TOPS–20 DECnet–20 Programmer's Guide and Operations Manual

AA-5091B-TM

January 1980

This manual describes DECnet–20, a product that together with TOPS–20 provides the DECSYSTEM–20 family of computers with a communications interface to DIGITAL's corporate network, DECnet.

This manual supersedes *TOPS–20 DECnet–20 Programmer's Guide and Operations Manual*, Order Number AA-5091A-TM.

OPERATING SYSTEM: TOPS–20 Version 4

SOFTWARE: DECnet–20 Version 2

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

NORTHEAST/MID-ATLANTIC REGION

Technical Documentation Center
Cotton Road
Nashua, NH 03060
Telephone: (800) 258-1710
New Hampshire residents: (603) 884-6660

CENTRAL REGION

Technical Documentation Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone: (312) 640-5612

WESTERN REGION

Technical Documentation Center
2525 Augustine Drive
Santa Clara, California 95051
Telephone: (408) 984-0200

digital equipment corporation • marlboro, massachusetts

First Printing, September 1978
Revision, January 1980

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DECnet	IAS
DECUS	DECsystem-10	MASSBUS
Digital Logo	DECSYSTEM-20	PDT
PDP	DECwriter	RSTS
UNIBUS	DIBOL	RSX
VAX	EduSystem	VMS
		VT

CONTENTS

FIGURES

Page

TABLES

REVISION HISTORY

Enhancements and Changes

This revision reflects the software as of Version 2 of DECnet-20. The major enhancements and changes to the software include:

- Increase in the maximum number of physical links supported for DECSYSTEMs-2040/2050/2060 to 8.
- Addition of the network file transfer utility (all models).
- Support of up to 128K of memory in increments of 32K for the DN20 communications front end.
- Provision for on-site generation of DECnet-20 for DECSYSTEMs-2040/2050/2060.
- Provision for support of the KDP line controller (4-line synchronous DMA device) and the DN200 (remote job entry station).
- Extension of error and event logging capabilities to include hardware detected errors and CHK11 diagnostic messages.
- Inclusion of additional features for the nonprivileged terminal user.
- Addition of new commands and tools to provide a more maintainable product than Version 1.

The contents of this revision supersede the entire contents of the DECnet-20 Version 1 manual.

NOTES

Chapters 1 through 5 in both the Version 1 and the Version 2 manuals cover the same topics.

Chapter 6 in the Version 2 manual describes "Running DECnet-20," which was described in Chapter 7 of the Version 1 manual.

Chapters 7 through 10 in the Version 2 manual are new chapters and therefore have no counterparts in the Version 1 manual.

All changes from TOPS-20 Version 3A, DECnet-20 Version 1 are marked with change bars. Chapters 7 through 10 are marked with change bars in the Table of Contents only.

PREFACE

This manual includes information about configuring, installing, programming, and operating DECnet-20. As such, it is directed to the following audience:

- The application programmer responsible for writing the programs that will be exchanging data with programs on other systems in the network. This person should be an experienced MACRO programmer with some knowledge of network applications.

- The system operator responsible for loading and starting the network software under the TOPS-20 monitor. This person should be experienced in TOPS-20 operations and be familiar with computer networks.

- The system manager or system programmer responsible for installing the network software and generating the new system. This will probably be the person who installed the TOPS-20 monitor.

- The terminal user responsible for network tasks that do not require the enabling of capabilities. All timesharing users should be familiar with the TOPS-20 Command Language.

This manual is also intended to be of use to field and corporate software service personnel engaged in the installation and support of the product.

The manual is organized into five parts as follows:

Part I, the Introduction, contains an overview of DECnet-20 and a discussion of system concepts, network options and network facilities.

Part II, the Programmer's Guide, contains the programming facilities that the MACRO programmer must use in order to perform the following network functions:

- Establishing a network connection using network-related monitor functions.

- Transmitting and receiving both data and interrupt messages over a network link using standard TOPS-20 I/O monitor calls.

- Terminating a network connection using network-related and standard monitor calls.

- Controlling the network using network-related monitor functions.

Part III, Operator's Guide, contains system loading procedures, DECnet-20 operator commands, recovery procedures, control procedures, and network-related error messages. These procedures include a number of examples printed in two colors for clarity. Terminal input is shown in red; system output is shown in black.

Part IV, Terminal User's Guide, contains TOPS-20 commands available to nonprivileged users. All network-related TOPS-20 commands, including user-interaction with the file transfer utility (NFT), are described. Examples are given in the form previously explained.

Part V, Generating DECnet-20, discusses the concepts, features, terms, and commands that are related to on-site configuration of a DECnet-20 system as well as the procedures for installing, verifying, and certifying the DECnet-20 system.

Part VI contains the appendixes.

The following are suggested guidelines for using this manual:

- Application programmers should read Parts I and II, and the appendixes.
- System operators should read Parts I and III, and the appendixes.
- Terminal users should read Parts I and IV.
- System managers and system programmers should read the entire manual.

DECnet-20 runs under the TOPS-20 monitor on all the DECSYSTEM-20 models. Both the application programmer's and the nonprivileged user's interface to DECnet-20 are the same regardless of the system model. However, there are differences in the installation and operation procedures between the DECSYSTEM-2020 and the other systems. Therefore, this manual uses the term DECSYSTEM-20 only when referring to all models. Specific systems are referred to by their model designations, such as DECSYSTEM-2020 or DECSYSTEM-2040, 2050, 2060.

The following documents are either referenced in this manual or may prove useful in implementing DECnet-20 facilities.

TOPS-20 Monitor Calls User's Guide	DEC-20-OMUGA-A-D
TOPS-20 Monitor Calls Reference Manual	AA-4166D-TM
TOPS-20 Software Installation Guide	AA-4195G-TM
TOPS-20 Operator's Guide	AA-4176D-TM
TOPS-20 Operator's Command Language Reference Manual	AA-H600A-TM
TOPS-20 Commands Reference Manual	AA-5115B-TM
TOPS-20 User's Guide	AA-4179C-TM

TOPS-10 and TOPS-20 Batch Reference Manual	AA-H374A-TK
TOPS-10 and TOPS-20 SYSERR Manual	AA-D533A-TK
TOPS-20 UETP Procedures/Reference Manual	AA-D606B-TM
TOPS-20 System Manager's Guide	AA-4169F-TM

PART I

INTRODUCTION

CHAPTER 1

SYSTEM OVERVIEW

1.1 DECnet

DECnet is the name given to the set of software products that extend the capabilities of various DIGITAL operating systems so that these systems can be interconnected to form computer networks.

DECnet is based on sets of rules (protocols) known collectively as DIGITAL Network Architecture, or DNA. These protocols govern the execution of major network chores, such as transferring messages and data over physical lines, using error detection and retransmission to guarantee the integrity of data transfer, multiplexing multiple logical messages over a single physical connection, and controlling which nodes are allowed to communicate and when they can do so.

Each operating system that supports DECnet implements some subset of the complete DNA. The subset of DNA that runs under TOPS-20 is called DECnet-20.

1.2 DECnet-20 OPTIONS

DECnet-20 task-to-task capability is required on all DECSYSTEM-20 systems before an option can be added. Currently, DECSYSTEMs-2040/50/60 with a 128K DN20 communications front end support up to eight task-to-task lines. The DECSYSTEM-2020 supports two task-to-task lines. Task-to-task communication requires a minimum of one synchronous line. In addition to multiline support, the capabilities of DECnet-20 include the Network File Transfer (NFT) utility.

One additional option is available for DECnet on the 2040/50/60 systems:

- RJE-20 Version 1. This option includes software for the DN200 remote batch entry station and provides the facility for the DN200 software to be down-line loaded, diagnosed, and maintained remotely by the host system.

A DECnet site may choose to divide the use of its lines between task-to-task and RJE.

No options are available for the DECSYSTEM-2020.

1.3 DECnet-20 STRUCTURE

DECnet-20 is a software product that extends the facilities of TOPS-20 so that tasks running on any member of the DECSYSTEM-20 family of computers can communicate with other tasks in a computer network.

When DECnet-20 is running on a DECSYSTEM-2040, 2050, or 2060, the network software resides partly in the main processor with the TOPS-20 monitor and partly in a separate processor (the DN20) designed to handle network communications functions. This latter processor is referred to as the communications front end to distinguish it from the console front end that controls the local command terminals and unit record peripherals. The main processor communicates with either front end through a DTE hardware interface. Figure 1-1 shows a DECSYSTEM-2040, 2050, 2060, a DTE, and a DN20 communications front end. The major protocol levels are also shown.

When DECnet-20 is running on a DECSYSTEM-2020, the network software resides entirely in the main processor; there is no separate communications front end. Figure 1-2 shows a DECSYSTEM-2020 and the location of the major protocols.

DECnet-20 has a multilayer structure; each layer, or level, corresponds to one or more protocols in the DIGITAL Network Architecture (DNA).

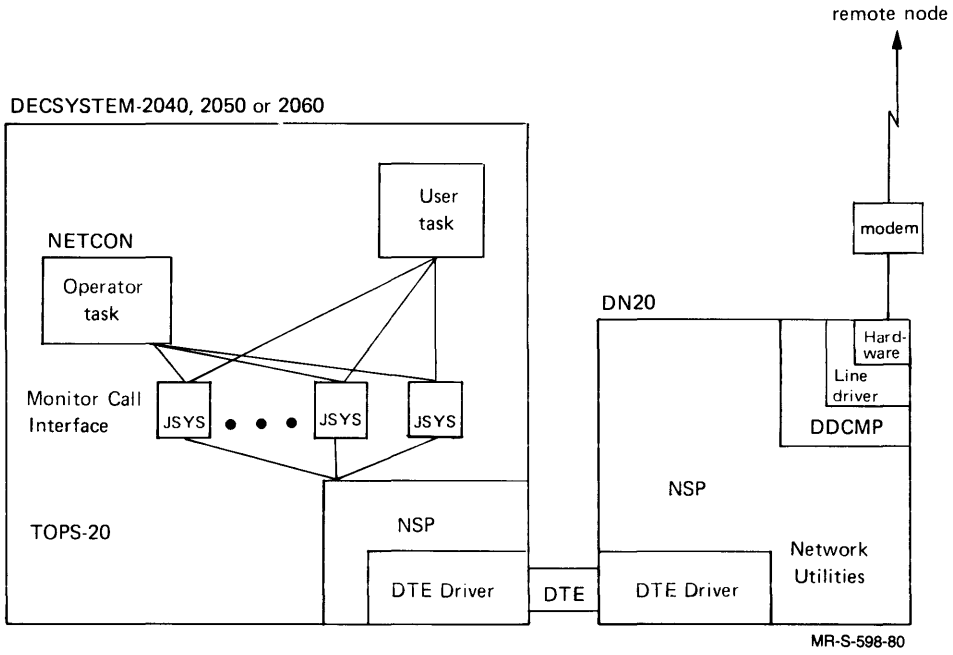


Figure 1-1 DECnet-20 on a DECSYSTEM-2040, 2050, 2060

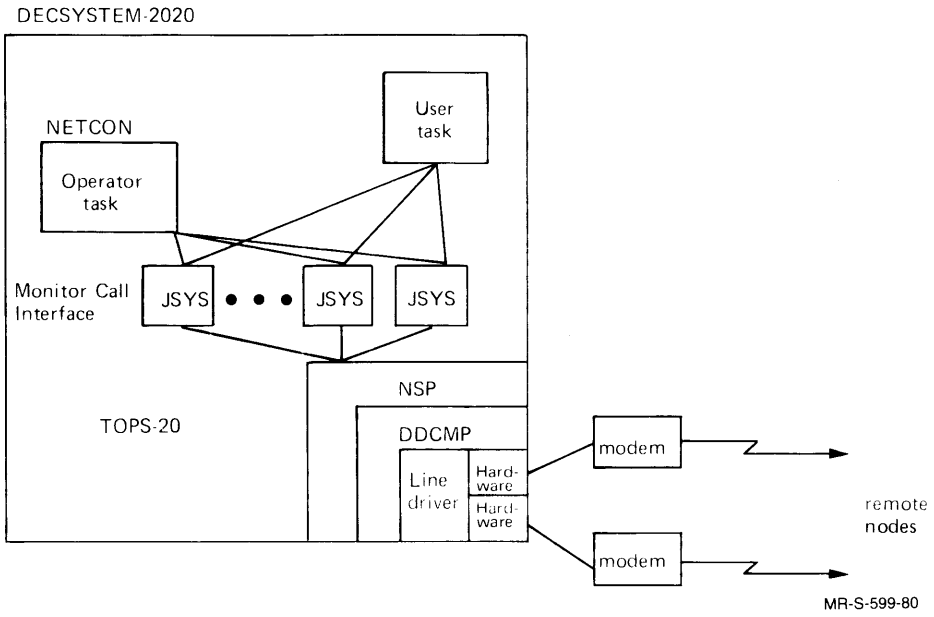


Figure 1-2 DECnet-20 on a DECSYSTEM-2020

1.3.1 Hardware Level

The lowest level of DNA protocol used in DECnet-20 is the hardware level. This level is implemented by line drivers that are tailored to the types and characteristics of the physical lines, modems, and modem control interfaces. The hardware level controls transmission techniques (synchronous or asynchronous, serial or parallel, full- or half-duplex), character synchronization, and modem operation. In Figures 1-1 and 1-2, this level is represented by the line drivers.

1.3.2 Physical Link Control Level

The next higher level of DNA protocol is the physical link control level. This level is implemented by the Digital Data Communications Message Protocol (DDCMP). DDCMP maintains control over the hardware level by using a 16-bit Cyclic Redundancy Check (CRC-16) to detect errors, retransmitting to correct errors, and numbering messages to ensure sequential data transfer. The next higher level of protocol, therefore, can be assured of a sequential and error-free data path.

In Figure 1-1, the physical link control level is represented by the DDCMP layer in the DN20. The two DTE drivers, each in their respective CPUs, perform DDCMP-like functions to control the flow of NSP messages between the host computer and its communications front end.

In Figure 1-2, the DECSYSTEM-2020 does not have a communications front end and therefore the DDCMP level is implemented entirely in the host computer.

1.3.3 Logical Link Control Level

The next higher level of DNA protocol used in DECnet-20 is the logical link control level as defined by the Network Services Protocol (NSP). NSP controls the flow of data between tasks in a network. This involves creating and terminating logical links (communications paths), multiplexing logical links into physical links, segmenting and collecting long messages, and ensuring end-to-end delivery of data in proper sequence. Logical links are described in detail in Chapter 2.

When DECnet-20 is running on a DECSYSTEM-2040, 2050, or 2060, the NSP protocol is implemented both in the TOPS-20 monitor running in the main processor and in the communications front end. In Figure 1-1, this level is represented by the NSP code in the main processor and in the DN20. When DECnet-20 is running on a DECSYSTEM-2020, the NSP protocol is implemented entirely in the main processor as shown in Figure 1-2.

NSP in the main processor provides the TOPS-20 user with programmable access to the DECnet environment via the TOPS-20 file system monitor calls. NSP also resolves the byte-size differences between the 36-bit word structure of the main processor and the 8-bit byte structure required by the network. Additional NSP protocol and network utilities in the communications front end (or in the main processor in the case of the DECSYSTEM-2020) perform the bulk of the NSP control functions.

1.3.4 User or Application Level

DECnet and its first three levels of protocol (hardware, DDCMP, and NSP) provide for the exchange of data between network tasks. How these tasks interpret this data involves a user level of protocol. In both Figures 1-1 and 1-2, this user level is represented by the user tasks running on the DECSYSTEM-20. The user level of protocol is responsible for interpreting and acting on the data exchanged.

1.4 DECnet-20 CAPABILITIES

DECnet-20 provides the TOPS-20 user with basic network task-to-task capabilities. That is, local system or user tasks written in MACRO-20 can exchange information with system or user tasks running in one or more adjacent nodes in a network.

The local task accomplishes this by using the TOPS-20 file system monitor calls to open, read and write information, and close files using a pseudo-device representing the network.

Below this user level, and transparent to the user, the network protocol takes over. NSP software running in the main processor (and in the communications front end when part of the system) manages the actual transfer of data over a logical link. User data is first reformatted into network-compatible segments and then transferred over the logical link. NSP also generates the appropriate control messages to open and close network connections.

A special network control task called NETCON is started at system initialization and runs under the system program SYSJOB. NETCON accepts operator commands (and command files) through the operator interface program ORION and the operator command parser OPR. Operator commands to NETCON will:

- Down-line load network software into the communications front end of a DECSYSTEM-2040, 2050, 2060.
- Load network software into the synchronous line controller on the DECSYSTEM-2020.
- Up-line dump the communications front end of a DECSYSTEM-2040, 2050, 2060 into a TOPS-20 file.
- Dump the synchronous line controller on the DECSYSTEM-2020 into a TOPS-20 file.
- Log pertinent network load, dump, and line status information into the TOPS-20 SYSERR file.
- Display line statistics for any line on any node in the network.
- Start and stop DECnet-20 lines on all DECSYSTEM-20 models.
- Start a line in cable-loopback or controller-loopback mode (DECSYSTEM-2020).
- Perform a loopback test on a designated line (DECSYSTEM-2040/50/60).

Operator commands are described in Chapter 6.

1.5 DECnet-20 PROGRAMMER INTERFACE

Using DECnet-20, a MACRO programmer can transfer data between user storage and the network in much the same way that data is transferred between user storage and files on a peripheral device. The peripheral device, in this case, is the network; the file is a logical link. File system monitor calls, such as GTJFN, OPENF, and CLOSF, control the making and breaking of network connections. Input/output monitor calls, such as SIN, SINR, BIN, SIBE, SOUT, SOUTR, and BOUT, handle the movement of messages and data across the network logical links. Several network-specific functions have been added to the MTOPR monitor call to provide the user with some form of logical link management. These new functions are described in subsequent chapters.

1.6 DECnet-20 OPERATOR INTERFACE

Operator commands to the network control program, NETCON, fall into two classes: generic DECnet commands and TOPS-20-specific DECnet commands.

Generic DECnet commands function the same on all DECnet computer systems. These are the LOAD NODE and DUMP NODE commands to down-line load and up-line dump the communications front end, the SHOW QUEUE command to check the status of any queued NETCON requests, the SHOW COUNTS command to display line counter information, the SET EXECUTOR command to specify which node is to process subsequent commands, the SHOW STATUS commands, the SET STATE command, and the LOOP LINE command to loop a standard test message a specific number of times.

TOPS-20-specific DECnet commands are extensions to the generic DECnet commands necessary for the implementation of DECnet under TOPS-20. On the DECSYSTEM-2040, 2050, and 2060, the SET NODE command establishes operating characteristics and the SET SECONDARY (TERTIARY)-LOAD-FILE command specifies the secondary (tertiary) bootstrap programs to be used for down-line loading the communications front end. On the DECSYSTEM-2020, the LOAD LINE and DUMP LINE commands load and dump the integral line controller. On all models, the INITIATE LOGGING, TERMINATE LOGGING, and SET LOGGING-INTERVAL commands control the recording of DECnet-20 line statistics.

1.7 DECnet-20 TERMINAL USER'S INTERFACE

As a terminal user (without OPERATOR or WHEEL privileges), you use the TOPS-20 Command Language (EXEC) to communicate with the system. The TOPS-20 User's Guide and the TOPS-20 Commands Reference Manual describe the TOPS-20 Command Language in detail.

Part IV of this manual describes EXEC commands that are specific to DECnet-20. It also includes Network File Transfer (NFT). NFT may be used by the nonprivileged user. NFT gains access to the files and devices on other nodes through the Data Access Protocol (DAP) of the DNA, but this interaction is transparent to you. Your concern is the function and format of the NFT commands.

CHAPTER 2

CONCEPTS AND FACILITIES

2.1 SYSTEM CONCEPTS

The basic concept of DECnet-20 is that the network is to be treated as just another TOPS-20 input/output device. TOPS-20 programs written to communicate with other tasks in the network use TOPS-20 file system monitor calls to request network functions.

This concept is represented graphically in Figure 2-1. The local NSP code interfaces with the TOPS-20 monitor to provide the user with network access using many of the same monitor calls that are used to access local peripherals. Figure 2-1 shows this local NSP code residing in the main processor of a DECSYSTEM-2020. In a DECSYSTEM-2040, 2050, 2060, the NSP code resides in both the main processor and in the communications front end. The communications front end, if present, is transparent to both local and remote user tasks.

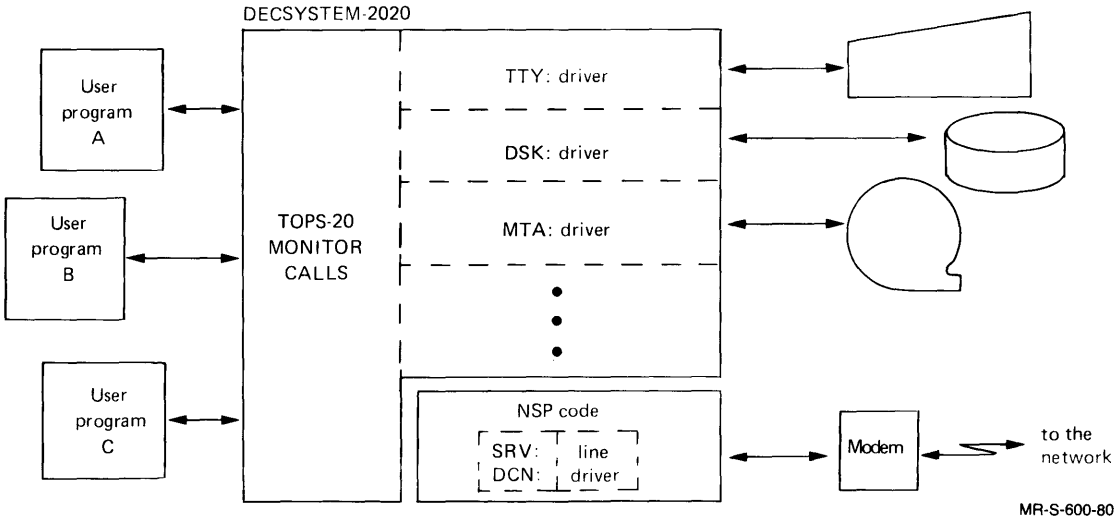


Figure 2-1 The Network as an I/O Device

To initiate a dialogue with a target task, a source task identifies the target task as a file on the network device DCN:. The source task then proceeds to open the network file, write to it, read from it, and eventually close it, very much as if it were a file residing on a local auxiliary storage facility.

To declare itself as being available for a connection by a source task, a target task identifies itself as a file on the network device SRV:. When the SRV: file is opened, any incoming connect request addressed to this target task is forwarded to it by DECnet-20. The request may be accepted or rejected. If the request is accepted, the connection is made and data can be exchanged between the two tasks via reads and writes to the file.

Other network concepts that will be used in subsequent chapters are physical and logical links, network job file numbers, network task identification, and network node identification. These concepts are introduced in the following subsections.

2.1.1 Physical and Logical Links

Physical and logical links are the basic elements of communications on a network. Physical links connect network nodes and logical links connect network tasks.

A physical link connects two network nodes and can take one of several forms. In Figure 2-2, the physical link between nodes ABLE and ABLER is the DTE interface, a hardware device between a DECSYSTEM-2040, 2050, 2060 and its communications front end. The physical link between nodes ABLER and BAKER or between nodes BAKER and CHARLY can be a relatively permanent connection such as a leased or private telephone line or cable. It can also be a temporary connection such as a satellite link or radio circuit.

