schedule call. If a clone is requested in a DEXEC(9) or DEXEC(23) call or 'RW' from REMAT, this error can occur if the type 6 file (if applicable) could not be opened, or there are no blank ID segments, or there are already 26 clones (.A thru .Z) for this program name.

SC05(2) - PROGRAM NOT LOADED

(EXECM) The DS monitor EXECM has issued the schedule request and found the target program to be non existent. Load the target program or correct the program name in the schedule call.

SC05(3) - PROGRAM NOT LOADED

(EXECW) The DS monitor EXECW has issued the schedule request and found the target program to be non existent. Load the target program or correct the program name in the schedule call

SMxx Error Codes

SM01(0) - FMP ERROR ON ACCOUNT FILE ACCESS

Account file bad or file not found. Contact System Manager.

SM02(0) - REQUIRED CLASS NUMBER NOT AVAILABLE

Possible generation error, LOGON requires a class number for its own use.

SM03(0) - SESSION LIMIT EXCEEDED

Allocated SAM block for Session Control Blocks is full, wait for someone to logoff.

SM04(0) - NO SUCH USER

Invalid USER.GROUP name. Enter a valid account name.

SM05(0) - ILLEGAL ACCESS (PASSWORD)

Invalid password supplied. Enter correct password.

SM06(0) - CONFLICT IN DEFINITION OF SESSION LU XX

This is an informational error message. The terminal being logged onto has a configuration table entry which is a duplicate of an entry in the user's account file. The user's account file definition is used.

SM07(0) - NO ROOM FOR SESSION CONTROL BLOCK

During Session Monitor initialization a SAM block is allocated to contain all Session Control Blocks. Receiving this error indicates this SAM block is filled and the maximum number of Sessions are now active. Some other Session must terminate before a successful access of the Remote Session Node can be accomplished.

SM08(0) - DUPLICATE SESSION IDENTIFIER

Logon attempted for an existing session.

SM09(0) - SESSION SWITCH TABLE OVERFLOW

The user account was defined with the total number of required SST entries greater than 70. Contact the System Manager.

SM10(0) - NO FREE ID SEGMENTS OR FMGR NOT FOUND

There are not enough free ID Segments to create and schedule a copy of file manager for this session. Consult your System Manager.

SM11(0) - FMP ERROR ON DISC MOUNT ATTEMPT

Attempt to mount a private or group cartridge belonging to this session failed.

SM12(0) - ACCOUNT FILE CORRUPT

Internal structure of account file is incorrect. Contact the System Manager.

SM13(0) - SESSION SHUTDOWN

The system is not ready for use. Try again later. When session is shutdown, only the system console is available for operator control.

TIMEOUT: NO REPLY FROM REMOTE

(RMOTE) The HP 3000 did not respond to the last command: try again. If timeouts occur often, the master time-out value needs to be increased. See your Network Manager.

TR STACK OVERFLOW

(RMOTE) The transfer stack is more than seven levels deep.

UNINITIALIZED @ READ

(RMOTE) Local and/or remote ID sequences do not match the HP 3000. Reinitialize DS or use DSMOD to change them.

WARNING--ILLEGAL OPTION

(RMOTE) Printed only if severity = 0. "SP" specified with input from RTE LU or an RTE file in non-spooled format. The option is ignored and processing continues.

WARNING: RMOTE BUFFER TOO SMALL!

(RMOTE) Printed only if severity = 0. RMOTE has insufficient buffer space at the end of the partition to hold some of the messages from the HP 3000. Size up RMOTE and establish a new virtual session.

DS Error: (Unexpected Errors)

Unexpected errors are indicated by the following:

```
DS ERROR: PROG=xxxxx STREAM=yyyyyy SEQ # =ddddd
          P=pppppp A=aaaaaa B=bbbbbb
```

where xxxxx = name of program reporting the error,

yyyyyy = stream code,

ddddd = sequence number of the request or reply being processed when the error occurred,

pppppp = P-register address where the program detected the error (you'll need listings),

aaaaaa = contents of the A-register when the error occurred, and

bbbbbb = contents of the B-register when the error occurred.