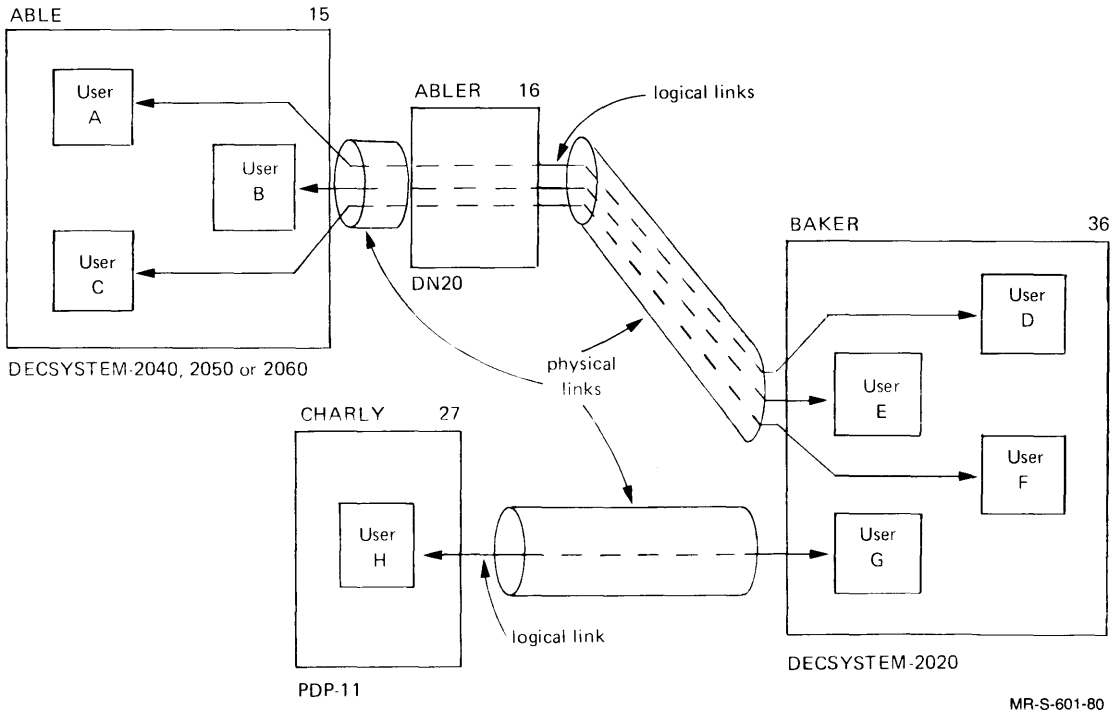


Figure 2-2 Logical and Physical Links

DECnet-20, running on a DECSYSTEM-2020, supports two physical links to the network as shown at node BAKER in Figure 2-2. DECnet-20, running on a DECSYSTEM-2040, 2050, 2060, supports up to eight physical links. One physical link to the network is shown at node ABLER in Figure 2-2.

A physical link can support one or more logical links.

A logical link connects two network tasks. A logical link usually shares a physical link with other logical links. A logical link can span more than one physical link as shown from node ABLE to node BAKER in Figure 2-2.

The simultaneous sharing of a physical link by multiple logical links is referred to as multiplexing. Among the functions of NSP in DECnet-20 are the mixing of outgoing logical link data from several users (multiplexing) for transmission over a physical link and the separating of incoming logical link data (demultiplexing) for distribution to individual users.

In Figure 2-2, the logical links between users A and D, B and E, and C and F, are multiplexed and span two physical links: ABLE to ABLER and ABLER to BAKER. The logical link between users G and H is the only data path on the physical link from BAKER to CHARLY and therefore no multiplexing is necessary.

2.1.2 Network Job File Number

The network job file number in DECnet-20 is analogous to the job file number (JFN) assigned to all input and output files being processed by TOPS-20 tasks. In the TOPS-20 file system, a JFN is associated with a file specification. In DECnet-20, where the network is treated as an input/output device, a network JFN is associated with a specification for a logical link, the network equivalent for a file.

The same monitor call (GTJFN) is used to obtain either a network or a file JFN. The information passed to the GTJFN monitor call for a network JFN is similar in format to that supplied for a file JFN. The format and usage of logical link specifications are explained in Chapter 3, Establishing a Network Connection.

2.1.3 Network Task Identification

A network task is any program that is engaged in, or intends to engage in, a network dialogue. Network tasks in DECnet-20 can have two distinct identities: a generic task identification and a unique task name.

The generic task identification is used to address a network task that provides a class of service to other network tasks (for example, a network utility program). Multiple copies of such a network program can be loaded, started, and identified by class of service. Other network tasks can then request this service by specifying a connection by generic identification. This guarantees a connection to one of the available copies all of which are assumed to provide the same service. The generic task identification consists of two parts: a one-byte object type (numeric) and an optional object descriptor (alphanumeric).

Object types 1 through 127 are reserved for DECnet utilities and control programs. Object types 128 through 255 are available for user-written installation-dependent network utilities. Object type 0 is reserved for addressing tasks by their unique task name. Appendix B contains the current list of DECnet object types.

The use of object descriptors is dependent upon the implementation of NSP on the remote node. If the remote node is running a system other than TOPS-20 DECnet-20, read the DECnet manuals for the system being used. In DECnet-20, object descriptors can be used with object types 128 through 255 if so designed into the user program.

A unique task name, on the other hand, is used to address a specific network task. Only one copy of such a task can be running at any one time on any one node. If a network task is identified by task name alone, it must be addressed by the special object type 0 and a descriptor that corresponds to the unique task name.

2.1.4 Network Node Identification

Each node in a network is identified by a unique node name and a unique node number. The node name must be one to six alphanumeric characters; the node number must be a positive integer between 2 and 127, inclusive. When DECnet-20 is installed on a DECSYSTEM-2040, 2050, 2060, the main processor and the DN20 communications front end are each considered separate network nodes. Each, therefore, must have a unique name and number.

Whenever a source task requests a connection to a target task, the source task must supply the local NSP with the name of the target node. The local NSP uses this node name to generate the destination address of the NSP message that requests a connection to the target task. Sending this request for connection to a target task is the first step in establishing a logical link.

During the installation procedures for DECSYSTEM-2040/2050/2060, the main processor (host) is given an identifying name and octal number. During on-site configuration, as part of the NETGEN procedure, the DN20 and the DN200, if present, are given node names and numbers. Within the same network, all identifying names and numbers must be unique. See Chapters 9 and 10 for the procedures for assigning node names and node numbers.

2.2 NETWORK FACILITIES

The TOPS-20 programmer can implement the task-to-task function of DECnet-20 using MACRO programs or subroutines. Specifically, a subset of the TOPS-20 file system monitor calls acts as the interface between the programmer and DECnet-20. These network-related monitor calls allow the MACRO programmer to:

- Declare a network task as willing to accept connections.
- Initiate a request for a connection to another network task.

- Accept or reject a request for a connection from another network task.
- Transmit data to and receive data from another network task.
- Interrogate the status of a logical link.
- Retrieve the connect attributes of a network task.
- Exchange high priority interrupt messages (up to 16 bytes in length) with other network tasks.
- Disconnect a network connection.

The network-related monitor calls and their functions are listed in Tables 2-1, 2-2, 2-3, and 2-4. Many of these calls are also used in TOPS-20 file processing and their calling sequences are fully described in the DECSYSTEM-20 Monitor Calls Reference Manual. TOPS-20 usage information for these calls is found in the DECSYSTEM-20 Monitor Calls User's Guide. DECnet-20 usage information for all the network-related calls and the calling sequences for the network functions of MTOPR appear in the next three chapters.

Table 2-1
Monitor Calls Used in DECnet-20

Monitor Call	Network Function
GTJFN	Get a network JFN
OPENF	Open a network connection
BIN	Receive a data byte
*BOOT	Provide maintenance and utility functions for communications software (see Table 2-2)
SIN	Receive a data string
SINR	Receive a data record (message)
BOUT	Transmit a data byte
SOUT	Transmit a data string
SOUTR	Transmit a data record (message)
SIBE	Test for input buffer empty
CLOSF	Close a network connection
MTOPR	Perform device-dependent control functions (see Table 2-3)
*NODE	Set node and line characteristics (see Table 2-4)

* BOOT and some functions of NODE are privileged monitor calls used in DECnet-20 system programs. Detailed descriptions of these monitor calls can be found in the DECSYSTEM-20 Monitor Calls Reference Manual.

Table 2-2
BOOT Monitor Call Functions Used in DECnet-20

Code	Symbol	Support*	Purpose
0	.BTROM	a&b	Puts line or DTE in MOP mode and activates the front-end ROM a: DN22 b: DN20/21 or console front end
1	.BTLDS	a&b	Load secondary bootstrap
2	.BTLOD	a&b	Loads the front end a: DN22 b: DN20/21 or console front end
3	.BTDMP	b	Dump the front end
4	.BTIPR	a&b	Initiate line protocol
5	.BTTPR	a&b	Terminate line protocol
6	.BTSTS	a&b	Determine line protocol
7	.BTBEL	b	Wait for front-end doorbell
10	.BTRMP	a&b	Read MOP message
11	.BTKML	a	Load KMC11
12	.BTKMD	a	Dump KMC11
13	.BTRLC	a	Return line counters
14	.BTCLI	a&b	Convert line-id to port number
15	.BTCPN	a&b	Convert port number to line-id
16	.BTSTA	a	Set station polling status
17	.BTSSP	a	Set line startup priority
20	.BTSTP	a	Set station polling priority
21	.BTSDD	a	Send DDCMP message
22	.BTRDD	a	Receive DDCMP message
23	.BTCHN	a	Assign a software interrupt channel
24	.BTSLS	a	Set line service type

* Support indicates the hardware system on which these calls are valid: (a) for DECSYSTEM-2020 and (b) for DECSYSTEMs-2040/50/60.

Table 2-3
MTOPR Monitor Call Functions Used in DECnet-20

Code	Symbol	Purpose
24	.MOACN	Set interrupt assignments
25	.MORLS	Read link status
26	.MORHN	Read host name
27	.MORTN	Read task name
30	.MORUS	Read user identification
31	.MORPW	Read password
32	.MORAC	Read account string
33	.MORDA	Read optional data
34	.MORCN	Read object type
35	.MORIM	Read interrupt message
36	.MOSIM	Send interrupt message
37	.MOROD	Read object-descriptor
40	.MOCLZ	Reject/Close a network connection
41	.MOCC	Accept a network connection
42	.MORSS	Read segment size

Table 2-4
NODE Monitor Call Functions Used in DECnet-20

Code	Symbol	Purpose
0	.NDSLN	Set local node name
1	.NDGLN	Get local node name
2	.NDSNM	Set local node number
3	.NDGNM	Get local node number
4	.NDSLP	Set loopback port
5	.NDCLP	Clear loopback port
6	.NDFLP	Find loopback port
7	.NDSNT	Set network topology information
10	.NDGNT	Get network topology information
11	.NDSIC	Set topology change interrupt channel
12	.NDCIC	Clear topology change interrupt channel
13	.NDGVR	Get NSP version information
14	.NDGLI	Get line information
15	.NDVFY	Verify node name

PART II

PROGRAMMER'S GUIDE

CHAPTER 3

ESTABLISHING A NETWORK CONNECTION

In order to establish a network connection, you need one task that is willing to accept a connection (a target task) and another task to initiate the request for a connection (a source task). In the following discussion, it is easier to imagine these two tasks as existing on different nodes; however, there is no restriction that prohibits tasks on the same node from engaging in a network dialogue with one another.

A TOPS-20 task that wants to declare itself as a target task, available for network dialogue with other network tasks, must first obtain a Job File Number (JFN) identifying itself as a file on device SRV:. The target task must then open that SRV: file in order to have a logical link assigned to it by NSP. The task should also set up some form of software interrupt mechanism to be informed asynchronously of the arrival of a connect initiate message from a source task. Whenever a connect initiate message arrives, the target task can interrogate the connect attributes of the source task and decide whether to accept or reject the connection.

A TOPS-20 task that wants to initiate a network dialogue with a declared target task must first obtain a JFN for a network connection identifying the target task as a file on device DCN:. It must then open that DCN: file to have a logical link assigned and to have a connect initiate message sent to the target task.

A TOPS-20 task, if it so desires, can declare itself as a target task and also act as a source task by initiating a dialogue with some other task; the two actions are not mutually exclusive.

Figure 3-1 is a general overview of the dialogue that takes place during the establishing of a network connection.

A task at node DALLAS issues a GTJFN monitor call identifying itself as a target task named TEX. A subsequent OPENF monitor call informs the NSP task at node DALLAS that TEX is ready to receive connection requests from the network.

A task at node BOSTON issues a GTJFN monitor call identifying itself as a task named TONY and specifying a network connection to task TEX at node DALLAS. A subsequent OPENF monitor call causes the NSP task at node BOSTON to send a connect initiate message to node DALLAS.

The NSP task at node DALLAS knows that task TEX is accepting calls and forwards the connect initiate message. Task TEX decides whether to accept or reject the connection and returns a connect confirm or connect reject message to the source task TONY.

The individual steps are explained in detail in the rest of this chapter.

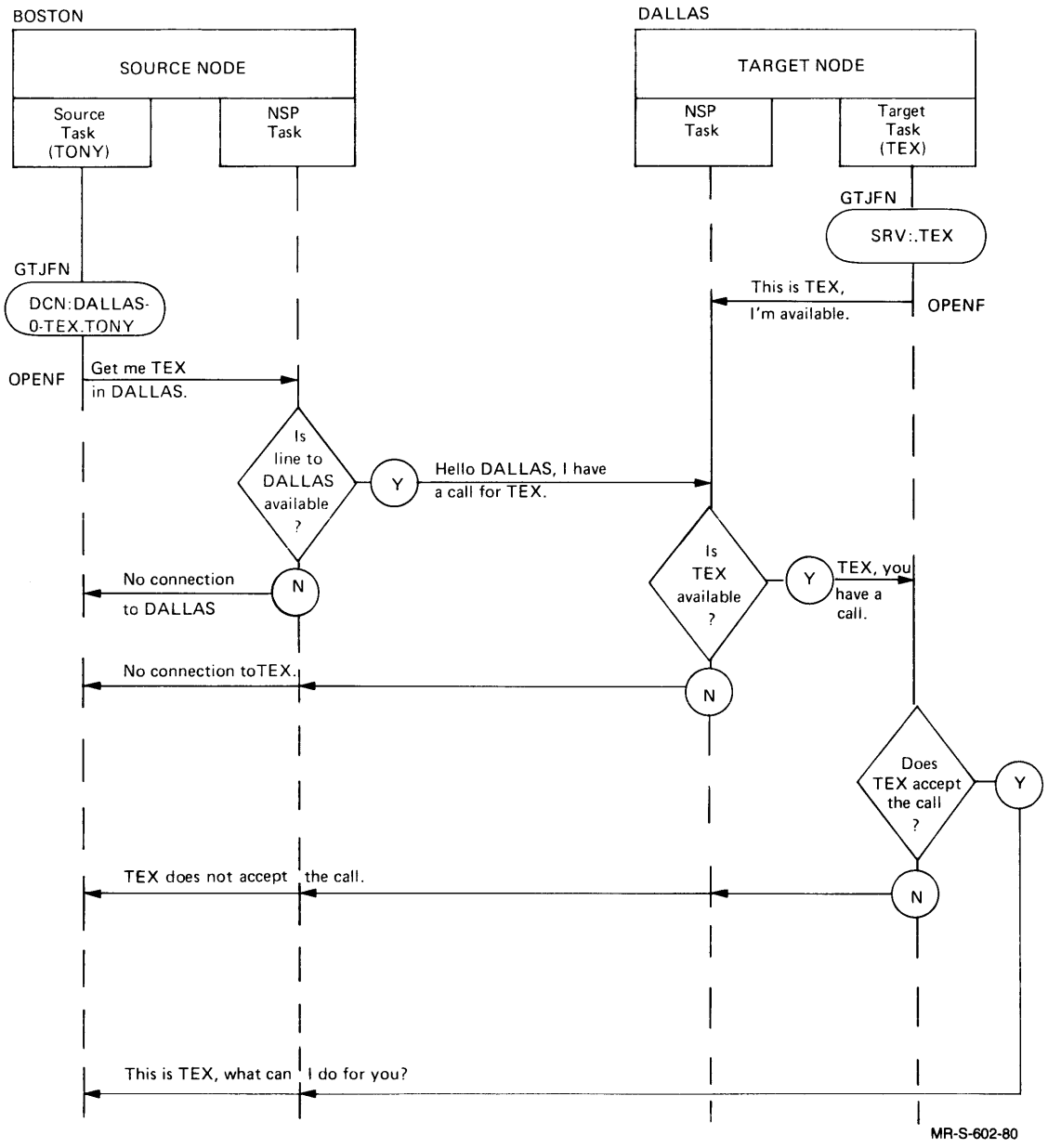


Figure 3-1 Establishing a Network Connection

3.1 OBTAINING A NETWORK JFN

The first step in establishing a network logical link is to obtain a Job File Number (JFN) for a network file on either the SRV: or DCN: device. Use the GTJFN monitor call in either its short or long form as described in detail in the Monitor Calls Reference Manual. The network file specification can be submitted interactively via the terminal, accessed from memory, or (in the long form) developed by a combination of both methods.

The general format of a GTJFN file specification in TOPS-20 is:

```
dev:<directory>filename.filetype.generation;T;P;A
```

When you use the GTJFN call to obtain a network JFN in DECnet-20, the network file specification takes the following form:

dev:	is replaced by one of the network pseudo-devices, SRV: or DCN:.
<directory>	is unused.
filename	is replaced by objectid-descriptor for an SRV: file and by hostname-objectid-descriptor for a DCN: file.
file type	is replaced by a task name uniquely identifying the task issuing the GTJFN.
generation	is unused.
;T;P;A	are replaced by network attributes.

The individual fields in the network file specification are described in detail in the following subsections.

3.1.1 Specifying a Target Task

Use the following format of the network file specification to obtain a JFN identifying yourself as a target task:

```
SRV:objectid-descriptor.taskname
```

where:

SRV:	is the logical device name for a target task.
objectid	is part of an optional generic identification for a target task. If included, it must be a nonzero object type expressed as a decimal number or an object name (see Appendix B). The numbers 1 through 127 are reserved for DECnet system tasks and require enabled WHEEL or OPERATOR privileges. Numbers 1 through 127 should not be assigned to user tasks unless the task provides the service and uses the protocol implied by the object type (see Appendix B). Numbers 128 through 255 are available to all tasks. If objectid is not specified, the target task must be addressed by its unique task name.
- (hyphen)	is a subfield separator that is required only if the descriptor is specified.

descriptor — is an optional modifier to be associated with the objectid. If specified, it must be 1 to 16 characters in length and contain only alphanumerics, hyphens, dollar signs, or underscores. If objectid is not specified, the descriptor must also be omitted. If descriptor is specified, it must also appear in the specification used by the source task to address this task (see Section 3.1.2).

NOTE

Some DECnet implementations do not allow a descriptor to be associated with a nonzero object type. When communicating with a non-DECnet-20 node, read the applicable documentation to determine any restrictions on the generic task identification.

. (period) — is a separator character and is required only if task name is specified.

taskname — is the unique task name by which a task is to be addressed independent of its generic identification. If taskname is specified, it must be 1 to 16 characters in length and contain only alphanumerics, hyphens, dollar signs, or underscores. If taskname is not specified, the monitor will assign one. (To subsequently determine the monitor-assigned task name, use the read task name function described in Section 3.4.3.)

The maximum lengths of the variable fields in the SRV: file specification as imposed by TOPS-20 are:

objectid	see object type and name in Appendix B
descriptor	16 characters
taskname	16 characters

The above maximums may be reduced by any size limitations imposed by the DECnet product running on the remote system.

3.1.2 Specifying a Network Connection

Use the following format of the network file specification to obtain a JFN identifying a target task that you wish to connect to:

```
DCN:hostname-objectid-descriptor.taskname;A1;A2...
```

where:

DCN: is the logical device name for a network connection.

hostname is the node name of the node on which the target task is running. If this field is omitted, the target task is assumed to be running on the local node.

- (hyphen) is a subfield separator and is required.

objectid is the identification of the target task. It is an object name or a numeric object type when addressing a target task by its generic identification (see Appendix B). The special object type 0 (or corresponding object name TASK) is used to address a target task by its unique task name. The objectid, when specified as a numeric object type, must be entered in decimal. This subfield is required.

- (hyphen) is a subfield separator that is required only if the descriptor is specified.

descriptor is an optional modifier to be associated with the objectid. If objectid is TASK or 0, this field must be the unique task name of the target task. If objectid identifies some other object type, this field must be the descriptor specified by the target task.

NOTE

Some DECnet implementations do not allow a descriptor to be associated with a nonzero object type. When you wish to communicate with a non-DECnet-20 node, read the applicable documentation to determine any restrictions on the generic task identification.

. (period) is a separator character and is required only if taskname is specified.

taskname is the unique taskname of the source task initiating the network connection. If taskname is specified, it must be 1 to 16 characters in length and contain only alphanumerics, hyphens, dollar signs, or underscores. If taskname is not specified, the monitor will assign one. (To subsequently determine the monitor-assigned task name, use the read task name function described in Section 3.4.3.)

;A1;A2... are a collection of attributes of the source task that are included in the connect initiate message sent to the target task. These attributes can be used by the target task to validate a network connection or to perform any other handshaking functions agreed to by both tasks. The allowable attributes are:

;USERID:userid — where userid consists of 1 to 16 contiguous alphanumeric ASCII characters (including the hyphen, dollar sign, and underscore) identifying the source task.

Example:

```
;USERID:ALIBABA
```

;PASSWORD:password — where password consists of 1 to 8 contiguous alphanumeric ASCII characters (including the hyphen, dollar sign, and underscore) required by the target task to validate the connection.

Example:

```
;PASSWORD:SESAME
```

The password can also be specified in binary to allow non-ASCII characters. The keyword for this type of entry is BPASSWORD.

;BPASSWORD:password — where password, in this context, consists of 1 to 8 octal triplets representing the required password. Each triplet represents an 8-bit byte.

Example:

```
BPASSWORD:123056002
```

;CHARGE:acctno — where acctno consists of 1 to 16 contiguous alphanumeric ASCII characters (including the hyphen, dollar sign, and underscore) representing the source task's account identification.

Example:

```
;CHARGE:ACCT-13C
```

;DATA:userdata	where userdata consists of 1 to 16 contiguous alphanumeric ASCII characters (including the hyphen, dollar sign, and underscore) representing user data. Example: ;DATA:THIS-IS-A-TEST The user data can also be specified in binary to allow non-ASCII characters. The keyword for this type of entry is BDATA.
;BDATA:userdata	where userdata, in this context, consists of 1 to 13 octal triplets representing user data. Each triplet represents an 8-bit byte. Example: ;BDATA:231337001

The attributes of a source task can be retrieved by a target task via functions of the MTOPR monitor call (see Section 3.4).

The maximum lengths of the variable fields in the DCN: file specification as imposed by TOPS-20 are:

hostname	6 characters
objectid	see object type and name in Appendix B
descriptor	16 characters
hostname-objectid-descriptor	39 characters including the hyphens
taskname	16 characters
;A1;A2...	see the description of the individual attribute

The above maximums may be reduced by any size limitations imposed by the DECnet product running on the remote host system.