Other miscellaneous unexpected errors are shown below:

```
DS ERROR:DS02(0) -- REPLY FLUSHED
STREAM =  000005 ORG NODE=     4 DEST NODE     5
TIME: DAY   221  13: 22: 18
```

A link timeout occurred while trying to send a slave reply to a waiting master.

```
DS ERROR: TCB NOT FOUND, POSSIBLE TIMEOUT
STREAM =  000001 ORG NODE=     4 DEST NODE     5
TIME: DAY   221  10:  0:  0
```

A reply arrived at a node for which there was no waiting master. The master may have timed out previously.

```
DS ERROR: SLAVE TCB NOT FOUND, POSSIBLE TIMEOUT
STREAM =  000001 ORG NODE=     4 DEST NODE     5
TIME: DAY   221  10:  0:  0
```

A slave program attempted to send a reply but the associated request's TCB could not be found. The slave TCB probably timed out.

In the following example, the I/O status is obtained from bits 0-7 in EQT word 5, set by the driver at the completion of the I/O request:

```
DS ERROR: COMM. READ ERROR, LU =  5,I/O STATUS =  10
TIME: DAY   221   8: 22: 50
```

A possible cause of the following error is the corruption of the table areas in SSGA (RES) and in SAM.

```
DS ERROR: PROG=GRPM  STREAM =    2 SEQ#= 12343
P=  24062 A=  32724 B=   2461
```

RTMLG Generation Errors

The following table contains a list of RTMLG error messages, their interpretation, and the appropriate action. Also refer to the DS/1000-IV Network Manager's Manual for other possible errors.

RTMLG GENERATION ERROR MESSAGES

MESSAGE	DESCRIPTION
ERR-AD	Undefined external from SYS MOD relocation or invalid entry in INT Table definition. Start RTMLG over again, or enter valid INT entry.
ERR-CE	Close error.
ERR-CH	Invalid channel number in EQT or in INT (>77B). Enter correct channel number.
ERR-DR	Invalid driver name: not DVxxx or driver not relocated as SYS.MOD. Enter correct EQT entry.
ERR-DU	Duplicate program name (from CHANGE PRAMS response). Repeat CHANGE PRAMS response.
ERR-EQ	Invalid EQT number in INT Table: >77B, =0, >EQT entry. Enter valid EQT response in INT Table.
ERR-IN	Execution interval error (in CHANGE PRAMS response). Enter correct CHANGE PRAMS response.
ERR-LU	Invalid Device Reference Number: >99 for DRN, >99 for subchannel, no corresponding EQT entry, no DRT entries. Enter correct DRT entry.
ERR-NM	No memory.
ERR-ON	Invalid ON,RTMLG parameters. Start RTMLG over again.
ERR-OO	No output file.
ERR-NA	Invalid interrupt mnemonic - only following allowed: EQT, PRG, ENT, ABS. Enter correct INT entry.

MESSAGE	DESCRIPTION
ERR-PA	Invalid parameter or parameter name error in CHANGE PRAMS; first character must be =<132 (octal) and =>4 (octal). Octal 47 to octal 55 not allowed. Input correct parameter.
ERR-PD	Partition already defined. Enter next partition definition.
ERR-PR	Parameter priority error >5-digit decimal number in CHANGE PRAMS priority. Enter correct CHANGES PRAMS entry.
ERR-PS	Partition size error. Partition bigger than remaining memory. Enter next partition definition (must be less than previous one).
ERR-PT	Partition definition error. Partition greater than maximum allowed. Enter next partition definition.
ERR-SO	System overflow loader symbol table and short fixup table overflow (used for EQT extensions). Redefine all EQT entries.
ERR-TB	Symbol table/ID segment overflow, or entered programs exceed ID segments, or no ID segments left for start-up program. Start RTMLG over again.