3.1.3 Examples of Network File Specifications

The following examples show various ways that a target task can declare itself and the corresponding ways that a source task must use to address the target task. These examples assume two DECnet-20 nodes with host node names of BOSTON and DALLAS.

Example 1

A task at node BOSTON wants to declare itself as the unique target task SAM. It does so with the specification:

```
SRV:.SAM
```

In order to request a connection to SAM at node BOSTON, a task TEX at node DALLAS would specify:

```
DCN:BOSTON-TASK-SAM.TEX
```

A task COD at node BOSTON requesting a connection to SAM at node BOSTON can omit the node name in the specification because the target node is the local node. It need only specify:

```
DCN:-TASK-SAM.COD
```

Example 2

A task at node BOSTON wants to declare itself as a generic service task, object type 128. It does so with the specification:

```
SRV:128
```

Task TEX at node DALLAS can connect to the above task with the specification:

```
DCN:BOSTON-128.TEX
```

Assume that the BOSTON task had included a descriptor in its specification such as:

```
SRV:128-PART1
```

The DALLAS task would then have to modify its specification to:

```
DCN:BOSTON-128-PART1.TEX
```

Example 3

Several tasks at node BOSTON, running the same utility program, want to declare themselves both generically as object type 129 and uniquely by task name. The respective specifications used to declare three such tasks are:

```
SRV:129.TOM
SRV:129.DON
SRV:129.TONY
```

A task TEX at node DALLAS, wanting to avail itself of the utility but not caring which copy of the program completes the connection, can use the specification:

```
DCN:BOSTON-129.TEX
```

If, for some reason, TEX had to connect to the particular task TONY, the specification must be submitted as:

```
DCN:BOSTON-TASK-TONY.TEX
```

Example 4

A task at node BOSTON declares itself as the target task XDATA with the specification:

```
SRV:.XDATA
```

Assume that the task XDATA restricts connections to those remote tasks that have a valid userid, password, and charge account. A specification from task TEX at node DALLAS to connect to XDATA would then have to include the above attributes, for example:

```
DCN:BOSTON-TASK-XDATA.TEX;USERID:RITTER;PASSWORD:SESAME
 ;CHARGE:ACCT-XYZ
```

The target task can then confirm the connect requirements by retrieving the network attributes using the read logical link data functions of the MTOPR monitor call. These functions are described in Section 3.4.

3.2 OPENING A NETWORK JFN

Having obtained a JFN for a network file specification, the network task must then open the file with the OPENF monitor call. The monitor passes control to the appropriate device handler, which for the SRV: and DCN: devices is the local NSP task. The events that occur when a network file is opened depend on whether the file represents a target task or a source task.

3.2.1 Opening a Target Task JFN

An OPENF monitor call for a JFN that represents a target task implies that the task is ready to accept connect initiate messages from other tasks in the network. The local NSP performs the following:

- Constructs a link data base for this connection.
- Places the target task on a list of available connections.

Subsequently, when a connect initiate message is received from a source task, the local NSP performs the following:

- Searches the list of available connections for a matching generic or unique task identification.
- Notifies the appropriate target task via a connect interrupt that it has a connect request pending and modifies the link status appropriately.

The target task can then access the logical link data (see Section 3.4) to determine whether to accept or reject the connection.

3.2.2 Opening a Network Connection

An OPENF monitor call for a JFN that represents a network connection implies a request for a connection to a target task. The local NSP task performs the following:

- Constructs a link data base for this connection.
- Generates a connect initiate message and forwards it to the host node specified by the host name in the DCN: file specification.
- Processes the resulting connect confirm or connect reject message, and notifies the source task of the acceptance or rejection via a connect interrupt.

NOTE

The successful completion of the OPENF monitor call for a network connection does not necessarily mean that a connect confirm message has been received by the local NSP. The source task should read the link status (Section 3.4.1) to ensure that a connection has been completed.

3.2.3 Limit on Open Links

DECnet-20 software sets a user quota of four open links per job. These can be any combination of SRV: and DCN: types. However, a task running with enabled WHEEL or OPERATOR privileges is not bound by this quota and may open as many links as the system will allow.

The system quota of open links varies according to the amount of free space available at the time. Free space, in turn, is dependent upon the current demands of other processes. Whenever a request to open a link cannot be completed because of insufficient free space, an appropriate error code is returned.

When DECnet-20 is generated on a DECSYSTEM-2040, 2050, 2060, the maximum number of logical links is set for the individual site according to the site's configuration. See Chapter 9 in this manual.

3.3 ENABLING FOR NETWORK INTERRUPTS

Whenever a MACRO task uses a SIN or SINR monitor call to input a data string from the network, the task will stop running (block) if no data is available. In situations where a network task is supporting multiple links, blocking after each SIN or SINR call severely impacts the speed of data transmission. Asynchronously notifying the task of the arrival of network data reduces idle time and increases the overall throughput.

DECnet-20 software provides a function of the MTOPR monitor call that allows a network task to enable software interrupt channels for any combination of the following types of network events:

- connect event pending (connect initiate, connect confirm)
- interrupt message available
- data message or disconnect received

The MTOPR calling sequence to enable for network interrupts expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MOACN (function code)

AC3: Control information specifying the changes in the interrupt assignments for this link. This control information is placed in three 9-bit fields that are defined as follows:

Field	Symbol	Used to signal
B0-B8	MO%CDN	Connect event pending
B9-B17	MO%INA	Interrupt message available
B18-B26	MO%DAV	Data available

The content of each of these fields must be one of the following:

Value	Meaning
nnn	Enable the channel specified by nnn
.MOCIA	Clear the interrupt
.MONCI	Do not change the previous setting

Valid channel numbers are 0-5 and 23-35, decimal.

3.3.1 Example

The following program segment illustrates one method of enabling interrupt channels for the three types of network events:

```
;  SET UP THE INTERRUPT CHANNELS AND GO INTO WAIT STATE

ENACHN: MOVEI   T1,.FHSLF                ;IDENTIFY CURRENT PROCESS
        MOVE    T2,[LEVTAB,,CHNTAB]      ;SPECIFY TABLE ADDRESSES
        SIR                              ;DEFINE PSI SYSTEM TABLES
        MOVX    T2,7B2                   ;SET BITS FOR CHAN 0,1,2
        AIC                              ;ACTIVATE CHANNELS 0,1,2
        EIR                              ;ENABLE THE SYSTEM
        MOVE    T1,NETJFN                ;GET NETWORK JFN
        MOVEI   T2,.MOACN                ;SET UP FUNCTION
        MOVX    T3,<FLD(0,MO%CDN)+FLD(1,MO%DAV)+FLD(2,MO%INA)>
        MTOPR                            ;ISSUE THE CALL
PAUSE:  WAIT                             ;WAIT FOR INTERRUPT
        .
        .
        .
LEVTAB: PC                               ;LEVEL 1 PC ADDRESS
        0
        0
CHNTAB: 1,,HELLO                         ;CONNECT INTERRUPT
        1,,HAVDAT                        ;DATA AVAILABLE INTERRUPT
        1,,INTRPT                        ;INTERRUPT MESSAGE INTERRUPT
        REPEAT  ^D33,<EXP 0>             ;UNUSED INTERRUPT CHANNELS
PC:     BLOCK 1                          ;LEVEL 1 PC
```

3.4 READING LOGICAL LINK DATA

Associated with each open logical link is a link data base, created and maintained by the local NSP. This data base contains information such as link status, link control data, allowable segment sizes, and data governing the transmission and receipt of data and interrupt messages.

Whenever a target task is notified of a pending connect request from a source task, the target task's data base will contain the connect attributes submitted by the source task. These attributes, as well as other link data, can be retrieved by a target task via network functions of the MTOPR monitor call. These functions retrieve the source's host name, task name, user identification, password, user account number, and optional user data. Using this data, the target task can decide whether to accept or reject the connection.

3.4.1 Reading the Link Status

The read link status function of MTOPR returns a 36-bit word of information regarding the status of the logical link.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORLS (function code)

The monitor call returns the following information in AC3:

AC3: Flag bits in the left half and a disconnect code in the right half

The flag bits are:

Symbol	Bit	Meaning
MO%CON	B0	Link is connected
MO%SRV	B1	Link is a server
MO%WFC	B2	Link is waiting for a connect
MO%WCC	B3	Link is waiting for a connect confirm
MO%EOM	B4	Link has the end of, or entire, message to be read
MO%ABT	B5	Link has been aborted
MO%SYN	B6	Link has been disconnected normally
MO%INT	B7	Link has an interrupt message available
MO%LWC	B8	Link has been previously connected

The various disconnect codes are listed in Appendix A. If a disconnect code does not apply to the current status of the link, the value of the right half of AC3 will be zero.

Example

Assume that a source task obtains a JFN for a connection to a target task and opens the JFN. A successful return from the OPENF call does not necessarily mean that the local NSP has received a connect confirm message from the target node or task. To ensure that a connection really exists, you can use a coding sequence such as:

```
CHKCON: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORLS               ;SET UP FUNCTION
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        TXNE    T3,MO%CON               ;NOW CONNECTED?
        JRST    CNCTED                  ;YES
        TXNE    T3,MO%WCC               ;NO, WAITING FOR CC?
        JRST    [MOVEI T1,^D1000        ;YES, WAIT A
                 DISMS                  ;SECOND AND
                 JRST CHKCON]           ;TRY AGAIN
        TXNN    T3,MO%LWC               ;NO, EVER CONNECTED?
        JRST    REJECT                  ;NO, CI REJECTED
        JRST    ABORT                   ;YES, LINK WAS ABORTED
CNCTED:                                 ;OK, CONTINUE
```

3.4.2 Reading the Host Name

The read host name function of MTOPR returns the ASCII name of the node at the other end of the logical link.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORHN (function code)

AC3: A byte pointer to the location where the node name is to be stored

The monitor call returns with an updated pointer in AC3 and the node name stored as specified.

Example

A target task may wish to give special connection privileges to tasks running on a particular remote node. The following program segment will retrieve the name of the node submitting the current connect request and store it at location NODNAM:

```
GTHNAM: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORHN               ;SET UP FUNCTION
        HRROI   T3,NODNAM               ;POINTER TO NODE NAME
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
          .
          .
          .
NODNAM: BLOCK   2                       ;REMOTE NODE NAME
```

3.4.3 Reading the Task Name

The read task name function of MTOPR returns the unique task name that is associated with your end of the logical link. If you had defaulted the task name in the network file specification, the call returns the monitor-supplied task name. In DECnet-20, the default or monitor-supplied task name is actually a unique number.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORTN (function code)

AC3: A byte pointer to the location where the task name is to be stored

The monitor call returns with an updated pointer in AC3 and the task name stored as specified.

Example

Target tasks, especially those that perform utility functions, often default their task names because they do not expect to be addressed other than generically. If a connected source task wishes to initiate a second connection to your particular target task, the connected task requires your unique task name. You can first retrieve the monitor assigned name using a program segment as follows:

```
GTDTNM: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORTN               ;SET UP FUNCTION
        HRROI   T3,TSKNAM               ;POINTER TO TASK NAME
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
         .
         .
         .
TSKNAM: BLOCK   4                       ;LOCAL TASK NAME
```

Once retrieved, the task name can be sent to the connected source task via a data transfer or interrupt message (see Sections 4.1.1 and 4.2.1).

3.4.4 Reading the User Name

The read user name function of MTOPR is valid only for target tasks. It returns the source task's ASCII user identification supplied in the connect initiate message.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORUS (function code)

AC3: A byte pointer to the location where the user identification is to be stored

The monitor call returns with an updated pointer in AC3 and the user identification stored as specified. If no user identification was supplied by the source task, AC3 continues to point to the beginning of the string and a null is returned as the only character.

Example

A target task may include code to reject connect initiate requests from all but a select group of userids. The following program segment retrieves the userid of the current connect request:

```
GTUSID: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORUS               ;SET UP FUNCTION
        MOVE    T3,UIDPTR               ;POINTER TO USERID
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        CAMN    T3,UIDPTR               ;CHECK IF ANY USERID
         JRST   REJECT                  ;NO - REJECT
          .
          .
          .
        check if userid is valid
          .
          .
          .
USERID: BLOCK   4                       ;SOURCE USERID
UIDPTR: POINT   7,USERID                ;USERID POINTER
```

3.4.5 Reading the Password

The read password function of MTOPR is valid only for target tasks. It returns the source task's password supplied in the connect initiate message.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORPW (function code)

AC3: A byte pointer to the location where the password is to be stored. Passwords may be binary; therefore, the byte pointer should accommodate 8-bit bytes unless you know that the password is ASCII.

The monitor call returns with an updated pointer in AC3 and the source task's password stored as specified. AC4 contains the number of bytes in the string; a zero value indicates that no password was supplied by the source task.

Example

In addition to screening a source task's userid, a target task may require the submission of a password before confirming a connect request. Whereas a userid is usually permanent, a password can be changed periodically to ensure security. The following program segment retrieves the password submitted by a source task in its connect request. The two tasks have agreed to use ASCII passwords:

```
GTPSWD: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORPW               ;SET UP FUNCTION
        MOVE    T3,PWDPTR               ;POINTER TO PASSWORD
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        JUMPE   T4,REJECT               ;REJECT IF NO PASSWORD
         .
         .
        check for valid password
         .
         .
         .
PWDPTR: POINT   7,PASSWD                ;PASSWORD POINTER
PASSWD: BLOCK   2                       ;SOURCE PASSWORD
```

3.4.6 Reading the Account String

The read account string function of MTOPR is valid only for target tasks. It returns the ASCII account string supplied by the source task in the connect initiate message.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORAC (function code)

AC3: A byte pointer to the location where the account string is to be stored

The monitor call returns with an updated pointer in AC3 and the source task's account string stored as specified. If no account string was supplied by the source task, AC3 continues to point to the beginning of the string and a null is returned as the only character.

Example

A target task that includes a cost distribution of its services may set up a chart of accounts and require each connecting task to supply an account identification. With this information the target task can control access, set budgets, check overruns, and provide billing data. The following program segment retrieves the account string supplied in a connect initiate message:

```
GTACCT: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORAC               ;SET UP FUNCTION
        MOVE    T3,ACCPTR               ;POINTER TO ACCT NO.
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        CAMN    T3,ACCPTR               ;CHECK IF ANY ACCT NO.
         JRST   REJECT                  ;NO - REJECT
          .
          .
        process the account number
          .
          .
          .
ACCTNO: BLOCK   4                       ;ACCOUNT STRING
ACCPTR: POINT   7,ACCTNO                ;ACCT NO. POINTER
```

3.4.7 Reading the Optional Data

The read optional data function of MTOPR returns the optional data supplied in any of the connect or disconnect messages.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORDA (function code)

AC3: A byte pointer to the location where the optional user data is to be stored. This field may be binary; therefore, the byte pointer should accommodate 8-bit bytes unless you know that the data is ASCII.

The monitor call returns with an updated pointer in AC3 and the optional data stored as specified. AC4 contains the number of bytes in the data string; a zero value indicates that no optional data was supplied.

Example

The user level protocol, agreed to by two corresponding tasks, may state that optional user data will always accompany a connect reject message. The following program segment will retrieve optional user data in binary:

```
GTDATA: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORDA               ;SET UP FUNCTION
        MOVE    T3,DATPTR               ;POINTER TO USER DATA
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        JUMPE   T4,NODATA               ;BRANCH IF NO DATA
          .
          .
          .
DATPTR: POINT   8,USRDAT                ;USER DATA POINTER
USRDAT: BLOCK   4                       ;USER DATA
```

3.4.8 Reading the Object Type

The read object type function of MTOPR is valid only for target tasks. It returns the object type that was used by the source task to address this connection. The result indicates whether the local task was addressed by its generic object type or its unique network task name.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORCN (function code)

The monitor call returns with the object type in AC3. A zero object type indicates that the target task was addressed by its unique network task name; a nonzero value indicates that it was addressed by its generic object type.

Example

Assume for example that the services of a target task depend on whether a source task connects to it by generic object type or by unique task name. The following program segment retrieves the object type used in the connect initiate message:

```
GTOBJT: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORCN               ;SET UP FUNCTION
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        JUMPE   T3,TSKCON               ;TEST TYPE OF CONNECT
OBJCON:  .                              ;CONNECTED BY OBJECT TYPE
         .
         .
TSKCON:  .                              ;CONNECTED BY TASK NAME
         .
         .
```

3.4.9 Reading the Object-Descriptor

The read object-descriptor function of MTOPR is valid only for target tasks. It returns the unique identification of the source task. This identification is in the format of object-descriptor and the contents depend on the DECnet implementation on the remote host. In addition, if the source task is running on a system that provides for group and user codes, this information is also returned. If the source task is on a DECnet-20 host, the data returned to the target task is TASK-taskname.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MOROD (function code)

AC3: A byte pointer to the location where the object-descriptor of the source task is to be stored

The monitor call returns with an updated pointer in AC3 and the object-descriptor stored as specified. In addition, if the source host system uses group and user codes, AC4 contains the following:

AC4: The group code in the left half and the user code in the right half

If the source host system does not provide for group or user codes, or if none were provided in the connect initiate message, AC4 contains zeros.

Example

A target task can retrieve the unique identification of the source task with the following program segment:

```
GTOBJD: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MOROD               ;SET UP FUNCTION
        HRROI   T3,OBJDES               ;POINTER TO OBJ. DESC.
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        JUMPN   T4,[HRRZM T4,USRCOD     ;SAVE USER CODE
                HLRZM T4,GRPCOD         ;SAVE GROUP CODE
                 JRST .+1]              ;RETURN
          .
          .
          .
OBJDES: BLOCK   5                       ;OBJECT-DESCRIPTOR
GRPCOD: BLOCK   1                       ;GROUP CODE
USRCOD: BLOCK   1                       ;USER CODE
```

3.4.10 Reading the Segment Size

The read segment size function of MTOPR returns the maximum segment size that can be sent over this link. The maximum segment size that can be sent over a logical link is a function of the buffering capabilities of the remote node and the particular implementation of DECnet that is running on the remote node. The local task can use this value to determine the optimum size of data records being transmitted over the link.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORSS (function code)

The monitor call returns with the maximum segment size, in bytes, in AC3. If the link has not been established, the monitor call takes the error return.

Example

The maximum segment size for a source task is transmitted to the target NSP in the connect initiate message. The target NSP, in turn, returns the target task's maximum segment size in the connect confirm message that is sent back to the source NSP. Each task can then retrieve this value with a program segment such as the following:

```
GTSGSZ: MOVE    T1,NETJFN              ;GET NETWORK JFN
        MOVEI   T2,.MORSS              ;SET UP FUNCTION
        MTOPR                          ;ISSUE THE CALL
         ERJMP  JYSXX                  ;JSYS ERROR
        MOVEM   T3,SEGSIZ              ;STORE SEGMENT SIZE
          .
          .
          .
SEGSIZ: BLOCK   1                      ;MAX SEGMENT SIZE
```

3.5 ACCEPTING OR REJECTING A CONNECTION

When the target task has decided to accept or reject the connection, it must inform the local NSP, via the monitor, of its intention. Two MTOPR monitor call functions are provided: .MOCC to accept a connection and return some data, and .MOCLZ to reject the connection and return some data along with a specific reject reason. If no data or specific reject reason is to be returned, the target task can accept or reject the connection implicitly, without using either of the two MTOPR functions. This is explained in the following subsections.

3.5.1 Accepting the Connection

Connections can be accepted either explicitly or implicitly.

You can accept a connection explicitly by sending a connect confirm message to the source task via the .MOCC function of the MTOPR monitor call. This method allows you to include up to 16 bytes of optional data in the connect confirm message.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MOCC (function code)

AC3: A byte pointer to any data to be returned

AC4: The count of bytes in the data string. A zero indicates no data. The maximum amount of data is 16 bytes.

You can accept a connection implicitly by performing one of the operations that NSP recognizes as indicating acceptance. These operations are:

- Issuing an output monitor call such as BOUT, SOUT, or SOUTR to the network JFN
- Issuing an input monitor call such as BIN, SIN, or SINR to the network JFN
- Placing yourself in an input or output wait state

When NSP senses one of the above conditions, it assumes that you are accepting the connection and sends a connect confirm message to the source task. The connect confirm message sent by NSP does not allow for optional data; if you must return optional data, use the explicit acceptance.

3.5.2 Rejecting the Connection

Connections can be rejected either explicitly or implicitly.

You can reject a connection explicitly by sending a connect reject message to the source task via the .MOCLZ function of the MTOPR monitor call. This method allows you to include a reject code as well as up to 16 bytes of optional data in the connect reject message.

The calling sequence expects the following monitor call arguments to be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: A reject code in the left half and .MOCLZ in the right half

AC3: A byte pointer to any data to be returned

AC4: The count of bytes in the data string. A zero indicates no data. The maximum amount of data is 16 bytes.

The reject code in AC2 is a 2-byte, NSP-defined decimal number indicating the reason that a target task is rejecting a connection. A list of these codes, applicable to both user and system tasks, appears in Appendix A.

You can reject a connection implicitly by closing the JFN of the logical link before accepting the connection either explicitly or implicitly. To close the JFN, use the CLOSF monitor call. When NSP senses that the logical link's JFN has been closed in response to a connect initiate message, NSP sends a connect reject message to the source task with a reject code of 38 (user aborted). In this situation, you must reopen the JFN in order to receive subsequent connect initiate messages.

3.5.3 **Examples**

The following program segment will send a connect confirm message to the source task that requested the connection:

```
ACCEPT: MOVE    T1,NETJFN               ;GET THE NETWORK JFN
        MOVEI   T2,.MOCC                ;CODE TO ACCEPT
        SETZ    T4,                     ;FLAG NO OPT. DATA
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
```

To include up to 16 bytes of ASCII user data with the connect confirm message, modify the above segment as follows:

```
ACCEPT: MOVE    T1,NETJFN               ;GET THE NETWORK JFN
        MOVEI   T2,.MOCC                ;CODE TO ACCEPT
        MOVE    T3,[POINT 7,MSGC]       ;POINTER TO USER DATA
        MOVEI   T4,^D16                 ;USER DATA BYTE COUNT
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
         .
         .
         .
MSGC:   ASCIZ   /OPEN UNTIL 10 PM/
```

The following program segments will send a connect reject message to the source task that requested the connection. The reason for the rejection is coded as 34 - access not permitted. With no user data, the instruction sequence is:

```
REJECT: MOVE    T1,NETJFN               ;GET THE NETWORK JFN
        MOVEI   T2,.MOCLZ               ;CODE TO REJECT
        HRLI    T2,.DCX34               ;ADD REJECT CODE 34
        SETZ    T4,                     ;FLAG NO USER DATA
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
```

Because reject code 34 is somewhat general, you may want to include ASCII user data to clarify the rejection. You can modify the above sequence as follows:

```
REJECT: MOVE    T1,NETJFN               ;GET THE NETWORK JFN
        MOVEI   T2,.MOCLZ               ;CODE TO REJECT
        HRLI    T2,.DCX34               ;ADD REJECT CODE 34
        MOVE    T3,[POINT 7,XPWD]       ;POINTER TO REJECT MSG
        MOVE    T4,^D14                 ;REJECT MSG BYTE COUNT
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
         .
         .
         .
XPWD:   ASCIZ   /WRONG PASSWORD/
```

Figures 5-1 and 5-2 are examples, in program context, of some of the programming sequences described in this chapter.

CHAPTER 4

TRANSFERRING INFORMATION OVER THE NETWORK

Once a network connection has been established, the task at either end of the logical link can send information to the task at the other end. DECnet-20 provides for three general types of information exchange:

- Data transfers
- Interrupt messages
- Network utility messages

Data transfers are primarily used by network tasks to move large blocks of data. Interrupt messages are used by network tasks to exchange small amounts (16 bytes or less) of high priority data that are not sequentially related to the main data flow. Network utility messages include link control messages, link service messages, and ACK/NAK messages.

Data transfers and interrupt messages are discussed in the remainder of this chapter. Network utility messages are internal to the DECnet-20 software and transparent to the user network tasks; they are described in the Network Services Protocol (NSP) functional specification listed in the bibliography of Appendix D.

4.1 TRANSFERRING DATA

Data transfers over a logical link involve the segmentation and restructuring of data at both the logical and physical link levels. The local NSP accepts data from the user program, segments it to conform to the maximum segment size allowable on that logical link, precedes each segment with an NSP header, and passes these NSP segments to the physical link management, DDCMP. DDCMP, in turn, segments the input from NSP to conform to the maximum segment size allowable on the physical link, precedes each segment with a DDCMP header to form a packet, and sends these packets over the physical media to the destination node.

At the destination node the reverse procedure takes place. DDCMP strips each packet of its DDCMP header and passes the packet data to the local NSP. The local NSP reassembles the NSP segments, strips off the NSP headers, and passes the data to the receiving task.

Data transfers on a logical link can take one of two forms: logical messages or continuous byte streams.

The logical message format provides for the transmission of information in discrete logical units called records, or messages. Data transmitted in this format can be retrieved by the receiving task on a message-by-message basis with NSP taking care of delimiting each logical unit.

The continuous byte stream format, on the other hand, does not implement any end-of-message indicators. Data transmitted in this format is presented to the receiving task as a continuous stream of data with no indication by NSP of the end of one message and the beginning of another. The receiving task must reconstruct the original messages via some prearranged protocol agreed to by both user tasks.

The logical message format allows for simpler user retrieval routines at the expense of not taking full advantage of the monitor's buffering capabilities. The continuous byte stream format allows NSP and DDCMP to pack data buffers without regard for the ends of messages; however, the user retrieval routines must then analyze the input stream and reconstruct the original logical messages.

4.1.1 Sending Data

You can send data to another task with either the SOUT or the SOUTR monitor call. In general, use SOUTR to send data in the logical message format and SOUT to send data as a continuous byte stream.

The exclusive use of SOUTR usually implies that both you and the other task have agreed to exchange information in the form of messages with some stated maximum length. Each use of the SOUTR monitor call results in the transmission of a logical message terminated with an end-of-message indicator. A DECnet-20 receiving task can then retrieve the message with the SINR (read record) monitor call (see Section 4.1.2).

The exclusive use of SOUT or BOUT usually implies that both you and the other task have agreed to exchange information in the form of a continuous byte stream. You can repeatedly fill your data buffer and execute the SOUT monitor call. Each SOUT transmits a buffer's worth of data to the destination node where it is presented to the receiving task. A DECnet-20 target task can then use the SIN and BIN (read data) monitor calls (see Section 4.1.2) to retrieve as many bytes at a time as it can handle in its data buffer. Normally, you would include a count byte at the beginning of each logical data group to provide the receiving task with a means to reconstruct the logical data.

You can also intermix the use of SOUT and SOUTR when you send data to a receiving task. For example, assume that you are programming a task with a data buffer limited to 300 bytes. The receiving task has a 1000 byte buffer and requires that data be sent to it in logical message format. You can send an 800 byte logical message by sending two SOUTs for 300 bytes each followed by a SOUTR for 200 bytes. The receiving task can then retrieve the entire message by using a SINR for 800 bytes or more.

4.1.2 Receiving Data

You can retrieve data from the network using the SINR, SIN, or BIN monitor calls. In general, use SINR to retrieve logical messages and SIN to retrieve data from a continuous byte stream.

The exclusive use of SINR usually implies that both you and the other task have agreed to exchange information in the form of messages with some stated maximum length. Each SINR monitor call results in the retrieval of one logical message. The sending task must have sent the message using the SOUTR monitor call (see Section 4.1.1).

The exclusive use of SIN usually implies that both you and the other task have agreed to exchange information in the form of a continuous byte stream. You can retrieve as many bytes at a time as your data buffer can handle. If your data buffer is 300 bytes in length, you can execute a SIN for 300 bytes and fill the buffer with data. The manner in which you then reconstruct the original logical messages depends on the user-level protocol agreed to by both tasks. For example, if each message is preceded by a count byte, you can use the BIN monitor call to read the first byte and obtain the number of bytes in the message and then issue a SIN for that number of bytes to retrieve the message. A subsequent BIN would read the number of bytes in the next message.

You can intermix the use of SIN and SINR when you wish to retrieve data from the network. However, a few precautions are in order. A SIN monitor call does not recognize an end-of-message indicator if one is present. Therefore, if you issue a SIN for 300 bytes and the link buffer contained the last 200 bytes of a logical message (sent with a SOUTR), you will retrieve those 200 bytes plus the first 100 bytes of the next message. The SIBE monitor call is useful here because it will return the number of bytes of the current message that are available in the link buffer. You can then issue a SIN for that number of bytes and not encroach on the succeeding message.

The SIBE monitor call should also be used just prior to dismissing a data available interrupt; this is to ensure that you have read all input currently in the buffer. Failure to test for an empty buffer can result in loss of a data interrupt. If data from the network arrives while the program is at interrupt level, no interrupt is given because the program has not yet dismissed the first interrupt. SIBE should always precede DEBRK to ensure no data is lost.

4.2 TRANSFERRING INTERRUPT MESSAGES

Although data transfers over a logical link are guaranteed to be received at the other end in the same order in which they were sent, it is occasionally necessary to bypass the normal flow of data and to send data that is to be delivered immediately. Events such as errors or status changes in one task or the other are situations that justify bypassing the normal data flow.

The logical link management level, NSP, allows for the transmission and reception of short high-priority messages called interrupt messages. An interrupt message is sent and accounted for independently of any buffered data messages and its delivery is guaranteed to be prompt. Interrupt messages are limited to 16 bytes in length and therefore are not very useful for exchanging data. They are most effectively used as event indicators and usually require the subsequent exchange of data by the two processes owning the logical link. In this respect, they closely resemble software interrupts. Consequently, DECnet-20 provides the network task with a monitor call function to enable an interrupt channel for the receipt of an interrupt message. (See Section 3.3.)

4.2.1 Sending Interrupt Messages

A network task communicating over a logical link can initiate an interrupt message at any time. Whether DECnet-20 will send the message over the link depends on conditions at the other end. If the task at the other end has not acknowledged a previous interrupt message, the sending task is notified via an error message. If the other task is not enabled for interrupt messages, the link will not accept the message.

To send an interrupt message, use the .MOSIM function of the MTOPR monitor call.

The calling sequence expects the following:

AC1: The JFN of the logical link

AC2: .MOSIM (function code)

AC3: A byte pointer to the message

AC4: The count of bytes in the interrupt message. The maximum message length is 16 bytes.

Example

The following program segment can be used to send an interrupt message to another task:

```
SNDMSG: MOVE    T1,NETJFN                ;GET NETWORK JFN
        MOVEI   T2,.MOSIM                ;SET UP FUNCTION
        HRROI   T3,MSG                   ;POINTER TO MESSAGE
        MOVEI   T4,^D14                  ;BYTE COUNT
        MTOPR                            ;ISSUE THE CALL
         ERJMP  JSYSXX                   ;JSYS ERROR
          .
          .
          .
MSG:    ASCIZ   /CLOSING AT 6PM/
```

4.2.2 Receiving Interrupt Messages

If the protocol used by network tasks includes interrupt messages, each task should provide a way to be asynchronously notified of the messages' arrival. The .MOACN function of the MTOPR monitor call allows a network task to enable specific channels for software interrupts (see Section 3.3). One of these interrupts signals the arrival of an interrupt message.

When a remote task sends your task an interrupt message, the appropriate channel presents a software interrupt to your task. To read the interrupt message, use the .MORIM function of the MTOPR monitor call.

The calling sequence expects the following:

AC1: The JFN of the logical link

AC2: .MORIM (function code)

AC3: A byte pointer to the receiving buffer. The maximum message length is 16 bytes.

The call returns with an updated pointer in AC3, the message stored in the buffer, and the count of bytes received in AC4.

Because interrupt messages are used to signal important asynchronous events, it is recommended that a task receiving an interrupt message read the interrupt message promptly. Furthermore, DECnet-20 does not acknowledge an interrupt message until the task reads it. This means that each network task is limited to one outstanding interrupt message and until that one is read, no others will be accepted.

Example

The following program segment retrieves an interrupt message:

```
INTRPT: MOVE    T1,NETJFN                    ;GET NETWORK JFN
        MOVEI   T2,.MORIM                    ;SET UP FUNCTION
        MOVE    T3,[POINT 8,MSGBUF]          ;POINTER TO MESSAGE
        MTOPR                                ;ISSUE THE CALL
         ERJMP  JSYSXX                       ;JSYS ERROR
          .
          .
          .
MSGBUF: BLOCK   4                             ;MESSAGE BUFFER
```

Figures 5-1 and 5-2 are examples, in program context, of some of the programming sequences described in this chapter.

CHAPTER 5

CLOSING A NETWORK CONNECTION

A network connection can be closed by either of the two connected tasks. A connection can be closed normally, thereby preserving the integrity of any data in transit; or, a connection can be aborted without regard to any undelivered data.

5.1 CLOSING A CONNECTION NORMALLY

A normal close is usually accomplished with the CLOSF monitor call specifying a network JFN in AC1. The CZ%ABT bit in AC1 must be off. All buffered data that is in transit at the time is delivered (unless the remote task executes an abort before the CLOSF has completed). The network JFN is then closed.

An MTOPR call with function code .MOCLZ also disconnects the logical link and completes the delivery of all buffered data; however, it does not close the JFN. This method of closing a link is only used if it is necessary to send user data (up to 16 bytes) to the remote task. In order to send user data and also close the JFN, the .MOCLZ function must be followed by a CLOSF call.

The calling sequence for the MTOPR call is:

AC1: The JFN of the logical link

AC2: 0 in the left half and .MOCLZ in the right half

AC3: A byte pointer to the user data. If the byte size is over 8, bytes are truncated to eight bits.

AC4: The count of bytes in the user data. The maximum is 16 bytes.

The network does not have explicit protocol for a normal close. That is, no one specific network control message is available to both disconnect a logical link and also automatically have all data correctly delivered. When you use the MTOPR or CLOSF monitor call to close a network connection, you are actually turning over control of the link to the local NSP. It is the job of the local NSP to ensure that all outstanding data packets have been properly acknowledged before sending the disconnect message to the remote NSP.

The remote NSP, in turn, notifies the remote task according to the protocol in effect at the remote node.

If the remote node is a DECnet-20 node, the NSP task receiving the disconnect message sets the MO%SYN bit in the remote task's link status to reflect that the link has been closed normally. If the remote task is not in the process of reading data, it is issued a data interrupt. If the remote task issues a SIBE call, it will be informed that no bytes are available. If the remote task attempts to read data, it will receive an end-of-file indication. In any case, reading the link status via the .MORLS function of MTOPR will indicate that the MO%SYN bit has been set.

Example

To close a logical link after delivering all the data currently in transit, use a program segment such as the following:

```
CLOSE: MOVE    T1,NETJFN                    ;GET NETWORK JFN
       CLOSF                                ;ISSUE THE CALL
        ERJMP  JSYSXX                       ;JSYS ERROR
        .
        .
        .
```

To close a logical link as above and also include ASCII user data for the target task, a program segment such as the following can be used:

```
CLOSED: MOVE    T1,NETJFN                   ;GET NETWORK JFN
        MOVEI   T2,.MOCLZ                   ;SET UP FUNCTION
        MOVE    T3,[POINT 7,MSG]            ;POINTER TO MESSAGE
        MOVEI   T4,^D14                     ;BYTE COUNT
        MTOPR                               ;ISSUE THE CALL
         ERJMP  JSYSXX                      ;JSYS ERROR
        CLOSF                               ;CLOSE THE JFN
         ERJMP  JSYSXX                      ;JSYS ERROR
         .
         .
         .
MSG:    ASCIZ   /BE BACK AT 6PM/            ;USER DATA
```

5.2 ABORTING A CONNECTION

You can abort a logical link with the CLOSF monitor call by specifying both the CZ%ABT bit and the network JFN in AC1. All buffered data in transit is discarded and the network JFN is closed. This operation can result in the loss of data and should only be used in a fatal error condition.

The .MOCLZ function of MTOPR, used to normally close a logical link in Section 5.1, can be used to abort a logical link if you insert a nonzero code in the left half of AC2. This method of aborting a link should only be used if it is necessary to send the remote task a specific reason code for the abort, up to 16 bytes of user data, or both. The .MOCLZ function with the abort option discards all buffered data in transit and closes the link; however, it does not close the network JFN. To close the JFN, the MTOPR call must be followed by a CLOSF call with the CZ%ABT bit set in AC1.

The calling sequence for the MTOPR call is:

AC1: The JFN of the logical link

AC2: A reason code, nn, in the left half and .MOCLZ in the right half

AC3: A byte pointer to the user data

AC4: The count of bytes in the user data. The maximum is 16 bytes

The reason code (nn) in the left half of AC2 is one of the nonzero codes listed in Appendix A.

With either the CLOSF or MTOPR monitor call, the local NSP sends a disconnect message to the remote NSP which, in turn, notifies the remote task according to some established protocol.

If the remote node is a DECnet-20 node, the NSP task receiving the disconnect message sets the MO%ABT bit in the remote task's link status to reflect that the link has been aborted. If the remote task is not in the process of reading data, it is issued a data interrupt. Any attempts to read data will result in an I/O error. Reading the link status via the .MORLS function of MTOPR will indicate that the MO%ABT bit has been set and the right half of AC3 will contain a disconnect code if one was given.

Example

To abort a logical link immediately without completing the delivery of any data in transit, use a program segment such as the following:

```
ABORT: MOVE    T1,NETJFN                    ;GET NETWORK JFN
       TLO     T1,(CZ%ABT)                  ;SET ABORT BIT
       CLOSF                                ;ISSUE THE CALL
        ERJMP  JSYSXX                       ;JSYS ERROR
         .
         .
         .
```

To abort a logical link as above and also include a specific reason code and user data, use a program segment such as the following:

```
ABORTD: MOVE    T1,NETJFN                   ;GET NETWORK JFN
        MOVEI   T2,.MOCLZ                   ;SET UP FUNCTION
        HRLI    T2,.DCX9                    ;CODE FOR USER ABORT
        MOVE    T3,[POINT 7,MSGX]           ;POINTER TO MESSAGE
        MOVEI   T4,^D16                     ;BYTE COUNT
        MTOPR                               ;ISSUE THE CALL
         ERJMP  JSYSXX                      ;JSYS ERROR
        TLO     T1,(CZ%ABT)                 ;SET ABORT BIT
        CLOSF                               ;CLOSE THE JFN
         ERJMP  JSYSXX                      ;JSYS ERROR
          .
          .
          .
MSGX:   ASCIZ   /RESTART XMISSION/          ;USER DATA
```

If the target task has the MO%ABT bit set in the link status word, the target task must use CZ%ABT or the CLOSF will fail. An example of a program segment using CZ%ABT follows:

```
MOVE    T1,NETJFN
CLOSF
 ERJMP  [ MOVE  T1,NETJFN                    ;If fail, then
          TLO   T1,(CZ%ABT)                  ;use CZ%ABT
          CLOSF
            JRST JSYSXX
          JRST  .+1]
```

5.3 SOURCE AND TARGET TASK CODING EXAMPLES

Figures 5-1 and 5-2 are examples of coding for source and target programs, respectively. The examples present, in context, many of the program segments given as examples in Chapters 3, 4, and 5. The examples are not complete programs that can be executed.

```
; Get JFN for Network Connection
;
        MOVX    T1,GJ%SHT
        HRROI   T2,[ASCIZ/DCN:NODEA-TASK-TARGET.SOURCE/]
        GTJFN
         ERJMP  NOGOOD          ;Failed, Probably out of resources
        MOVEM   T1,OURJFN       ;Successful, save our JFN
;
; OPENF to create the Logical Link
;
        MOVX    T2,<FLD(^D7,OF%BSZ)+OF%WR+OF%RD> ;Open for read and
                                                 ;write
        OPENF
         ERJMP  NOGOOD          ;Failed
;
; Wait for network connect to succeed or fail
;
CHKST:  MOVX    T1,^D1000       ;Wait before checking status
        DISMS
        MOVE    T1,OURJFN       ;Check link status
        MOVX    T2,.MORLS
        MTOPR
         ERJMP  NOGOOD
        TXNE    T3,MO%CON       ;Connected?
         JRST   HELLO           ;Yes, proceed to HELLO
        TXNE    T3,MO%WCC       ;No, are we waiting still?
         JRST   CHKST           ;Yes, delay some more
; If we get here, target process or network rejected cannot attempt
        JRST    NOGOOD          ;We lose
;
; Send data to Target task
;
HELLO:  MOVE    T1,OURJFN
        HRROI   T2,[ASCIZ/Hello Target!
/]
        SETZM   T3
        SOUTR
         ERJMP  NOGOOD          ;Network went away
;
; CLOSF to disconnect logical link
;
        MOVE    T1,OURJFN
        CLOSF
         ERJMP  [ MOVE  T1,OURJFN       ;Failed, use CZ%ABT
                  TXO   T1,CZ%ABT       ; to close link
                  CLOSF
                   JFCL                 ;Don't Care if it fails
                  JRST  .+1]
        HALTF                           ;Stop source task
;
;Include what you wish to do on failure of logical link
;
NOGOOD: .
        .
OURJFN: BLOCK   1
```

Figure 5-1 Example of Source Task Coding

```
;
; Get JFN for Network Connection
;
START:  MOVX    T1,GJ%SHT
        HRROI   T2,[ASCIZ/SRV:.TARGET/]
        GTJFN
         ERJMP  NOGOOD          ;Failed, Probably out of resources
        MOVEM   T1,OURJFN       ;Successful, save our JFN
;
; Start setting up interrupt system for network JFN
;
        MOVX    T1,.FHSLF       ;Set up interrupt system first
        MOVE    T2,[LEVTAB,,CHNTAB]
        SIR                     ;Set interrupt system tables
        MOVX    T2,3B1          ;Enable channels 0 and 1
        AIC                     ;Activate interrupt channels
        EIR                     ;Enable for interrupts
;
; OPENF to make us available to the network
;
        MOVX    T2,<FLD(^D7,OF%BSZ)+OF%WR+OF%RD> ;Open for read and
                                                 ;write
        OPENF
         ERJMP  NOGOOD          ;Failed
;
; Finish setting up interrupt system for network JFN
;
        MOVE    T1,OURJFN       ;Set up Connect and Data Interrupts
        MOVEI   T2,.MOACN
        MOVX    T3,<FLD(0,MO%CDN)+FLD(1,MO%DAV)+FLD(.MONCI,MO%INA)>
        MTOPR
;
PAUSE:  WAIT                    ;Wait for Interrupts
;
LEVTAB: PC                      ;Level 1 PC address
        0
        0
CHNTAB: 1,,HELLO                ;On connect go to HELLO
        1,,READIT               ;On data interrupt try to read it
        REPEAT  ^D34,<0>        ;Zero fill rest of table

PC:     BLOCK   1               ;Level 1 PC save location
;
; Process interrupt on Connect channel
;
HELLO:  MOVE    T1,OURJFN       ;Always accept the connection
        MOVX    T2,.MOCC
        SETZB   T3,T4           ;No optional data
        MTOPR
         ERJMP  NOGOOD          ;Something blew up
        DEBRK                   ;Done, wait some more
```

Figure 5-2 Example of Target Task Coding

```
;
; Process interrupt on Data channel
;
READIT: MOVE    T1,OURJFN       ;Any data?
        SIBE
         JRST   READ            ;Yes, Process it
        MOVX    T2,.MORLS       ;No, See if link still connected
        MTOPR
         ERJMP  NOGOOD          ;An error here is not likely, but...
        TXNN    T3,MO%CON       ;Link Still Connected?
         JRST   DISCON          ;No, process link down
        DEBRK                   ;Yes, then wait for another interrupt
;
READ:   HRROI   T2,BUFFER       ;Put data into buffer
        MOVNI   T3,^D1000       ;- Size of buffer
        SINR                    ;Read the data
         ERJMP  NOGOOD          ;Shouldn't happen
        SETZM   T1              ;Store a zero byte
        IDPB    T1,T2
        HRROI   T1,BUFFER
        PSOUT                   ;Output the message
        JRST    READIT          ;Process any more input
;
; Process disconnect on link
;
DISCON: MOVE    T1,OURJFN       ;Close our JFN
        CLOSF
         ERJMP  [ MOVE  T1,OURJFN       ;Try CLOSF with CZ%ABT
                  TXO   T1,CZ%ABT
                  CLOSF
                   JFCL         ;Don't care if it fails
                  JRST .+1]
        JRST    START           ;Start over
;
; Include what you wish to do on failure
;
NOGOOD: .
        .
        .
;
BUFFER: BLOCK   ^D1000          ;Buffer save location
OURJFN: BLOCK   1               ;OURJFN save location
;
```

Figure 5-2 (Cont.) Example of Target Task Coding

PART III

OPERATOR'S GUIDE

CHAPTER 6

RUNNING DECnet-20

The Operator Command Language program, OPR, provides the operator with one command language to communicate with several TOPS-20 components, including the DECnet-20 control program, NETCON. OPR identifies commands to NETCON when you type "NCP" (Network Control Program). You may type NCP followed by the NCP command as shown below:

```
$OPR
OPR>NCP SHOW STATUS LOCAL
OPR>
08:18:33        NCP request # 5 [SHOW STATUS LOCAL]

     Status of local node as of 17-May-79 08:18:33

     Node Name is 2102, # 64, System = 2102 Development System,
     TOPS-20
     Routing Version = 3.0.0, Communications Version = 3.0.0
     State is On, Default Host = Unknown

               Function Completed Successfully
```

or, you may ENTER NCP as shown below:

```
OPR>ENTER NCP
NCP>SHOW STATUS LOCAL
NCP>
08:19:10        NCP request # 6 [SHOW STATUS LOCAL]

     Status of local node as of 17-May-79 08:19:10

     Node Name is 2102, # 64, System = 2102 Development System,
     TOPS-20
     Routing Version = 3.0.0, Communications Version = 3.0.0
     State is on, Default Host = Unknown

               Function Completed Successfully

NCP>
```

OPR processes NCP commands for syntax only (for correct format and presence of required keywords, for example). Commands that are syntactically acceptable are passed by OPR to ORION, the operator control program. ORION, in turn, forwards the NCP commands to NETCON, the program that controls the processing and execution of NCP commands. Specifically, it is the NCP (processing) module of NETCON that receives the commands from ORION. When processing is complete, the NCP module passes all but the simplest commands to the NICE module on the executor node. Unless changed using the SET EXECUTOR command, the executor node will be the node where NETCON is running.

6.1 THE NETWORK CONTROL PROGRAM

DECnet-20 operations are controlled by the TOPS-20 program called NETCON. NETCON exists only as an executable file, NETCON.EXE. The control file and the three major source programs used in building NETCON are on the DECnet-20 distribution tape as NETCON.CTL, NETPAR.MAC, NCP.MAC, and NCU.MAC, respectively. Within the NETCON program, command processing is accomplished by the NCP module; command execution is accomplished by the NCP module or the NICE module, sometimes referred to as the Network Control Utility (NCU). As an operator you interact with NCP through OPR. The NCP module interacts with the NICE module.