Program Download Error Messages

When an error occurs on program download and if the message logging option has been selected, the following message will be displayed on the logging LU.

```
     DOWNLOAD OF FILE:<file name>::-<lu>:<type>

                                     AM
     AT DAY nnn <hr>,<min>:<sec>        error message
                                     PM
```

ERROR MESSAGE	REASON
FAILED:CKSM ERR	'PROGL' calculates a checksum on each record read before sending it to the remote. If the file is not absolute binary, or has been corrupted, the check will fail and this message will be printed.
FAILED: CLASS ER	A class-I/O error occurred on the attempt to transmit a record (not a transmission error). This would be a catastrophic error, most likely an indication of corrupted tables in RES or the RTE class table.
FAILED:LINE ERR	An error occurred in trying to re-queue a record back onto the transmission line after a transmission error has ocurred. 'PROGL' simply delays transmission on all lines for 200 ms. if an error occurs on any line. This would also be a catastrophic error.
WAS ABORTED	A new download request arrived before completing the previous one. This may occur if the operator restarts the CBL sequence, or (if the remote program load feature is enabled) a power failure occurs during a down-load. RPL restarts on power recovery. This error message is to notify the operator that a re-start has occurred. No operator action is required. The message of a new down-load prints immediately afterward.

If any file-open or file-read error occurs, the message:

```
                 OPEN
     /PROGL:FILE      ERROR -nnnnn
                 READ
```

will be printed, followed by:

```
                                                     AM
     DOWNLOAD OF FILE <namr> AT DAY nnn, <hr>:<min>     HAS FAILED.
                                                     PM
```

RTE-L Mini-Loader Errors

The messages shown below are printed if, and only if, a positive, legal print LU number is given for the <error LU> parameter. The code shown in parenthesis with each call is for the first return parameter. Unless otherwise specified, other return parameters are undefined. For example, if PRAMS (1) = 0 (no error condition) then PRAMS(2) thru PRAMS(5) are undefined.

```
DSxx ERROR. RETRYING
       or
DS ERROR DSxx(n), REPORTING NODE yyyyy
```

(-4) xx= a two-digit code in the range of 00 thru 09
n= a one-digit qualifier code
yyyyy= node reporting the error

Refer to the particular DS error in this appendix.

DUPLICATE PROGRAM NAME

(2) A program already exists by the same name.

FILE OPEN ERROR -(FMP error code)

(-2) A FMP error is returned.

FILE READ ERROR-(FMP error code)

(-5) FMP 'read' error.

ILLEGAL BG LOAD ATTEMPT

(6) An attempt was made to load a program in backround, but load/swap modules included in RTE-L gen.

IMPROPER 'RUN' STRING, OR NONE GIVEN
or
MUST SPECIFY REMOTE NODE!

(-1) Improper 'run' string given, required parameters not supplied.

NO BLANK ID SEGMENTS
(3) No blank ID segments available.

PRGM NOT RELOCATED W/CORRECT SNAP

(5) Program not relocated with correct snapshot file. Checksum in file does not match system checksum at RTE-L

PROGRAM <AAAAA> DOWN-LOAD COMPLETE

(0) No error. On function code 0 requests, RTE-L memory address where slave is placing ID segment is returned in p1 of the reply. <AAAAA> is the name the program has at the RTE-L, i.e., the first five characters of the file name.

REMOVE <program> [MR]

(4) Where <program> is the 5-character program name and [MR] are two ASCII characters which are printed only if the program's down-load has completed (i.e., is memory-resident). This error occurs when two programs attempt to occupy the same memory locations. If scheduled with wait from another program, then the name of the program occupying the same memory space as the one you wanted, is returned in P2-P4 of the reply. P5 will contain two ASCII characters, identifying whether the program is being loaded (two blanks) or has been completely transferred into memory (MR). If it has not been completely transferred, there is the possibility that its master aborted prematurely or the transfer is on-going.

THAT'S NOT AN RTE-L LOAD-FILE!

(-3) File specified was not a real RTE-L program load file (checksum word of ID segment did not match sum of words 1 thru 30).