NCP commands are directed:

- To the NETCON program itself
  (an example would be the SHOW EXECUTOR command)
- To the network
  (an example would be the LOAD REMOTE command)
- To a specific operating system and executed locally
  (an example would be SET LOCAL LOOPBACK ENABLED command)

It is the commands directed to the network that require the use of the Network Information and Control Exchange (NICE) protocol implemented in the NICE (NCU) module of NETCON. Through the use of a common protocol, NICE, the user is relieved of all concern with differences in implementations among the various families of DIGITAL computers. Any node that is a member of a given network can access the information and control facilities of a remote node on the same network if the following conditions are met:

- The node owning the information or facilities permits access
- The node owning the information or facilities has implemented the required NICE function

If a remote node is not able to honor a request, the system responds "? Request failed," followed by a message. A complete list of NCP command responses is given in Section 6.5.1.

Examples:

```
NCP>LOOP LINE KDP_0_0
NCP>
15:36:10          NCP request # 11 [LOOP LINE KDP__ 0 __0]
                  ?  Request failed, Invalid function or option

                  (The current executor was the host node.
                  Before the LOOP LINE command, the user should
                  have given the command SET EXECUTOR with the
                  node name of the DECnet-20 communications front
                  end.)

                        * * * * *

NCP>SHO COU LIN DTE_1
NCP>
15:23:41          NCP request # 8 [SHOW COUNTS LINE DTE20 __ 1]
                  ?  Request failed, Insufficient status

                  (The line id should have been a valid lineid on
                  the executor node.)
```

Not all DECnet systems implement all possible functions. Note that communication is still possible within the restrictions of the set of commands implemented by the remote node. You can refer to the manuals specific to the operating systems of the nodes in your network to determine the capabilities of their operating systems. (Within operating systems further restrictions are possible due to insufficient memory.)

At the network planning level, the system manager may find it advisable to limit some of the available NCP commands to specific privileged users or groups.

The tasks that NETCON performs on any given system depends on the configuration of the specific system. Given the device drivers and optional hardware required, NETCON can perform the following tasks.

For the DECSYSTEM-2020, NETCON can:

- Load a synchronous line controller
- Dump a synchronous line controller
- Control the state of a synchronous line
- Display the status of NCP requests
- Display line status and activity
- Control the logging of network events and line counters in the SYSERR file
- Start a line in cable-loopback or controller-loopback mode

For the DECSYSTEM-2040, 2050, 2060, NETCON can:

- Set up the load and dump parameters for the communications front end
- Set up the load and dump parameters for a remote job entry station
- Load a communications front end
- Load a remote job entry station
- Dump a communications front end
- Dump a remote job entry station
- Display the status of NCP requests
- Display line status and activity
- Control the logging of network events and line counters in the SYSERR file
- Turn event/error logging ON/OFF at either the line level (line drivers logging messages to NICE) or the NICE level (forwarding of messages to host)
- Perform loopback test on designated line
- Define an adjacent node other than the host node as the node that will load or dump a target node
- Specify the protocol to be used by the communicating nodes

6.2 DECnet-20 STARTUP PROCEDURES

The procedure for loading and starting DECnet-20 depends on the model of the DECSYSTEM-20.

On the DECSYSTEM-2040, 2050, 2060, there is a communications front-end processor that must be loaded. This occurs automatically when TOPS-20 is loaded or it can be performed manually at any time after TOPS-20 is loaded.

On the DECSYSTEM-2020, there is no separate DECnet-20 communications front-end processor to be loaded. The standard DECSYSTEM-2020 monitor supports DECnet. Therefore, whenever you load TOPS-20, you are also automatically loading DECnet-20 support code and support for the synchronous line controller.

6.2.1 Automatic Startup on a DECSYSTEM-2020

The automatic startup of DECnet-20 on a DECSYSTEM-2020 is merely a function of loading the TOPS-20 monitor. The DECnet code is integrated with the TOPS-20 code. Figure 6-1 is a sample of what you might see when bringing up TOPS-20 with DECnet-20 on a DECSYSTEM-2020. Note that the ORION program runs under SYSJOB, the NETCON program runs as a separate job, and the OPR subjob runs under PTYCON.

```
(Mount the PS:  disk pack on drive 0.  Then press the BOOT  button  on
the DECSYSTEM-2020 control panel.)

BT SW


[PS MOUNTED]

System restarting, wait...
ENTER CURRENT DATE AND TIME: 10-OCT-79 900

YOU HAVE ENTERED WEDNESDAY, 10-OCTOBER-1979 9:00AM,
 IS THIS CORRECT (Y,N) Y
WHY RELOAD? T/S

RUN CHECKD? N

RUNNING DDMP


SYSJOB 4(10) STARTED AT 10-OCT-79 0900
RUN SYS:ORION
RUN SYS:QUASAR
RUN SYS:MOUNTR
RUN SYS:INFO
RUN SYS:MAILER
RUN SYS:MAPPER
RUN SYS:LPTSPL
RUN SYS:CDRIVE
RUN SYS:SPRINT
RUN SYS:FAL
JOB 0 /LOG OPERATOR XX OPERATOR
ENA
^ESET LOGIN PSEUDO
^ESET LOGIN CONSOLE
^ESET OPERATOR
PTYCON
GET SYSTEM:PTYCON.ATO
/
JOB 1 /LOG OPERATOR XX OPERATOR
ENA
RUN SYS:BATCON
/
JOB 2 /LOG OPERATOR XX OPERATOR
ENA
RUN SYS:NETCON
/
```

Figure 6-1 Startup Dialogue for TOPS-20 DECnet-20 on a DECSYSTEM-2020

```
SJ  2:
SJ  1:
SJ  0:
SJ  0:  10/10-INSTALLATION-TEST SYSTEM, TOPS-20 MONITOR 4(3100)
SJ  1:  10/10-INSTALLATION-TEST SYSTEM, TOPS-20 MONITOR 4(3100)
SJ  2:  10/10-INSTALLATION-TEST SYSTEM, TOPS-20 MONITOR 4(3100)
SJ  0: @LOG OPERATOR OPERATOR
SJ  1: @LOG OPERATOR OPERATOR
SJ  2: @LOG OPERATOR OPERATOR
SJ  0:  JOB 1 ON TTY43 10-OCT-79 09:00:42
SJ  0: @ENA
SJ  1:  JOB 2 ON TTY44 10-OCT-79 09:00:42
SJ  2:  JOB 3 ON TTY45 10-OCT-79 09:00:43
SJ  0: $^ESET LOGIN PSEUDO
SJ  1: @ENA
SJ  0: $^ESET LOGIN CONSOLE
SJ  1: $RUN SYS:BATCON
SJ  0: $^ESET OPERATOR
SJ  0: $PTYCON
SJ  0: PTYCON> GET SYSTEM:PTYCON.ATO
SJ  0: PTYCON>SILENCE
SJ  2: @ENA
SJ  2: $RUN SYS:NETCON

[From OPERATOR on line 45: SYSTEM IN OPERATION]
SJ  0: PTYCON.LOG.1
SJ  0: PTYCON> W ALL
SJ  0: OPR(0)     3          OPERATOR   OPR        TI      0:0:6
SJ  0: PTYCON> CONN OPR
SJ  0: [CONNECTED TO SUBJOB OPR(0)]
```

Figure 6-1 (Cont.) Startup Dialogue for TOPS-20 DECnet-20 on a DECSYSTEM-2020

6.2.2 Automatic Startup on a DECSYSTEM-2040, 2050, 2060

On a DECSYSTEM-2040, 2050, 2060, the automatic startup of DECnet-20 involves a number of related operations.

One of the first programs executed when TOPS-20 is loaded is SYSJOB, a system process that starts up other system processes. SYSJOB executes the command file PS:<SYSTEM>SYSJOB.RUN. You edit this file to include startup commands for the NETCON program (see Chapter 10). NETCON has already been described at the beginning of this chapter. ORION is the operator's interface to NETCON. Network commands from the operator are parsed by the operator command parser (OPR) and then passed to ORION which, in turn, forwards them to NETCON.

The SYSJOB.RUN file also starts a PTYCON job and executes the PTYCON.ATO command file. In Chapter 10, the PTYCON.ATO file is edited to include starting the subjob OPR. OPR, in turn, executes the NCP.CMD file which sets up the DECnet-20 data base and loads the communications front-end. Figure 6-2 is an example of a startup dialogue for bringing up TOPS-20 with DECnet-20 on a DECSYSTEM 2040, 2050, or 2060. The example is taken from an operating system running TOPS-20 Version 4 with DECnet Version 2, including the RJE option. Prior to the example, the procedures you will follow in Chapters 9 and 10 (DECnet configuration, installation, and acceptance) have been accomplished. Startup procedures for your system may show differences according to your site's hardware and software; for example, the availability of extended addressing and cache memory hardware depends on the model selected for your site.

```
(Mount the PS:  disk pack on drive 0.  Then, press the  ENABLE  rocker
switch  while  pressing  the  DISK  rocker switch on the console front
end.)

RSX-20F YB13-41 6:34 21-SEP-79

[SY0: REDIRECTED TO DB0:]
[DB0: MOUNTED]
KLI -- VERSION VB12-12 RUNNING
KLI -- KL10 S/N: 2136., MODEL B, 60 HERTZ
KLI -- KL10 HARDWARE ENVIRONMENT:
        MOS MASTER OSCILLATOR
        EXTENDED ADDRESSING              (Note: Extended addressing
        INTERNAL CHANNELS                 is a Model B feature only.)
        CACHE                            (Note: Cache memory is available
                                          only on DECSYSTEMs-2050/2060.)
KLI -- MICROCODE VERSION 231 LOADED
KLI -- ALL CACHES ENABLED

LOGICAL MEMORY CONFIGURATION.
  ADDRESS   SIZE   INT   TYPE CONTROLLER
000000000   128K    4    MA20  0__1
004000000   768K    4    MF20  11

KLI -- CONFIGURATION FILE WRITTEN
KLI -- BOOTSTRAP LOADED AND STARTED


[PS MOUNTED]

System restarting, wait...
ENTER CURRENT DATE AND TIME: 9-OCT-79 1204

YOU HAVE ENTERED TUESDAY, 9-OCTOBER-1979 12:04PM,
 IS THIS CORRECT (Y,N) Y
WHY RELOAD? T/S

RUN CHECKD? N

RUNNING DDMP


SYSJOB 4(10) STARTED AT 9-OCT-79 1204
RUN SYS:ORION
RUN SYS:QUASAR

*****
 9-OCT-79 12:04:39 - TGHA V2 IS RUNNING FOR THE FIRST TIME.
*****
RUN SYS:MOUNTR
RUN SYS:INFO
RUN SYS:MAILER
RUN SYS:MAPPER
RUN SYS:LPTSPL                       (Note: this example shows that
RUN SYS:LPTSPL                        three line printer spoolers are
RUN SYS:LPTSPL                        controlled by this system's
RUN SYS:CDRIVE                        monitor--one for RJE station.)
RUN SYS:SPRINT
RUN SYS:FAL
```

Figure 6-2 Startup Dialogue for TOPS-20 DECnet-20 on a DECSYSTEM-2040, 2050, 2060

```
JOB 0 /LOG OPERATOR XX OPERATOR
ENA
^ESET LOGIN PSEUDO
^ESET LOGIN CONSOLE

^ESET OPERATOR
PTYCON
GET SYSTEM:PTYCON.ATO
/
JOB 1 /LOG OPERATOR XX OPERATOR
ENA
RUN SYS:BATCON
/
JOB 2 /LOG OPERATOR XX OPERATOR
ENA
RUN SYS:NETCON
/
SJ  0:
SJ  0:  10/9-INSTALLATION-TEST SYSTEM, TOPS-20 MONITOR 4(3117)
SJ  1:
SJ  1:  10/9-INSTALLATION-TEST SYSTEM, TOPS-20 MONITOR 4(3117)
SJ  2:
SJ  2:  10/9-INSTALLATION-TEST SYSTEM, TOPS-20 MONITOR 4(3117)

*
SJ  2: @LOG OPERATOR OPERATOR
SJ  1: @LOG OPERATOR OPERATOR
SJ  0: @LOG OPERATOR OPERATOR
SJ  2:  JOB 2 ON TTY207 9-OCT-79 12:04:51
SJ  1:  JOB 1 ON TTY206 9-OCT-79 12:04:52
SJ  0:  JOB 3 ON TTY205 9-OCT-79 12:04:53
SJ  2: @ENA
SJ  1: @ENA
SJ  0: @ENA
SJ  2: $RUN SYS:NETCON
SJ  1: $RUN SYS:BATCON
SJ  0: $^ESET LOGIN PSEUDO
SJ  1: $^ESET LOGIN CONSOLE
SJ  0: $^ESET OPERATOR
SJ  0: $PTYCON
SJ  0: PTYCON> GET SYSTEM:PTYCON.ATO
SJ  0: PTYCON> SILENCE

[From OPERATOR on line 210: SYSTEM IN OPERATION]
SJ  0: PTYCON.LOG.1
SJ  0: PTYCON> W ALL
SJ  0: OPR(0)     3          OPERATOR   OPR        TI       0:0:1
SJ  0: PTYCON> CONN OPR
SJ  0: [CONNECTED TO SUBJOB OPR(0)]
SJ  2:  0:29:45         NCP REQUEST   1 [LOAD NODE DN20A]
SJ  2:                  FUNCTION COMPLETED SUCCESSFULLY
```

Figure 6-2 (Cont.) Startup Dialogue for TOPS-20 DECnet-20 on a DECSYSTEM-2040, 2050, 2060

6.2.3 Manual Startup on a DECSYSTEM-2020

To manually start DECnet-20 on a DECSYSTEM-2020, you must load the KMC11 line controller with the file COMIOP.KMC. This file was restored to the directory <SUBSYS> from the DECnet-20 distribution tape during the installation procedure in Chapter 10. From an enabled TOPS-20 prompt, enter the following dialogue:

```
$OPR(RET)
OPR>ENTER NCP(RET)
NCP>LOAD CONTROLLER KDP_0 FROM PS:<SUBSYS>COMIOP.KMC(RET)
NCP>EXIT(RET)
$
```

6.2.4 Manual Startup on a DECSYSTEM-2040, 2050, 2060

To manually start DECnet-20 on a DECSYSTEM-2040, 2050, 2060, you must load the communications front end with DECnet-20 software. The following sequence of commands identifies the load and dump files, loads the front end, and starts logging line statistics. From an enabled TOPS-20 prompt, enter the following dialogue:

```
$OPR(RET)
OPR>ENTER NCP(RET)
NCP>SET EXECUTOR hostnodename(RET)
NCP>SET SECONDARY-LOAD-FILE DTE20 PDP-11 PS:<SUBSYS>DTEMPS.BIN(RET)
NCP>SET TERTIARY-LOAD-FILE DTE20 PDP-11 PS:<SUBSYS>DTEMPT.BIN(RET)
NCP>SET NODE nodename SERVER servername DTE20_1(RET)
NCP>SET NODE nodename PROTOCOL-TYPE NETWORK-SERVICES-PROTOCOL(RET)
NCP>SET NODE nodename LOAD-FILE PS:<SUBSYS>nodename.SYS(RET)
NCP>SET NODE nodename DUMP-FILE PS:<SUBSYS>nodename.DMP(RET)
NCP>LOAD NODE nodename(RET)
NCP>
hh:mm:ss NCP REQUEST # n [LOAD NODE nodename]
     Function completed successfully
NCP>INITIATE LOGGING LINE-COUNTERS KDP_0_0 nodename(RET)
NCP>EXIT(RET)
$
```

where:

hostnodename	is the node name assigned to the main or central processor.
nodename	is the node name assigned to the communications front end. See Section 9.4, Configuring for Task-to-Task.
servername	is the node name assigned to the host processor. See the NODE command in the editing of the 4-CONFIG.CMD file in Chapter 10.

NOTE

If the front-end data base is intact and correct as set up during the loading of TOPS-20, you need only enter the LOAD NODE nodename command to start DECnet-20.

6.3 NCP COMMAND SUBSET OF OPR

To use the NCP commands, you must first run the OPR program and access the NCP subset of commands. In order to run OPR you should be ENABLEd; if not, you will receive the following message:

```
@OPR(RET)

10:31:42          --Insufficient Privileges Enabled--
@
```

ENABLE before running OPR:

```
        (ESC)
@ENABLE (CAPABILITIES)(RET)
$OPR(RET)
OPR>ENTER (COMMAND SUBSET) NCP(RET)
NCP>
```

The NCP commands fall into three general categories:

- Data base commands - to build or change the load, dump, operating, and logging parameters in a node's network data base.
- Action commands - to load and dump nodes or lines, alter line states, control loopback testing, monitor network requests, and display line status and statistics.
- Miscellaneous commands - to change the location of the NCP command processor, and to turn EVENT/ERROR logging on or off.

Some NCP commands are system-dependent; in the following command descriptions, such commands are flagged with the appropriate model numbers.

NOTE

For compatibility with other DECnet systems, the keyword NODE in any NETCON command may be replaced with the keyword REMOTE.

The table that follows is an alphabetized list of available NCP commands with their applicable keywords and arguments. See the description of each command for system-dependencies.

Table 6-1
NCP Command Summary

Command		KEYWORDS and/or Arguments			
DISABLE	EVENT-LOGGING	{KNOWN / LINE}	LINES / lineid		
DUMP	CONTROLLER	ctlrid	filespec		
	NODE	nodename	[filespec]		
ENABLE	EVENT-LOGGING	{KNOWN / LINE}	LINES / lineid		
EXIT					
INITIATE	LOGGING	LINE-COUNTERS	lineid	nodename	
LOAD	CONTROLLER	ctlrid	filespec		
	NODE	nodename	[filespec]		
*LOOP	LINE	lineid	COUNT xx	LENGTH yy	WITH {MIXED / ZEROES / ONES}
PUSH					
RETURN					
SET	EXECUTOR	nodename			
	LOCAL	LOOPBACK	{ENABLED / DISABLED}	lineid	
	LOGGING-INTERVAL	LINE-COUNTERS	nminutes		
	NODE	nodename	AUTO-DUMP	{ENABLED / DISABLED}	
			AUTO-LOAD	{ENABLED / DISABLED}	
			DUMP-FILE	filespec	
			LOAD-FILE	filespec	
			PROTOCOL-TYPE	NETWORK-SERVICES-PROTOCOL	
			SERVER	servernode	lineid
	SECONDARY-LOAD-FILE	devtype	cputype	filespec	
	STATE	LINE	lineid	{CABLE-LOOPBACK / CONTROLLER-LOOPBACK / MAINTENANCE / ON / OFF}	
	TERTIARY-LOAD-FILE	devtype	cputype	filespec	
SHOW	COUNTS	LINE	lineid		
	EXECUTOR				
	QUEUE	NCP-REQUESTS			
	STATUS	{KNOWN / LINE / LOCAL}	LINES / lineid		
TAKE		filespec			
TERMINATE	LOGGING	LINE-COUNTERS	lineid	nodename	

*The LOOP LINE command may contain any combination of the three keywords COUNT, LENGTH, and WITH.

6.3.1 Data Base Commands

6.3.1.1 **Defining the Bootstrap Programs (2040, 2050, 2060) -** The following two commands specify the secondary and tertiary bootstrap programs to be used when a request to load a node is received.

The format of the commands is:

```
SET  SECONDARY-LOAD-FILE  devtype cputype filespec
SET  TERTIARY-LOAD-FILE   devtype cputype filespec
```

where:

devtype specifies the type of line device (on the target node) over which NETCON will be loading the bootstrap program. This can be the DTE20 or DMC11.

cputype specifies the type of CPU the bootstrap program will be loaded into. This is the PDP-11.

filespec specifies the appropriate secondary or tertiary bootstrap program to be loaded.

If the node to be loaded is a communications front end, use the following commands:

```
SET SECONDARY-LOAD-FILE DTE20 PDP-11 PS:<SUBSYS>DTEMPS.BIN
SET TERTIARY-LOAD-FILE DTE20 PDP-11 PS:<SUBSYS>DTEMPT.BIN
```

If the node to be loaded is a remote job entry station such as a DN200, use the following commands:

```
SET SECONDARY-LOAD-FILE DMC11 PDP-11 PS:<SUBSYS>SECDMC.SYS
SET TERTIARY-LOAD-FILE DMC11 PDP-11 PS:<SUBSYS>TERDMC.SYS
```

6.3.1.2 **Defining the Loading Node (2040, 2050, 2060) -** The SET NODE SERVER command is used to specify the adjacent node that will perform the load or dump operation.

The format of the SET NODE SERVER command is:

```
SET NODE targetnode SERVER servernode serverlineid
```

where:

targetnode is the name of the node that will be loaded or dumped.

servernode is the name of the node adjacent to the targetnode that will perform the loading or dumping.

serverlineid — is the identification of the line or interface on the servernode that connects to the targetnode. The format of serverlineid is:

dev_ctlr_unit

where:

dev	is the type of device on the servernode that interfaces or connects to the targetnode. This can be DTE20, DMC11, DUP11, or KDP. (DUP11s are not supported for all configurations.)
(underscore)	is a required separator character.
ctlr	is the number of the specific device controller.
(underscore)	is an optional separator character, required if unit is specified.
unit	is the specific line unit on a KDP device.

6.3.1.3 **Defining the Default Load File (2040, 2050, 2060)** - The SET NODE LOAD-FILE command is used to specify the default load file for a node.

The format of the SET NODE LOAD-FILE command is:

```
SET NODE targetnode LOAD-FILE PS:<SUBSYS>targetnode.SYS
```

where:

targetnode — is the name of the node that is to be loaded.

6.3.1.4 **Defining the Default Dump File (2040, 2050, 2060)** - The SET NODE DUMP-FILE command is used to specify a default dump file for a node.

The format of the SET NODE DUMP-FILE command is:

```
SET NODE targetnode DUMP-FILE PS:<SUBSYS>targetnode.DMP
```

where:

targetnode — is the name of the node that is to be dumped.

6.3.1.5 **Defining the Protocol (2040, 2050, 2060)** - The SET NODE PROTOCOL-TYPE command is used to specify the protocol to be followed by nodes when communicating.

The format of the SET NODE PROTOCOL-TYPE command is:

```
SET NODE targetnode PROTOCOL-TYPE protocoltype
```

where:

targetnode — is the name of the node that is being loaded or dumped.

protocoltype — is the specific protocol to be followed by the target and server nodes when communicating.

NOTE

Version 2 of DECnet-20 restricts the following command variable as shown:

protocoltype NETWORK-SERVICES-PROTOCOL
RSX20F-QUEUED-PROTOCOL
(Only NETWORK-SERVICES-PROTOCOL is currently supported.)

6.3.1.6 **Initiating Logging (all models)** - The INITIATE LOGGING command is used to enable the periodic recording of line activity at a node. For a DECSYSTEM-2040/50/60 this command is restricted to the DN20-side of the DTE. This means that the nodename parameter can not be the host's node name. (See Section 6.3.2.4.) The line statistics are requested by NETCON and are recorded in the form of SYSERR entries to the TOPS-20 ERROR.SYS file. SYSERR entries are described in detail in Appendix C.

The format of the INITIATE LOGGING command is:

```
INITIATE LOGGING LINE-COUNTERS (FOR LINE) lineid (ON NODE) nodename
```

where:

lineid — identifies the line for which the statistics are being recorded. The format of lineid is:

```
dev_ctlr_uni
```

where:

dev — is the device type: DMC11, DUP11, KDP, or DTE20.

ctlr — is the controller number.

uni — is the unit number.

nodename — is the name of the node at the particular end of the line being recorded. This means that a line may have its statistics reported from either end.

Both lineid and nodename are required entries.

6.3.1.7 **Terminating Logging (all models)** - The TERMINATE LOGGING command is used to stop the recording of line activity at the specified line and node.

The format of the TERMINATE LOGGING command is:

TERMINATE LOGGING LINE-COUNTERS (FOR LINE) lineid (ON NODE) nodename

See Section 6.3.1.6 for a definition of lineid and nodename.

6.3.1.8 **Setting Logging Interval (all models)** - The SET LOGGING-INTERVAL command is used to specify how often NETCON will log line statistics. This interval applies to all lines enabled for logging. If this command is not given, the default interval is 60 minutes.

The format of the SET LOGGING-INTERVAL command is:

SET LOGGING-INTERVAL (FOR) LINE-COUNTERS (TO) n (MINUTES)

where:

n is the number of minutes between requests for line counters.

6.3.1.9 **Enabling Local Loopback (2020)** - The SET LOCAL LOOPBACK ENABLED command is used to inform the local NSP that a particular line is going to be operated in loopback mode. One effect of this is that data being exchanged over local logical links (connections between tasks on the same node) is not routed entirely in the software but is actually sent to the line controller. Whether the data is reflected in the controller itself or in the cable beyond is determined by the SET STATE LINE commands described in Sections 6.3.2.9 and 6.3.2.10.

Another effect of the SET LOCAL LOOPBACK ENABLED command is that the local NSP now expects to receive a reflection of any node initialization message sent when the line is activated.

The format of the SET LOCAL LOOPBACK ENABLED command is:

SET LOCAL LOOPBACK ENABLED lineid

where:

lineid is a required entry and identifies the line to be operated in loopback mode. The format of lineid is described in Section 6.3.1.6.

NOTE

The line identified by lineid must be turned off before issuing this command. See the SET STATE LINE OFF command in Section 6.3.2.11.

6.3.1.10 **Disabling Local Loopback (2020)** - The SET LOCAL LOOPBACK DISABLED command is used to inform the local NSP that a particular line is no longer being operated in loopback mode. The effects of loopback operation described in Section 6.2.1.9 are no longer in effect.

The format of the SET LOCAL LOOPBACK DISABLED command is:

```
SET LOCAL LOOPBACK DISABLED lineid
```

where:

lineid is a required entry and identifies the line that is no longer in loopback mode. The format of lineid is described in Section 6.3.1.6.

NOTE

The line identified by lineid must be turned off before issuing this command. See the SET STATE LINE OFF command in Section 6.3.2.11.

6.3.1.11 **Controlling Automatic Reloading (2040,2050,2060)** - The SET NODE AUTO-LOAD command controls whether or not the communications front end is automatically reloaded after a front-end crash.

The format of the SET NODE AUTO-LOAD command is:

```
SET NODE nodename AUTO-LOAD {ENABLED  }
                            {DISABLED }
```

where:

nodename identifies the DECnet-20 communications front end for which you are setting the autoload control.

ENABLED causes the DECnet-20 communications front end to be automatically reloaded after a front-end crash. When the system is initially loaded, ENABLED is in effect.

DISABLED causes the DECnet-20 communications front end not to be automatically reloaded after a front-end crash.

6.3.1.12 **Controlling Automatic Dumping (2040,2050,2060)** - The SET NODE AUTO-DUMP command controls whether or not the communications front end is automatically dumped after a front-end crash.

The format of the SET NODE AUTO-DUMP command is:

```
SET NODE nodename AUTO-DUMP{ENABLED }
                           {DISABLED}
```

where:

nodename — identifies the communications front end for which you are setting the autodump control.

ENABLED — causes the communications front end to be automatically dumped after a front-end crash. When the system is initially loaded, ENABLED is in effect.

DISABLED — cause the communications front end not to be automatically dumped after a front-end crash.

6.3.2 Action Commands

6.3.2.1 **Loading a Node (2040, 2050, 2060)** - The LOAD NODE command loads and starts a target node with a specified or default program file.

The format of the LOAD NODE command is:

```
LOAD NODE targetnode [FROM (FILE) filespec]
```

where:

targetnode — is the name of the node that is to be loaded.

filespec — is the file specification for the specific program to be loaded. If filespec is not included, the default load file specification defined in Section 6.3.1.3 is used. If neither file specification is specified, the system replies:

```
REQUEST #n [LOAD NODE targetnode]
CONFIGURATION DATA BASE ERROR
```

where n is the NCP request number that NETCON assigned to the LOAD NODE command.

NOTE

Version 2 of DECnet-20 restricts the following command variable as shown:

targetnode - the node name assigned to the communications front end.

6.3.2.2 **Dumping a Node (2040, 2050, 2060)** - The DUMP NODE command initiates a memory dump operation.

The format of the DUMP NODE command is:

```
DUMP NODE targetnode [TO filespec]
```

where:

targetnode — is the name of the node that is to be dumped.

filespec — is the file specification for the dump file. If filespec is not included, the default dump file defined in Section 6.3.1.4 is used.

NOTE

Version 2 of DECnet-20 restricts the following command variable as shown:

targetnode - the node name assigned to the communications front end.

6.3.2.3 **Checking NETCON's Request Queue (all models)** - The SHOW QUEUE command checks NETCON's queue of network requests and displays both those that are currently in process and those not yet started.

The format of the SHOW QUEUE command is:

```
SHOW QUEUE (OF) NCP-REQUESTS
```

The response will first list those requests that are in progress and then those not started yet, if any.

```
hh:mm:ss
NCP requests in progress:
   #3 - DUMP NODE DN20
NCP requests on the queue but not yet started:
   #4 - LOAD NODE DN20
```

If there are no outstanding requests, the response is:

```
hh:mm:ss  There are no outstanding NCP requests.
```

6.3.2.4 **Displaying Line Statistics (all models)** - The SHOW COUNTS command displays line statistics for a specified line. For a DECSYSTEM-2040/50/60, the capability of displaying line statistics is limited to the DN20. A request to display line statistics when the executor is the host node will receive an error response as described in Section 6.1. See Section 6.3.3.1 for details on changing the executor of a NETCON command.

The format of the SHOW COUNTS command is:

```
SHOW COUNTS LINE lineid
```

where the format of lineid is the same as in Section 6.3.1.6.

Assuming you are logged in at the host node (a DECSYSTEM-2050, for example), the following is a possible example:

```
NCP>SET EXECUTOR DN20A
NCP>SHOW COUNTS LINE DMC11_1_0
NCP>
10:30:46          NCP request # 32 [SHOW COUNTS LINE DMC11_1_0]

     Counts for line DMC11_1_0_0, as of 5-Jun-79 10:30:45
     Seconds since zeroed          2488
     Blocks received                978
     Blocks sent                    978
     Retran, line errors              3
     Receive line errors              0
     Retran, not line errors          0
     Receive timeouts                 3
               Function completed successfully
NCP>
```

6.3.2.5 **Exiting from NCP Subset (all models)** - The RETURN command returns you to operator command level, OPR>. Typing EXIT or CTRL/C to the OPR> prompt then returns you to TOPS-20 command level. If you wish, you may type a CTRL/C to the NCP> prompt and return directly to TOPS-20 command level. Typing EXIT will also return you to TOPS-20 command level.

6.3.2.6 **Loading a Line Controller (2020)** - The LOAD CONTROLLER command loads and starts a communications line controller from a specified file.

The format of the LOAD CONTROLLER command is:

```
LOAD CONTROLLER ctlrid (FROM) filespec
```

where:

ctlrid identifies the controller being loaded. The format of ctlrid is:

```
dev_ctlr
```

where:

dev is the device type, KDP

ctlr is the controller number

filespec is the file specification for the specific program to be loaded. The distributed program is COMIOP.KMC.

Both ctlrid and filespec are required entries.

6.3.2.7 **Dumping a Line Controller (2020)** - The DUMP CONTROLLER command initiates a memory dump of a communications line controller to a specified file.

The format of the DUMP CONTROLLER command is:

```
DUMP CONTROLLER ctlrid (TO) filespec
```

where:

ctlrid identifies the controller being dumped. The format of ctlrid is described in Section 6.3.2.6.

filespec is the identification of the file receiving the memory dump.

Both ctlrid and filespec are required entries.

6.3.2.8 **Starting a Line (all models)** - The SET STATE LINE ON command initializes the DDCMP line protocol on a specified line. The local NSP, notified that the line is active, sends a node initialization message over the line.

The format of the SET STATE LINE ON command is:

```
SET STATE LINE lineid ON
```

where:

lineid is a required entry and identifies the line being started. The format of lineid is described in Section 6.3.1.6.

6.3.2.9 **Starting a Line for Controller Loopback (2020)** - The SET STATE LINE CONTROLLER-LOOPBACK command initializes the DDCMP line protocol and results in a node initialization message being sent as in Section 6.3.2.8. In addition, the line controller represented by the devctlr portion of lineid is notified to internally reflect any messages that are directed to the specified line.

The format of the SET STATE LINE CONTROLLER-LOOPBACK command is:

```
SET STATE LINE lineid CONTROLLER-LOOPBACK
```

where:

lineid is a required entry and identifies the line that is to be started in controller loopback mode. The format of lineid is described in Section 6.3.1.6.

6.3.2.10 **Starting a Line for Cable Loopback (2020)** - The SET STATE LINE CABLE-LOOPBACK command initializes the DDCMP line protocol and results in a node initialization message being sent as in Section 6.3.2.8. When a loopback connector is placed on the line to reflect data, the synchronous line unit's internal clock provides the necessary synchronization to correctly receive the reflected data, replacing the modem clock.

The format of the SET STATE LINE CABLE-LOOPBACK command is:

```
SET STATE LINE lineid CABLE-LOOPBACK
```

where:

lineid — is a required entry and identifies the line that is to be started in cable loopback mode. The format of lineid is described in Section 6.3.1.6.

6.3.2.11 **Stopping a Line (all models)** - The SET STATE LINE OFF command terminates the DDCMP line protocol on a specified line. The line is no longer available for transmitting data.

The format of the SET STATE LINE OFF command is:

```
SET STATE LINE lineid OFF
```

where:

lineid — is a required entry and identifies the line being deactivated. The format of lineid is described in Section 6.3.1.6.

6.3.2.12 **Performing a Loopback Test (2040,2050,2060)** - The LOOP LINE command performs a loopback test on a specified line on the current EXECUTOR node. The communications front end should be the current EXECUTOR. To determine the current EXECUTOR node, use the SHOW EXECUTOR command (Section 6.3.3.2). To change the current EXECUTOR node, use the SET EXECUTOR command (Section 6.3.3.1).

The format of the LOOP LINE command is:

```
LOOP LINE lineid COUNT count LENGTH length WITH { ZEROES }
                                                { MIXED  }
                                                { ONES   }
```

where:

lineid — specifies the line being tested; see Section 6.3.1.6 for the definition of lineid.

count — Specifies the number of blocks of data to be used in the test. This number must be entered in octal.

length — specifies the length of each block of test data.

ZEROES, MIXED, ONES — specifies that the test data will be made up of all zeroes, all ones, or mixed ones and zeroes. One of these keywords must be entered.

6.3.2.13 **Interrogating Node Status (all models)** - The SHOW STATUS LOCAL commands display the status of the node that is processing the command. The node that is processing the command (EXECUTOR node) can be determined by using the SHOW EXECUTOR command (Section 6.3.3.1). Use the SET EXECUTOR command (Section 6.3.3.1) to change the EXECUTOR node.

The format of the SHOW STATUS LOCAL command is:

```
SHOW STATUS LOCAL
```

The response includes date, time, node name, node number, the operating system, the routing and communications versions of NSP being used, the state of the node, and the default host node, if known.

6.3.2.14 **Interrogating Line Status (all models)** - The SHOW STATUS LINE commands display the status of all or specific lines on the node that is processing the commands. The node that is processing the command (Executor node) can be determined by using the SHOW EXECUTOR command (Section 6.3.3.2). Use the SET EXECUTOR command (Section 6.3.3.1) to change the EXEUCTOR node.

The formats of the SHOW STATUS LINES commands are:

```
SHOW STATUS LINE lineid
          and
SHOW STATUS KNOWN LINES
```

where:

lineid — Identifies the specific line for which the status is being requested. See Section 6.3.1.6 for the definition of lineid.

KNOWN LINES — requests the status of all lines known to the EXECUTOR node. KNOWN LINES is not available for 32K DN20s.

6.3.3 Miscellaneous Commands

6.3.3.1 **Setting and Displaying Executor Node (all models)** - The SET EXECUTOR command specifies which node is to process subsequent commands. The default executor node at system startup is the physical node at which your login occurred. The SHOW EXECUTOR command indicates which node is currently processing your commands.

The format of the SET EXECUTOR command is:

```
SET EXECUTOR nodename
```

where:

nodename — is the name of the node that is to process subsequent NETCON commands.

The format of the SHOW EXECUTOR command is:

```
SHOW EXECUTOR
```

The response takes the form:

```
Current EXECUTOR is node nodename
```

6.3.3.2 **Turning Event/Error Logging On or Off (2040,2050,2060)** - In the DECnet communications front end (the DN20) the line drivers (DTE, KDP, DMC) report hardware errors and protocol error thresholds (DDCMP). Event/error logging is supported for DECSYSTEMs-2040/50/60 with 128K words in the DN20 DECnet-20 communications front end.

The NICE task in the DN20 acts as a concentrator for messages coming from all processes, and forwards these messages to NETCON in the host computer. Thus there are two ways to control the flow of event-logging messages: the drivers can be notified to send or not send messages to the NICE task in the DN20; the NICE task can be notified to forward or not forward the messages to NETCON.

To enable or disable event logging at the line level (messages no longer sent to the NICE task in the DN20), the command format is:

```
{ENABLE  }
{DISABLE } EVENT-LOGGING {LINE lineid  }
                         {KNOWN LINES }
```

To enable or disable the forwarding of event-logging messages to NETCON, the command format is:

```
{ENABLE  }
{DISABLE } EVENT-LOGGING
```

When a DISABLE EVENT-LOGGING command is given with a line parameter, the messages are lost from the time of DISABLE to the time you ENABLE EVENT-LOGGING. When a DISABLE EVENT-LOGGING command is given without a line parameter, all event/error messages are lost until another ENABLE EVENT-LOGGING command is given.

The command to enable or disable the forwarding of event-logging messages to NETCON is independent of the on/off setting of the lines.

6.3.3.3 **Getting Another Copy of the TOPS-20 Command Processor (all models)** - The PUSH command gives you another copy of the TOPS-20 command processor. The format is:

```
PUSH
```

6.3.3.4 **Executing NCP Commands From a File (all models)** - The TAKE command will accept any file specification of a file containing NCP commands. The default file type is .CMD. The format is:

```
TAKE filespec
```

6.4 RESTART PROCEDURES

The NETCON process can be restarted with the following command sequence:

```
@ENABLE(RET)
$ADVISE OPERATOR(RET)
 TTY213, EXEC
 TTY2, OPR
 TTY211, FAL
 TTY210, NETCON
 TTY207, BATCON
 TTY205, PTYCON
TTY: 210(RET)
 [Pseudo-terminal, confirm](RET)
 Escape character is <CTRL>E, type <CTRL>^? for help
 OPERATOR job 3 NETCON

 LINK FROM OPERATOR, tty 3
  [Advising]
  (CTRL/C)
 $RESET(RET)
 $RUN SYS:NETCON(RET)
  (CTRL/E) ;(Note: ^E does not echo on terminal)
  [Advice terminated]
 $DISABLE(RET)
 @
```

When NETCON is restarted all knowledge of the network is lost. Therefore you must repeat all commands to identify the various DN20 (and DN200) data base elements. (See Section 6.3.1.)

If the ORION process fails while the NETCON process is running, the following NETCON warning message is displayed:

```
% NETCON:  ORION is not running
```

Whenever this occurs, ORION must be reloaded using the ^ESPEAK command.

The ORION process can be restarted with the following command sequence:

```
@ENABLE(RET)                          ;enable capabilities
$^ESPEAK(RET)                         ;give ^ESPEAK command
 [PLEASE TYPE SYSJOB COMMANDS - END WITH ^Z] ;
KILL SYS:ORION(RET)
RUN SYS:ORION(RET)                    ;restart ORION
(CTRL/Z)                              ;end the ^ESPEAK command
$DISABLE(RET)                         ;disable capabilities
@
```

In addition, because the new copy of ORION has no knowledge of NETCON's location, NETCON must also be reloaded. The initialization routine in NETCON sends a message to ORION identifying itself.

6.5 OPERATOR MESSAGES

When running DECnet-20, there are a number of messages that can appear at the operator's console. Some of these abort NETCON and stop network processing; others are warnings and allow NETCON to continue. Still others merely confirm network commands. DECnet-20-related messages can be categorized as follows:

- Responses (normal or error) resulting from the execution of NETCON commands
- Warning messages due to temporary inhibitions to network operations
- Fatal error messages that reflect program problems severe enough to abort the NETCON process
- Error messages from the NETCON command parser (OPR) due to problems in parsing commands or forwarding commands through ORION to NETCON

6.5.1 NETCON Command Responses

All NCP commands that are processed by the NICE module of the NETCON program are assigned a Request Number and placed in the NCP-REQUEST queue. The user receives a meaningful response and the Request Number. Currently, some commands that are not processed by the NICE module respond only with the NCP prompt to indicate the program is ready for the next command. Examples of commands with the NCP prompt as the only response are SET EXECUTOR, SET LOGGING-INTERVAL, and SET LINE. All SHOW commands are followed by a response and time stamp (no Request Number).

Following are examples of each of the three types of responses.

```
NCP>set executor 2102
NCP>

(NCP prompt only)

NCP>show executor
NCP>
16:33:11                  Current EXECUTOR is node 2102
(Response and time stamp)

NCP>show sta kn li
NCP>
16:33:52          NCP request #23 [SHOW STATUS KNOWN LINES]

                Status as of 18-Oct-79 16:33:51

                Line ID         State           Adjacent Node

                DTE20_0_0       Off
                DTE20_1_0       Off
                DTE20_2_0       Off
                DTE20_3_0       Off
                  Function completed successfully

NCP>

(Response, Request Number, and time stamp)
```

If the NSP link disappears, the message that is output is:

NSP link died

If a command cannot be successfully completed, one of the following messages may be displayed and NCP will prompt for the next command.

Invalid function or option

NCU process not available

No room for new entry

In all of the following situations, NETCON continues to operate. If the condition cannot be remedied by retrying, refreshing data files, or correcting hardware problems, report it to the Software Services representative or submit a Software Performance Report (SPR).

File I/O error

This message is due to either a file read error or a disk write error. For a read error, try to restore a good copy of the file. For a write error, have the hardware problem corrected.

Line protocol error

DTE20 (or DN200) hardware problem occurred during a down-line load operation or a problem with the bootstrap ROM in the communications front end.

Configuration data base error

Invalid or missing entries in the network data base.

Invalid file

One of the load or bootstrap files cannot be found or is not in the correct format.

Invalid server id

Invalid node id

The command specifies a node name that does not follow the naming convention for nodes. (See Section 2.1.4.)

Line communications error

An error occurred in the DTE20 (or DN200) hardware or a system was generated that cannot fit in available memory.

Line in wrong state

Invalid line id

The line identification given as an argument in the NCP command is invalid. (See Section 6.3.1.2.)

Network communications error

Normally this message means that NETCON cannot communicate with the NICE process on a node. The problem could be at either end (resource problems, for example).

6.5.2 NETCON Warning Messages

NETCON warning messages are displayed whenever unexpected conditions arise during the execution of network control functions or network utilities. NETCON continues to run and the condition will either disappear due to improved system resources or will worsen to the point of a fatal error message. Warning messages are preceded by a line containing the time of day in the format hh:mm:ss.

% NETCON: Could not open any JFNs for the NCU task

% NETCON: Could not create a PID for receiving ORION messages

% NETCON: Could not send "HELLO" message to ORION

% NETCON: Could not set up PID interrupt channel

% NETCON: No more free pages available

% NETCON: Attempt to return an illegal page address

% NETCON: Attempt to return a page that is already on the free list

% NETCON: No more free space available

% NETCON: Attempt to return a free block outside the free pool area

% NETCON: Attempt to return space already marked as in the free pool

% NETCON: ORION is not running

% NETCON: Failed to successfully send a message to OPR

% NETCON: Unimplemented NETCON command

% NETCON: Unknown internal NETCON message type

% NETCON: Exhausted table space for the known nodes

% NETCON: All server JFNs in use

% NETCON: NCU process halted prematurely

% NETCON: Unknown message type received from ORION

% NETCON: JSYS error

% NETCON: Could not get name of local node

% NETCON: Could not log line counters for communications line

% NETCON: Could not initiate Logging Process

% NETCON: Invalid line-id

% NETCON: Could not open file

% NETCON: Bad data in load file

The foregoing warning messages reflect potential or actual program problems; they should be reported via Software Performance Reports (SPRs).

6.5.3 NETCON Fatal Error Messages

Whenever NETCON finds itself in a situation where it cannot effectively continue operation, a fatal error message is displayed at the operator's console. Fatal error conditions should be reported to the Software Services representative and documented via a Software Performance Report (SPR). The SPR should include the text of the error message as well as information with which the error can be reproduced.

The fatal error messages listed below are the result of some internal program error. Since NETCON is no longer running, any attempt at recovery must begin with reloading NETCON as described in Section 6.3.

? NETCON: Pushdown list overflow at location: nnnnnn

NETCON tried to add an item to the pushdown list and exceeded the specified list size.

? NETCON: Illegal memory read at location: nnnnnn

NETCON tried to read data from an area that appears to be inaccessible.

? NETCON: Illegal memory write at location: nnnnnn

NETCON tried to write data to an area that appears to be unavailable for writing.

? NETCON: Illegal instruction at location: nnnnnn

NETCON tried to execute what it thought was an instruction and found an invalid instruction code.

? NETCON: Machine size exceeded at location: nnnnnn

NETCON ran out of some monitor resource while executing a JSYS operation.

? NETCON: I/O data error or parity error at location: nnnnnn

NETCON encountered an unrecoverable data or parity error while reading a file or writing to disk.

? NETCON: Fatal error at location: nnnnnn

NETCON encountered an undetermined fatal error at the specified location.

? NETCON: Could not open topology server links.

This is possibly due to too many open network links.

Each of the above fatal error messages is followed by some specific monitor error message or the following:

(Unknown monitor error code)

6.5.4 Other Error Messages

Before a NETCON command is processed by NETCON, the command is handled by both the OPR and ORION programs. Each of these processes checks the command for syntax and intent.

The following error messages are issued by OPR due to errors detected by OPR. In each case, the command is ignored and OPR prompts for the next entry. Reenter the command correctly.

?Invalid command "text"

This message occurs when OPR does not recognize the initial word of the command line as a valid command name. "text" is the word that OPR tried to parse as a command name.

?Can't find file "filespec"

This message occurs when a file, specified in a command line, cannot be found. "filespec" is the actual string submitted to the GTJFN monitor call.

?Can't Open TAKE Command File

This message occurs when the command file specified in a TAKE command cannot be opened. (The OPENF monitor call failed.)