UNREC FN CODE

(1) Unrecognized function code (RPRTL is not the master program).

Multidrop DS/1000-IV Errors

INVALID SECURITY CODE

Security code specified in the FCL7 runstring is incorrect. It must be 29150.

LU INVALID

The logical unit number for the first or second argument is not within the range of the device reference table (DVT).

DRIVER IS NOT TYPE 7

The EQT, which corresponds to the first argument, is not attached to a type 7 driver.

EQT IS NOT AVAILABLE

The first LU EQT is not available because it has been linked by the Multipoint driver. The EQT is actually initialized.

DRIVER IS NOT TYPE 65

The DS node LU does not point to an EQT which indicates a type 65 driver attachment.

DSLIN Errors

PRIMARY CONNECT TIMED OUT. CONNECTING AS SECONDARY STATION ON LU XX

(DSLIN) Initialization request has timed out without a reply from the HP 3000. The board is then connected as a secondary station.

BREAK FLAG SET

(DSLIN) During a request for initialization, DSLIN has received a break request.

LINE IS UP BUT 3000 IS NOT REPLYING

(DSLIN) Initialization request has not resulted in a reply within 50 seconds. Board is connected as a secondary station. Check for initialization at HP 3000.

DS/1000 HAS NOT BEEN INITIALIZED

(DSLIN) When trying to initialize an HP 3000 LU, DSLIN finds DS uninitialized. Run DINIT.

DSLIN IS ONLY USED FOR BISYNC LINKS

(DSLIN) The node is initialized for HSI communication. If you want to use PSI, make sure the PSI versions of QUEX and QUEZ are loaded then rerun DINIT and specify PSI 3000 LUs.

DSLIN cannot initialize an X.25 pool LU.

LU nn IS NOT IN THE 3000 LU TABLE

(DSLIN) The LU specified is not an HP 3000 LU. Specify an HP 3000 LU or reinitialize DS and specify this LU as an HP 3000 LU.

DSLIN cannot initialize an X.25 pool LU.

LU nn HAS BEEN INITIALIZED WITH BUFFER SIZE xxxx

(DSLIN) Some other copy of DSLIN is initializing this LU or a program has it locked.

CANNOT LOCK LU nn

(DSLIN) DSLIN is unable to lock the LU for initialization.

THE BOARD ON LU nn DOES NOT CONTAIN BISYNC FIRMWARE! (BOARD TYPE = X)

(DSLIN) Board contains wrong firmware. Check for an HDLC board associated with this LU.

I/O ERROR at iiiii. STATUS = vvvvv

(DSLIN) Status returned from the driver indicates an error.

```
iiiii  =  INITIALIZE BOARD
          GET PARAMETERS
          PRIMARY CONNECT
          SECONDARY CONNECT
          AWAITING REPLY
          DISCONNECT

vvvvv  =  LINE FAILURE
          TIMEOUT
          OVERRUN
          REMOTE BUSY
          UNINITIALIZE
          WRONG MODE
          ILLEGAL REQUEST
          CARD FAILURE
```

SESSIONS STILL OPEN ON LU xxx. THE LU WAS NOT CLOSED.

(DSLIN) There were other users on the PSI BISYNC line when you tried to close it, so DSLIN terminated without closing the line. Try running DSLIN again later.

PLEASE ENTER THE LU.

(DSLIN) You entered a carriage return in response to the question, "HP 3000 TO INITIALIZE:." Enter the lu of the PSI BISYNC line that you want to initialize.

Index

READER COMMENT SHEET

DS/1000-IV
User's Manual

91750-90002 **March 1985**

Update No. ________
(If Applicable)

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications. Please use additional pages if necessary.

FROM:

Name	________________
Company	________________
Address	________________
	________________
	________________

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070 CUPERTINO,CALIFORNIA

POSTAGE WILL BE PAID BY ADDRESSEE

Hewlett-Packard
Information Networks Division
19420 Homestead Road
Cupertino, California 95014
Atten: Customer Information Products Dept.

FOLD FOLD