?Command error: string

This message occurs when a command line cannot be parsed. String is the text resulting from the translation of a JSYS error number by the ERSTR monitor call.

The following messages are issued by OPR due to errors detected by ORION. In each case, recovery is possible.

hh:mm:ss No Processor for Object

This message occurs if NETCON is not running or if ORION has been reloaded without reloading NETCON. Reload NETCON as described in Section 6.3 and reenter the command.

hh:mm:ss Insufficient Privileges Enabled

This message occurs if OPR is being run without having OPERATOR or WHEEL privileges enabled. The message is displayed when OPR is first invoked, and again for each command submitted to OPR. Obtain OPERATOR or WHEEL privileges, enable them (ENABLE CAPABILITIES command), and rerun OPR.

The following messages are also issued by OPR due to errors detected by ORION; however, the errors are due to communications problems between OPR and ORION and should be reported to the software specialist.

hh:mm:ss Message Too Short

ORION received a message from OPR that was shorter than the allowed minimum.

hh:mm:ss Bad Message Length

ORION received a message that was shorter than the length specified in the message.

hh:mm:ss Illegal Message Type

ORION received a message with an illegal message type code.

hh:mm:ss Wrong ORION Message Version Number

ORION received a message specifying a version number different from that of the running version.

hh:mm:ss Unidentified Processor Type

ORION received a command intended for a processor other than NETCON.

hh:mm:ss Illegally Formatted Command Message

ORION received a command that it did not understand.

If the OPR program is started at a time when ORION is not running, the initial message that OPR uses to identify itself to ORION cannot be sent. Information in the following format is displayed at the operator's console; the actual values will, of course, be different.

```
?Stop code - SOF - in module OPR on 13-Oct-77 at 15:04:06
     Reason:  Send to ORION Failed
     Program is OPR Version 1(12) using GLXLIB Version 1(20)
     Last GLXLIB error:  0 (No errors yet)
     Last TOPS-20 error:  601130 (Invalid index into system PID table)

Contents of the ACs (Crash block starts at location 600000)

 0/  777777777777          400000       612000616              0
 4/             0               0               0              0
10/             0               0               0              0
14/             0               0               0  777731001321

Last 9 Stack Locations:

-1(P)/                  0   -2(P)/              0    -3(P)/         0
-4(P)/                  0   -5(P)/              0    -6(P)/         0
-7(P)/                  0   -8(P)/              0    -9(P)/         0

HLT:401202/    HALTF
```

PART IV

TERMINAL USER'S GUIDE

CHAPTER 7

COMMUNICATING WITH THE SYSTEM

As a nonprivileged terminal user, you communicate with the system by using the TOPS-20 Command Language (EXEC). This chapter assumes that you are familiar with the most frequently used TOPS-20 commands (both for timesharing and Batch). TOPS-20 commands that relate directly to DECnet functions are described in this chapter.

NOTE

If you have had little experience with the TOPS-20 Command Language, refer to the list of suggested documents in the Preface of this manual. As an absolute minimum, you should read the following manuals before continuing with this and the following chapter:

TOPS-20 User's Guide
TOPS-20 Commands Reference Manual
TOPS-10 and TOPS-20 Batch Reference Manual

You should know, or have readily at hand, the node name of your central processor and the name of all remote nodes with whom you will communicate. If remote nodes require a user name, password, or account, you will need to know the specific way in which this information must be formatted. Your installation should have the basic user's manuals for all systems accessible to you via DECnet. If you need help, see your system manager or operator.

The TOPS-20 operating system in conjunction with DECnet software allows you to do the following:

- List accessible DECnet nodes using the TOPS-20 INFORMATION DECNET command.
- List your logical DECnet node using the TOPS-20 INFORMATION JOB-STATUS command.
- Direct output to accessible DECnet nodes using the TOPS-20 SET LOCATION command or the TOPS-20 DESTINATION-NODE: node switch for queue-class commands.

To delete files from an accessible DECnet node and to transfer files to or from an accessible DECnet node, use the Network File Transfer (NFT) program. (The NFT program is described in Chapter 8, Network File Transfer.)

7.1 INFORMATION COMMAND

The INFORMATION command has two options that give DECnet information: INFORMATION DECnet and INFORMATION JOB-STATUS.

7.1.1 Information DECnet

INFORMATION DECNET lists the accessible DECnet nodes.

The format of the INFORMATION DECNET command is as follows:

```
@INFORMATION (ABOUT) DECNET NODES
```

Example:

```
@inf(ESC)ORMATION (ABOUT) decnet(RET)

 Accessible DECNET nodes are:  1031, 2102,  4097,  DN200,  DN20A,
 DN20L, SYS880
@
```

(Note that NODES is assumed as default if omitted.)

7.1.2 Information Job-Status

INFORMATION JOB-STATUS lists your logical node if your logical node is not your physical node.

The format of the INFORMATION JOB-STATUS command is as follows:

```
@INFORMATION (ABOUT) JOB-STATUS
```

Example:

```
@INFORMATION (ESC) (ABOUT) JOB-STATUS(RET)
 Job 41, User SKOGLUND, MISC:<SKOGLUND>, Account 341, TTY225
 Located at 1031
@
```

7.2 SET LOCATION COMMAND

The SET LOCATION command instructs the TOPS-20 operating system to regard the specified node as your logical node. Your logical node determines where your queued output goes. When you log in, your logical node is the same as your physical node. Currently, this command is effective only when the specified node is controlled by the local host. For example, the SET LOCATION command is useful when you wish to send several messages or print requests to the remote station (DN200).

The format of the SET LOCATION command is:

```
@SET LOCATION (TO) node::
```

where:

node:: The name of the node that becomes your logical node. If no node name is entered, the node name defaults to the name of your physical node.

Example:

```
@set location dn200::(RET)
@print test.txt(RET)
[Job TEST Queued, Request-ID 550, Limit 27]
@
```

If you give the INFORMATION JOB-STATUS command immediately following the SET LOCATION command, you can check to be sure your logical location has changed before you continue:

```
        (ESC)
$set locATION (TO) dn200::(RET)
@i j(RET)
 Job 62, User CIRINO, Account 341, TTY106
Located at DN200
(The above example shows how abbreviation may be used.)
```

Remember that the request remains on the queue until it is honored. If it appears that the request is being ignored, use the INFORMATION DECNET command to see if the DN200 is still available. If it is available, repeat the INFORMATION DECNET command later; if it is not available, check with your operations staff if the job is critical. (The DN200 may require manual loading.)

Printed on the DN200 printer is:

```
This is a test file!

(the contents of the file TEST.TXT).
```

The SET LOCATION command may also be used to direct requests to the local site's remote station (DN200). If the operator is not at the terminal of the remote station, the response to your PLEASE request will be delayed. The example below shows the input and output at both the local site and the remote station:

Typed on user's terminal at host site:

```
@set location dn200::(RET)

  (ESC)                  (ESC)
@infORMATION (ABOUT) joB-STATUS(RET)

 Job 33, User CIRINO, Account 341, TTY106
Located at DN200
@please turn printer on(RET)

[Operator Notified at 15:21:09...Please wait for reply]
```

Output to console at remote site:

```
2102::OPR>
2102::
15:21:09   <8>   --Message from Timesharing User--
                 JOB 33 CIRINO at Terminal 106
                 PLEASE turn printer on
```

Input by remote operator:

```
2102::OPR>RESPOND 8 PRINTER IS ON(RET)
2102::OPR>
```

Answer received at host site:

```
[Operator Response Received at 15:21:58]
PRINTER IS ON
```

7.3 /DESTINATION-NODE SWITCH

The /DESTINATION-NODE switch is used with the PRINT command to direct output to the specified node. When this switch is used with the SUBMIT command, the log file is directed to the specified node.

The format for the /DESTINATION-NODE switch is as follows:

```
/DESTINATION-NODE:node::
```

where:

node:: The name of the node where output is directed.

Example:

```
@PRINT FOO.BAR/DESTINATION-NODE:ABC::(RET)
```

7.4 ADDITIONAL FEATURES AVAILABLE TO NONPRIVILEGED USERS

The File Transfer Program described in the next chapter can be run by a nonprivileged user.

Appendix F describes three programs that use DECnet-20's task-to-task communications capabilities. These programs, NRT20, NRTSRV, and STNRT, allow a privileged or nonprivileged user at a terminal to log in to a remote host on the same network as the user's local host. Appendix F describes the function and use of these three programs.

All users may use the SYSERR program to type or print network error and event logging reports. (See Section 10.3.7.)

CHAPTER 8

NETWORK FILE TRANSFER

8.1 OVERVIEW

The Network File Transfer (NFT) program allows you to access or delete files residing on DECnet hosts that provide network file access capabilities. These capabilities are usually provided by a File Access Listener (FAL) program. NFT deletes files from a remote host and transfers sequential files between your local host and a remote host; FAL checks for requests made by NFT. The NFT and FAL programs communicate using the Data Access Protocol (DAP).

All network file transfers must be direct requests between the local host and one remote host. Files can be transferred from your local host to a remote host or from a remote host to your local host.

The files deleted or transferred using NFT can be text, program, data, control, or any other sequential files.

DECnet-20 does not support network file spooling. Unless you are using the wildcard feature, you can make only one file transfer request at a time and that request must be for only one file to be transferred. (See Section 8.2.1.)

8.1.1 Specifying File Access Information

Each file deletion or transfer request must include a valid user identification, account, and password for the system to be accessed. The FAL at the remote host is responsible for verifying your access to the requested file and the subsequent honoring or rejecting of your request. The requirements of the remote node determine the values you specify in access information switches or in response to prompts for access information. This security measure is necessary to protect a node's files from unauthorized access or accidental deletion. You must enter either the particular access parameter required by the remote node, or a carriage return if the remote node does not require a parameter or has an established default value.

NFT prompts you for access information (user, account, password) when you type the first NFT command that requires such information. If the access is successful, all subsequent file requests to the node addressed will use the access information that you provided in response to the prompt. If you type the SET DEFAULTS command as the first NFT command, or if you have set defaults for the node in an NFT.INIT file in your logged-in directory, you will not be prompted for access information. Whether you supply access information in response to a prompt from NFT, by the SET DEFAULTS command typed to your terminal, or by SET DEFAULTS commands in an NFT.INIT file, the values will be remembered. (The NFT.INIT file is read each time you run NFT.)

Access information entered in response to a prompt or with a SET DEFAULTS command remains effective until changed with another SET DEFAULTS command. Access information switches are used to override default values already established. Switch values are effective only for the command in which entered.

8.2 NFT COMMANDS

You call the NFT program by typing NFT or R NFT in response to the TOPS-20 operating system prompt. After you type NFT and press RETURN, the NFT program prints the prompt NFT>. If you want to see the list of valid NFT commands, type the question mark character (?).

Example:

```
@NFT(RET)
NFT>? one of the following:
 COPY          DELETE     DIRECTORY      EXIT           HELP
 INFORMATION   SET        SUBMIT         TAKE           TYPE
NFT>
```

NFT also offers the recognition feature of TOPS-20. If you type an NFT command after the NFT prompt and then press the ESCape key, NFT displays one or more guidewords to prompt you for the next response. Guidewords are enclosed in parentheses. After a guideword, type the question mark character (?) to display the list of valid responses. Additional guidewords and valid responses can be shown by repeating the alternate use of the ESCape key and the question mark character until all the input requirements for that command are satisfied.

The file specifications for remote files must have the format required by the operating system at the remote host. The operating systems and corresponding formats include the following:

Operating system	File Specifications Format
TOPS-20	device:<directory>filnam.ext.gen
VAX/VMS	device:[username]filnam.ext;gen
RSX, RSTS and IAS	device:[UIC]filnam.ext;gen

8.2.1 SET DEFAULTS Command

The SET DEFAULTS command establishes the default access information to be used with all subsequent NFT commands for the specified node. The values established with the SET DEFAULTS command remain in effect until you exit from NFT or change the values with another SET DEFAULTS command. The SET DEFAULTS command does not prompt for omitted information. However, NFT does prompt for required omitted access information switches at the time you type the first command that requires a switch not previously set with a SET DEFAULTS command.

Access information values may be changed for a specific command by including the desired access information switch. After the command containing the switch has been executed, the access information values revert to the previously established default values.

The format of the SET DEFAULTS command is as follows:

```
NFT>SET DEFAULTS (FOR NODE) node::/switches
```

where:

node::	is the node name to which the default values are assigned.
/switches	are any combination of access information switches (see Table 8-1) and, in addition, the switch /OSTYPE:operating system. Valid values for the operating-system argument are TOPS20, RSX11, RSTS, VMS, or IAS.

Examples:

```
@NFT<RET>
                  <ESC>
NFT>SET DEFAULTS (FOR NODE) ALPHA::/USER:JONES/OSTYPE:VMS<RET>

                  <ESC>
NFT>SET DEFAULTS (FOR NODE) DELTA::/USER:CLEMENS/PASSWORD:TOPS20<RET>

                  <ESC>
NFT>SET DEFAULTS (FOR NODE) GAMMA::/USER:REILLY/ACC:UETP<RET>
NFT>
```

You can place SET DEFAULTS commands in an initialization file that will be read when you run NFT. The initialization file must be in your logged-in directory and must be called NFT.INIT. Values set in an NFT.INIT file are effective until they are changed in the NFT.INIT file. Access information established in the NFT.INIT file may be changed by typing a SET DEFAULTS command, or may be overridden by an access information switch. The SET DEFAULTS command you type at the terminal will be effective for the current NFT run unless you change or override it. The access information switch is effective only for the command in which it is specified.

Following is an example of an NFT.INIT file currently in use. The INFORMATION DECNET command included at the end provides a convenient way to check the currently available nodes whenever you run NFT.

```
@type ps:<PTAYLOR>nft.init<RET>
SET DEFAULTS  2102::/USER:PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
SET DEFAULTS  1031::/USER:PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
SET DEFAULTS  SYS880::/USER:GUEST/ACCOUNT:FOO/OSTYPE:RSX11/PASS:DUMB
SET DEFAULTS  4097::/USER:REILLY.PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
SET DEFAULTS  2137::/USER:REILLY/ACC:UETP
INFORMATION DECNET
```

You may clear the default access information for a node by typing the SET DEFAULTS command without specifying any access information. To clear the default access information for a node other than the node where your logged-in directory is located, the node name must be included. If you omit node name, the system assumes the node to be the node where your logged-in directory is located.

Two examples are shown below. In Example 1 there is no NFT.INIT file in the logged-in directory. In Example 2 there is an NFT.INIT file with default access information for four nodes as shown. The NFT.INIT file also includes the INFORMATION DECNET command. Including this command in the NFT.INIT file saves you from having to type the command at the beginning of each NFT run. Note in the first example that the default access information is available for the node where your logged-in directory is located even though you have not given a SET DEFAULTS command. NFT establishes these parameters each time the NFT or R NFT command is given.

When you exit from NFT, all default access information as cleared or set in the NFT run from which you have exited is lost. If you run NFT again and type the INFORMATION DEFAULTS command before any SET DEFAULTS commands are given the response will always be either the default access information for the node where your logged-in directory is located (no NFT.INIT file) or defaults given by the SET DEFAULTS commands in the NFT.INIT file.

Example 1.

```
@NFT<RET>

    <ESC>                    <ESC>
     ↓                        ↓
NFT>inFORMATION (ABOUT) deFAULTS<RET>
Node 2102::/USER:CIRINO/ACCOUNT:341/OSTYPE:TOPS20

         <ESC>
          ↓
NFT>set deFAULTS (FOR NODE)<RET>

    <ESC>                    <ESC>
     ↓                        ↓
NFT>inFORMATION (ABOUT) deFAULTS<RET>
Node 2102::/OSTYPE:TOPS20
NFT>
```

Example 2.

```
@NFT(RET)
 Accessible DECNET nodes are:  2102, 1031, 4097, DN200, DN20A,
DN20L

     (ESC)                    (ESC)
NFT>inFORMATION (ABOUT) deFAULTS(RET)
Node 2102::/USER:PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
Node 1031::/USER:P:TAYLOR/ACCOUNT:341/OSTYPE:TOPS20
Node 4097::/USER:REILLY.PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
Node SYS880::/USER:GUEST/ACCOUNT:FOO/OSTYPE:RSX11

          (ESC)
NFT>set deFAULTS (FOR NODE)(RET)

     (ESC)                    (ESC)
NFT>inFORMATION (ABOUT) deFAULTS(RET)
Node 2102::/OSTYPE:TOPS20
Node 1031::/USER:PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
Node 4097:/USER:REILLY.PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
Node SYS880::/USER:GUEST/ACCOUNT:FOO/OSTYPE:RSX11

          (ESC)
NFT>set deFAULTS (FOR NODE) 1031::(RET)

     (ESC)                    (ESC)
NFT>inFORMATION (ABOUT) deFAULTS(RET)
Node 2102::/OSTYPE:TOPS20
Node 1031::/OSTYPE:TOPS20
Node 4097::/USER:REILLY.PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
Node SYS880::/USER:GUEST/ACCOUNT:FOO/OSTYPE:RSX11
NFT>
```

8.2.2 INFORMATION Command

The INFORMATION command has two options: DEFAULTS and DECNET.

The INFORMATION DEFAULTS command displays the current settings of the default switches for a specified node.

The format of the INFORMATION command is as follows:

```
NFT>INFORMATION (ABOUT) DEFAULTS
```

NFT displays the information about defaults in the following format:

```
node::/USER:userid/ACCOUNT:account-string/OSTYPE:ostype
```

where:

node::	is the name of the node for which the defaults listed have been set.

userid	is the valid user identification at the specified node.
account-string	is the valid account at the specified node.
ostype	is the operating system at the specified node.

Example:

```
@NFT(RET)
                  (ESC)
NFT>INFORMATION (ABOUT) DEFAULTS(RET)

Node 210A::/USER:SKOGLUND/ACCOUNT:341/OSTYPE:TOPS20
NFT>
```

The INFORMATION DECNET command displays the list of accessible DECnet nodes.

The format of the INFORMATION DECNET command is as follows:

```
NFT>INFORMATION (ABOUT) DECNET
```

Example:

```
@NFT(RET)
                  (ESC)
NFT>INFORMATION (ABOUT) DECNET (RET)
 Accessible DECNET nodes are:  1031, 2102, DN20A, DN20L
NFT>
```

8.2.3 COPY Command

The COPY command transfers files from the local node to a remote node or transfers files from a remote node to the local node.

The format of the COPY command is as follows:

```
NFT>COPY (FROM) node1::filespec1/switches (TO) node2::filespec2/switches
```

where:

node1::	is the node name of the source host from which the file is transferred. The default node is the local node.
filespec1	is the file specification of the file to be transferred. The specification must be in the format required by the operating system at node1::.
node2::	is the node name of the destination host to which the file is transferred. The default node is the local node.
filespec2	is the file specification to be given to the transferred file. The specification must be in the format required by the operating system at node::2.
/switches	are one or more of the switches defined in Table 8-1, as required.

Table 8-1
COPY Command Switches

Access Information Switches (Valid also with SET DEFAULTS, DIRECTORY, DELETE, SUBMIT, and TYPE commands)

/USER:userid

Sets the user identification associated with the node specified. Userid must be an ASCII string of 1 to 16 characters.

/ACCOUNT:account

Sets the account associated with the user identification at the node specified. Account must be an ASCII string of 1 to 16 characters.

/PASSWORD:password

Sets the password associated with the user identification at the node specified. Password must be an ASCII string of 1 to 8 characters.

Processing Mode Switches

/ASCII

Sets the file processing mode to ASCII.

/IMAGE

Sets the file processing mode to IMAGE. IMAGE indicates that the file is sent or received exactly as stored on disk.

/MACY11

Sets the file processing mode to MACY11. This switch is required to transfer PDP-11 object code. The MACY11 file format is produced by a TOPS-10/20 PDP-11 cross-assembler.

Record Length Switches (used only in combination with one of the processing mode switches)

/FIXED:nn

Defines a file as consisting of fixed length records of nn bytes.

/VARIABLE:nn

Defines a file as consisting of variable length records of maximum size of nn bytes.

Examples:

When a sequential file transfer is between two DECSYSTEM-20s, you may allow all fields except the filespec fields to be defaulted by omitting the switches and the node specification that represents the local node.

Each COPY command in the first three examples is valid for a TOPS-20 to TOPS-20 transfer. LOCAL is the name of the local node; REMOTE is the name of the remote node.

Example 1.

```
@NFT<RET>

NFT>COPY<ESC> (FROM) LOCAL::*.MAC<ESC> (TO) REMOTE::*.MAC<RET>

LOCAL::PS:<USER>ABC.MAC=>REMOTE::PS:<USER>ABC.MAC [OK]
LOCAL::PS<USER>XYZ.MAC=>REMOTE::PS:<USER>XYZ.MAC [OK]
```

The system responds, in this case, by naming all transferred files ending in .MAC. This use of the wildcard function is permitted only if both nodes support wildcarding.

Example 2.

```
@NFT<RET>

NFT>COPY<ESC> (FROM) ZOOM.*<ESC> (TO) REMOTE::ZOOM.*<RET>

LOCAL::PS:<USER>ZOOM.EXE=>REMOTE::PS:<USER>ZOOM.EXE [OK]
LOCAL::PS:<USER>ZOOM.MAC=>REMOTE::PS:<USER>ZOOM.EXE [OK]
```

In the above example, the system interprets the wildcard designation for file type, transfers the two extant files beginning with the file name ZOOM, and defaults the unspecified FROM node name to the local node name. Note also the complete file specification inserted by the system in both examples. You did not need to type the structure (PS:) or user <USER>.

Example 3.

```
@NFT<RET>

NFT>COPY<ESC> (FROM) REMOTE::ABC.TXT<ESC> (TO) ABC.TXT<RET>

REMOTE::PS<USER>ABC.TXT=>LOCAL::PS:<USER>ABC.TXT [OK]
```

In this example, the file is being transferred from the remote to the local node, whereas in the first two examples the files were being transferred from the local to the remote site.

Example 4.

```
@NFT(RET)
NFT>COPY TPARS.MAC/ASCII SY5101::DB0:TPARS.MAC/ASCII/VARIABLE:132(RET)
TPARS.MAC => SY5101::DB0:TPARS.MAC [OK]

NFT>DIRECTORY SY5101::DB0:*.MAC(RET)

  SY5101::DB0:[200,200]
TPARS.MAC;1;P775600              6 11264(8)    15-Aug-79 17:02:07
```

The above example differs from the first three examples in two ways. First, a file is being copied from the local system to a non-TOPS-20 system. (SY5101 is an RSX operating system.) Second, guidewords are not used. Note that this example includes the /ASCII switch for the TOPS-20 source node, and includes the /ASCII/VARIABLE:132 switches for the RSX destination node.

Specifying File Processing Mode

When transferring files between TOPS-20 systems, you do not need to specify the file processing mode because TOPS-20 has established defaults based on file bytesize. The defaults are:

File byte size	Default processing mode
7	/ASCII
all except 7	/IMAGE

Specifying Source File Processing Mode

When transferring local files to a non-TOPS-20 system, you must specify the local (source) file processing mode unless the file is bytesize 7.

When transferring files from the RSTS, IAS, or RT11 operating systems to your local system, you must specify the remote (source) file processing mode.

NOTE

Failure to match remote (source) file processing switches with the actual data mode of the remote file to be transferred will have unpredictable results.

Specifying Destination File Processing Mode

For each source file processing mode specified, there is a default destination file processing mode. This default value will be assumed if no destination file processing mode switch is specified. The defaults are:

Source mode		Default destination mode/record length
/ASCII	(TO)	/ASCII or /ASCII/VARIABLE
/ASCII/FIXED	(TO)	/ASCII
/ASCII/VARIABLE	(TO)	/ASCII
VMS print file format	(TO)	/ASCII
/IMAGE	(TO)	/IMAGE
/IMAGE/FIXED	(TO)	/IMAGE/FIXED
/IMAGE/VARIABLE	(TO)	/IMAGE/VARIABLE
/MACY11	(TO)	/IMAGE/VARIABLE
/MACY11/FIXED	(TO)	/IMAGE/FIXED
/MACY11/VARIABLE	(TO)	/IMAGE/VARIABLE

TOPS-20 NFT permits only the following source/destination file processing mode combinations when transferring a file TO a remote system:

Local mode		Remote mode
/ASCII	(TO)	/ASCII
/ASCII	(TO)	/ASCII/VARIABLE
/IMAGE	(TO)	/IMAGE
/IMAGE/FIXED	(TO)	/IMAGE/FIXED
/IMAGE/VARIABLE	(TO)	/IMAGE/VARIABLE
/MACY11	(TO)	/IMAGE/VARIABLE
/MACY11/VARIABLE	(TO)	/IMAGE/VARIABLE
/MACY11/FIXED	(TO)	/IMAGE/FIXED
/MACY11/IMAGE	(TO)	/IMAGE

TOPS-20 NFT permits only the following source/destination file processing mode combinations when transferring a file FROM a remote system:

Remote mode		Local mode
/ASCII	(TO)	/ASCII
/ASCII/FIXED	(TO)	/ASCII
/ASCII/VARIABLE	(TO)	/ASCII
/IMAGE	(TO)	/IMAGE
/IMAGE	(TO)	/MACY11/IMAGE
/IMAGE/FIXED	(TO)	/IMAGE/FIXED
/IMAGE/FIXED	(TO)	/MACY11/FIXED
/IMAGE/VARIABLE	(TO)	/IMAGE/VARIABLE
/IMAGE/VARIABLE	(TO)	/MACY11/VARIABLE
/IMAGE/VARIABLE	(TO)	/MACY11

8.2.4 DELETE Command

The DELETE command deletes files from a remote node.

The format of the DELETE command is as follows:

```
NFT>DELETE (REMOTE FILES) node::filespec/switches
```

Example:

```
@nft(RET)
 Accessible DECNET nodes are:  2102, 1031, 4097, DN200, DN20A
NFT>delete 2102::sep.txt(RET)
Access information for node 2102::/USER:cirino/ACCOUNT:341
Password:
2102::PS:<CIRINO>SEP.TXT.5 [OK]
NFT>
```

If no access information values have been established before the DELETE command, NFT will prompt for the required access information unless you supply switches with the DELETE command. These switches are effective only for the command in which you specify them.

8.2.5 DIRECTORY Command

The DIRECTORY command returns a directory listing of the files at the specified remote node. The system prints the directory heading and then lists the files in alphabetic order. For each file the following information is listed:

- Name, type, generation number
- Protection code
- Size in pages
- Length in bytes and byte size (in parentheses)
- The date and time the file was originally created or, if modified, the date last modified

The directory heading (node, structure, directory name) and the file name, type, and generation number are always in the format required by the remote site. All other information is listed in TOPS-20 format.

The format of the DIRECTORY command is as follows:

```
NFT>DIRECTORY (OF REMOTE FILES) node::filespec/switches
```

If no access information values have been established before the DIRECTORY command, NFT will prompt for the required access information unless you supply switches with the DIRECTORY command. These switches are effective only for the command in which you specify them.

Several examples follow. Both the environment and the timing of the DIRECTORY command influence the input/output associated with the command. Therefore, each example is preceded by the conditions that would call for the input as shown and result in the output as shown.

Example 1.

Conditions: The NFT DIRECTORY command is for one file on your own logged-in directory. There is no NFT.INIT file. The DIRECTORY command is the first command given in this NFT run. NFT knows USER and ACCOUNT because you logged in on this node. NFT always prompts for password unless it has been established with a SET DEFAULTS command. NFT does not print passwords.

```
@NFT(RET)

     (ESC)
      ↓
NFT>dirECTORY (OF REMOTE FILES) login.cmd(RET)
Access information for node 2102::/USER:KAMANITZ/ACCOUNT:341
Password:

  2102::PS:<KAMANITZ>
LOGIN.CMD.7;P777700       1 39(36)     14-Sep-79 14:34:01
NFT>
```

Example 2.

Conditions: Same as Example 1 except that an NFT TYPE command is given before the DIRECTORY command. There is no prompt for the password after the DIRECTORY command is given. The TYPE command was the first command and the password was entered in response to the prompt following the TYPE command. NFT remembers the password.

```
NFT>type switch.ini(RET)
Access information for node 2102::/USER:KAMANITZ/ACCOUNT:341
Password:
EDIT/SAVE:5/ISAVE:5

     (ESC)
      ↓
NFT>dirECTORY (OF REMOTE FILES) login.cmd(RET)

  2102::PS:<KAMANITZ>
LOGIN.CMD.7;P777700       1 39(36)     14-Sep-79 14:34:01
NFT>
```

Example 3.

Conditions: The DIRECTORY command is for the complete directory on another TOPS-20 node that is a member of your network. You have an account for node 4097. There is no NFT.INIT file. The DIRECTORY command is the first command given that requires access to node 4097. A prompt is given for each access information parameter.

```
@NFT<RET>
NFT>DIRECTORY <ESC> (OF REMOTE FILES) 4097::<RET>
Access information for node 4097::
User: CRUGNOLA<RET>
Account: 341<RET>
Password:

  4097::PS:<CRUGNOLA>
CALEND.EXE.1;P777700      5 2560(36)   10-Apr-78 11:23:48
DIDDLE.ZZZ.1;P777700      1 6(36)       6-Aug-79 15:51:41
LA36.CMD.1;P777700        1 74(7)      13-Mar-78 16:39:48
LOGIN.CMD.2;P777700       1 21(7)      13-Mar-78 16:36:23
MAIL.TXT.1;P770400        1 175(7)     26-Jul-79 13:01:46
PTYCON.ATO.1;P777700      1 1220(7)    19-May-78 13:34:26
SWITCH.INI.2;P777700      1 39(7)      18-May-78 15:33:10
VT50.CMD.1;P777700        1 28(7)      13-Mar-78 16:37:44
VT52.CMD.1;P777700        1 60(7)      13-Mar-78 16:38:49
ZIP.Q.1;P777700           1 45(7)      14-Jun-78 17:06:25
NFT>
```

Example 4.

Conditions: Same as Example 3 except that the DIRECTORY command uses the wildcard feature to request all files of type .CMD.

```
@NFT<RET>
NFT>DIRECTORY <ESC> (OF REMOTE FILES) 4097::*.CMD<RET>
Access information for node 4097::
User: CRUGNOLA<RET>
Account: 341<RET>
Password:

  4097::PS:<CRUGNOLA>
LA36.CMD.1;P777700        1 74(7)      13-MAR-78 16:39:48
LOGIN.CMD.2;P777700       1 21(7)      13-MAR-78 16:36:23
VT50.CMD.1;P777700        1 28(7)      13-MAR-78 16:37:44
VT52.CMD.1;P777700        1 60(7)      13-MAR-78 16:38:49
NFT>
```

Example 5.

Conditions: Two NFT Directory commands are directed to an RSX11 node that is a member of your network. You have an account on SY5101. Your logged-in directory has an NFT.INIT file with a SET DEFAULTS command that establishes values for USER, ACCOUNT, PASSWORD, and OSTYPE for 5101. The INFORMATION DECNET command is also included in the NFT.INIT file. The DIRECTORY command is the first command given that requires access to SY5101. The directory heading and file specifications are in RSX11 format. All other output is in TOPS-20 format. All values apply to the remote directory. The first DIRECTORY command is for all files on structure DK0. The second command is for all files of type .CMD on structure DB0. The wildcard feature is allowed because it is implemented by RSX11.

```
@NFT(RET)
 Accessible DECNET nodes are:  1031, 2102, 4114, DN20H, SY5101

                (ESC)
NFT>DIRECTORY (OF REMOTE FILES) SY5101::(RET)

  SY5101::DK0:[200,200]
INSTALL.CMD;17;P775600   1 1536(8)   27-Dec-78 17:44:26

                (ESC)
NFT>DIRECTORY (OF REMOTE FILES) SY5101::DB0:*.CMD(RET)

  SY5101::DB0:[200,200]
MERGE.CMD;1;P565600      1 512(8)    12-May-78 15:45:35
FLOPPY.CMD;1;P565600     1 512(8)    12-May-78 15:45:36
PLO.CMD;1;P565600        1 512(8)    12-May-78 15:45:36
PLOT.CMD;1;P565600       1 1024(8)   12-May-78 15:45:37
COPIES.CMD;3;P775600     1 512(8)     7-Sep-79 16:45:14
NFT>
```

8.2.6 EXIT Command

The EXIT command ends NFT execution. See the example in the HELP command which follows.

8.2.7 HELP Command

The HELP command displays the HELP file for NFT. The HELP file contains a description of all NFT commands and switches. The example that follows illustrates both the EXIT and HELP commands. CTRL/O was typed after the first two sentences of the HELP file. The file is approximately 4 pages and you can read it at your convenience.

Example:

```
$R NFT(RET)
NFT>HELP(RET)
NFT - Network file transfer program          23-Apr-79
```

NFT is the user interface to the network file transfer system. The services NFT provides are actually performed by a FAL (File Access Listener) process at the accessed node.

```
^O...
NFT>EXIT (RET)
$
```

8.2.8 SUBMIT Command

The SUBMIT command allows a Batch control file to be submitted to the Batch input queue at a specified remote node. The network file transfer system does not check to determine that the request is honored. This command is valid only for nodes that support remote command file submission.

The format of the SUBMIT command is as follows:

```
NFT>SUBMIT (REMOTE FILES) node::filespec/switches
```

where:

node::	is the node name of the remote host where the file is submitted.
filespec	is the file specification of the remote file.
/switches	are the access information switches.

8.2.9 TAKE Command

The TAKE command allows commands to be executed from a command file.

The format of the TAKE command is as follows:

```
NFT>TAKE (COMMANDS FROM) filespec1 (LOGGING OUTPUT ON) filespec2
```

where:

filespec1	is the file specification of the local command file.
filespec2	is the file specification of the local file for logging output.

When commands are executed that cause a prompt for access information, the command file execution is momentarily suspended, and you are given the prompt for the access information at your terminal. After you enter the required access information, the command file execution is resumed. This feature allows you to omit passwords from your command files.

8.2.10 TYPE Command

The TYPE command displays the file specified on your terminal. The file is transferred in ASCII.

The format of the TYPE command is:

```
NFT>TYPE (REMOTE FILES)node::filespec/switches
```

8.3 NFT ERROR MESSAGES

In the course of running NFT, you will probably receive error messages. NFT error messages preceded by % are warning messages. Warning messages may indicate errors or may give you information. You respond to them by taking the indicated action or adjusting your procedures on the basis of the information given. Error messages preceded by ? are fatal error messages. Except where otherwise stated, fatal errors can be handled by you alone, or by you with the help of a more experienced user.

In most messages, both the cause of the error and the action required are apparent from the text of the message. Where this is not the case, interpretive text is supplied following the message.

8.3.1 NFT Warning Message

%File attributes do not match processing mode

The file attributes at the remote site do not match those specified in the COPY command. This message will be received only if you are reading a file at a remote node running an operating system other than TOPS-20.

8.3.2 NFT Fatal Error Messages

?Cannot close logical link ... - TOPS-20 text for JSYS error

?Cannot do requested file format conversion

Check the allowable source/destination file processing mode combinations (Section 8.2.3).

?Cannot establish requested mode for input

?Cannot establish requested mode for output

?Cannot get JFN for logical link ... - TOPS-20 text for JSYS error

?Cannot open command file

Examine the file specification of your .CMD file for errors.

?Cannot open logging file

Examine the file specification of your .LOG file for errors.

?Cannot open logical link ... - TOPS-20 text for JSYS error

?Destination file must be specified

?Invalid account string

?Invalid destination processing mode

?Invalid password

?Invalid record length

?Invalid SET command

?Invalid switches for local file

?Invalid switches for remote file

?Invalid switch terminator

?Invalid use of wild cards

?Invalid user-id

?Logical link reception error

Check to see if your directory on the target system is over quota.

?Logical link transmission error

?Logical link was aborted ... - disconnect reason text

?Logical link was terminated

?Remote file attributes not supported

?Remote node is not responding

?Remote node refused connection ... - disconnect reason text

?Remote status ... - error text

If your network includes nodes running other than the TOPS-20 Operating System, you may receive this message. The error text that follows Remote status will be from the non-TOPS-20 operating system. Approximately 200 such messages are possible. Your local NFT repeats this "foreign" message verbatim. You will need help from your Software Support Specialist. Because the error message originates in the operating system at the remote site, the problem cannot be readily resolved.

?Remote system does not support default mode

?Remote system does not support file submission

The NFT SUBMIT command is not implemented at the remote node.

?Remote system does not support requested mode

?Remote to Remote transfers not supported

This message is displayed if you use the COPY command to transfer a file from one remote node to another remote node.

?Source file must be specified

?TOPS-20 text for JSYS error

This message consists of any appropriate JSYS error message not specifically covered in other messages in this list.

8.3.2.1 **Internal NFT Program Errors** - These fatal errors should not occur. If any one of these errors does occur, you will need the help of your Software Support Specialist. These errors are internal to the NFT program.

?Dap message buffer is full

?Invalid link index

?Logical link not established

?Invalid argument length

?Function not implemented

PART V

GENERATING DECnet-20

CHAPTER 9

USING THE CONFIGURATION TOOLS

If you are running DECnet-20 on a DECSYSTEM-2020, your only concern with this chapter is Section 9.4. This is the section that describes how you create the needed directories and restore the task-to-task files from the distribution tape. You should ignore all references to DN20 and DN200. The steps you execute in Section 9.4 are Steps 4, 5, and 6. Go now to Section 9.4.

Because of the several options now available in both hardware and software, each DECnet node on a DECSYSTEM-2040, 2050, or 2060 has a set of tools for customizing the DECnet configuration. There are no model dependencies.

Approximately three hundred files spread over two distribution tapes are used to tailor the software for various DECnet-20 configurations. The major file types and files are outlined in Section 9.3.

The tools for configuring the DECnet-20 front-end node (DN20 or DN21) for single or multi-line task-to-task are on the DECnet-20 Task-to-Task Distribution Tape. A successfully installed TOPS-20 Operating System running a Version 4 monitor is a prerequisite to on-site configuration for task-to-task capability. Configuring for task-to-task must precede configuring for RJE. The system may be in timesharing or standalone mode.

You will create two directories and one subdirectory on PS: to store temporarily the files needed for configuration. You must use the parameters given in the step-by-step procedures. You will need at least 3900 pages for working storage and 3900 pages for permanent storage. Ensure that space is adequate before you begin the procedures.

9.1 CONFIGURATION OVERVIEW

The logical steps in configuring the DECnet-20 communications front end or remote station are summarized below. A detailed step-by-step procedure is outlined in Section 9.4. The purpose of the general description below is to acquaint you with the configuration environment and thus provide a reference point for the information that precedes the step-by-step procedure.

- Log in and enable with OPERATOR or WHEEL privileges.

- Create one directory and one subdirectory for DECnet generation files:

  PS:<DNGEN>

  PS:<DNGEN.CMDS>

  The various files to be copied into these directories are on the distribution tape with the identical structure and directory names.

- Create one directory for documentation and source files:

  PS:<DECNET>

- Create one directory for example command and output files for acceptance testing.

- Mount the appropriate distribution tape (Task-to-Task or RJE) and use the DUMPER utility program to copy the directory contents from the tape to the two created directories on disk.

- Create a directory for your own customized files:

  PS:<DN20-1>

  or

  PS:<DN200>

- Prepare to build the customized directory. Preparation involves logical definitions, CONNECT and ACCESS commands, and a TAKE of the appropriate .CMD file. There are three .CMD files:

  DN20BLD.CMD

  Sets up the area for multiline task-to-task.

  DN200BLD.CMD

  Sets up the area for the RJE (DN200) option.

  32KBLD.CMD

  Sets up the area for a single line task-to-task DN20 (32K front end).

- Run NETGEN to define the configuration. This is the core of the configuration procedure and the step that demands preparation on your part.

- Submit the .CTL file now in your customized directory to the Batch system. Output will be files for the system (DN20 or DN200) generation and a .LOG file.

- Inspect the .LOG file output by the batch job to check for possible errors in the build process.

- Continue or rerun NETGEN as indicated by the inspection of the .LOG file. (Use the NETGEN command RESTORE.)

- Run VNP20. This program creates the system (DN20 or DN200) image.

- The system image created by the VNP20 program now exists as PS:<DN20-1> nodename.SYS (or PS:<DN200> nodename.SYS if you were configuring for RJE). If a change in the structure name or directory name or both is desired for your site, use the COPY command to accomplish the change at this time.

The file named nodename.SYS is the LOAD-FILE for the named network node. This file will be set up for loading into the front end as part of the installation procedure.

At the end of your first NETGEN run for the DN20, if no errors occur, your system will include basic task-to-task capability. A second run is necessary to generate RJE. In addition, you must repeat the configuration procedures whenever you change the network configuration (add a line, add a node, or add an option, for example).

Network File Transfer (NFT) is a utility program that is part of the basic DECnet task-to-task package. No customizing is required for NFT.

9.2 PREPARING TO USE THE CONFIGURATION TOOLS

Your configuration (and installation) procedures will run smoothly if you are well-prepared. It is strongly suggested that you observe the guidelines below before you begin to use the configuration tools.

- Read this entire chapter. Configuring the DECnet-20 communication front end requires an understanding of parameters not normally of concern to you. Also, terms used in NETGEN have a specific meaning to the program. All parameters referenced in the configuration process will be explained before a step-by-step procedure for configuring the front-end.

- Hardware parameters specifically entered or accepted as defaults MUST reflect the actual hardware in place at your site. Software parameters MUST be within the given range. Accepting defaults for software parameters is a safe procedure, but may not give optimum performance if you have a high load average and the maximum number of lines. Do NOT enter a value other than the default without understanding the tradeoffs described in the documentation. (As previously noted, all 32K systems must be configured exactly as described in the BEWARE file.)

- If you have any doubt about the value to type for a parameter, check with your system manager.

- You may find it helpful to make a written list of names, symbols, numbers and text strings you will need for use at the terminal. You will need a node name for both the DECnet communications front end and the host or central processor.

- Read the DECnet-20 BEWARE file if there is one.

- TOPS-20 Version 4 has a new operator command syntax and a new operator interface. You can successfully complete procedures by following the directions. However, you will be better prepared to handle unexpected situations if you are familiar with the new OPR-ORION facilities under GALAXY.

- You will use both DUMPER and the batch subsystem during configuration. Depending on your experience, you may wish to review these programs. Commands that check your progress through the procedures may be helpful.

- Set aside approximately two hours for your on-site configuration. (Actual time may be less or more, depending on you, the configuration, and your operating system.) Secure a terminal and location where you will not be interrupted.

9.3 THE CONFIGURATION TOOLS AND RELATED FILES

The term "configuration tools" refers to NETGEN, TKB20, VNP20, and DNMAC.

Two standard TOPS-20 DUMPER-formatted magnetic tapes contain the files needed. The distribution tapes are the Task-to-Task Release Tape and the DN200 RJE Release Tape. (There is also a tape with unsupported software for customer use.) The tapes contain the same information for all systems with the same unbundled options.

You must use the task-to-task tape to set up your system for task-to-task capability before you use the RJE tape (if this option is purchased).

The task-to-task tape contains all the files needed to support task-to-task capability for any configuration. Also included are the network control task (NETCON) and its associated loader files, and required .BWR, .HLP, and .DOC files. The major files and file types that are used to build your specific DECnet configuration are:

- PDP-11 files for all modules needed for any configuration of the DN2x. This includes object modules, symbol tables, and memory maps.

- DN20BLD.CMD. This is the command file for building a DN2x front end for single or multiple line task-to-task systems. (For single line task-to-task systems of 32K words, the comparable file is 32KBLD.CMD; for RJE the command file is DN200BLD.CMD.) When the commands in this file have been executed, your front-end directory will contain the following files:

    - BUILD-ALL.CTL. This is the control file for the batch job that follows the NETGEN program.

    - RSX11S.TSK. This file sets up the DN2x for the front-end memory image.

    - RSX11S.STB. This is the symbol table file for the front end.

- DNMAC.EXE. This is the cross assembler for PDP-11 macro source files.

- NETGEN.EXE. This is the program that allows you to enter parameters for your site. These parameters will be used to construct CETAB, the data base for the node you are configuring.
- TKB20.EXE. This program constructs PDP-11 formatted task images from object files.
- VNP20.EXE. This program creates the front-end system image.

9.3.1 The NETGEN Program

NETGEN is the interactive program that you use to describe the hardware and software for the node you are configuring.

9.3.1.1 **Features** - NETGEN has default values for all values that you type in by command or in response to a prompt. Where these values represent the correct values for your configuration, you need only type carriage return. The program will assume the default for that parameter.

If you type a ? following the NETGEN prompt (NETGEN>), the program responds with a list of valid commands.

```
NETGEN>?  Command name, one of the following:
 CCBs           Exclude        Exit           Finished
 Flow-count     Help           Include        Information
 Logical-links  Node           NRM-pipes      Number-of-nodes
 Password       RDB-size       RDBs           Restore
 Save           SDB-size       SDBs           Software-ID
 Transmit-buffers
NETGEN>
```

Now you know the possible commands, but not the possible responses. NETGEN will continue to help you via the ? feature. Type the command you wish to give and press the space bar. Again type ?. For example:

```
NETGEN>include ?  Device type, one of the following:
 Card-reader    DMC11          DTE20          DUP11
 KDP            Line-printer   Task
NETGEN>include
```

Assume you wish to inform NETGEN that your controller is a KDP. Type KDP. Never assume you have completed a command. Press the space bar and again type ?. The program responds with both a prompt for more information and the permissable range of values within which your response must fall. (Assuming the command has additional parameters to be entered):

```
NETGEN>include kdp ?  KDP Number, 0 to 2, decimal number
NETGEN>include kdp 0 (RET)
Number of DUP11 lines on this KDP(1-4):
```

In the last example you could also respond by typing KDP<ESC>. This would give you the guideword (number), but not the valid limits of the parameter. For example:

```
                  (ESC)
                    |
                    v
NETGEN>include kdp (number)
```

Using ? or <ESC> will not result in confusing messages. A typed <ESC> that has no purpose, that is, no further response is necessary, will have no effect. A ? typed when the command has been completed will result in the message shown below:

```
NETGEN>include dupll (number) 0 ?  confirm with carriage return
```

Recognition input (typing <ESC> to cause completion of commands and printing of guidewords) is available in NETGEN. This feature works the same for NETGEN commands as for TOPS-20 commands. The added information provided by the guidewords is even more helpful in NETGEN because the commands will not be familiar to you. In the following examples, the same three instructions require more than twice as many key strokes using full input rather than recognition input. As you read the recognition input, notice how the guidewords lead you to the correct response.

Full input:

```
NETGEN>PASSWORD RECEIVE MARL<RET>
NETGEN>SAVE PS:<DN20-1>NETGEN-CONFIG.IMAGE<RET>
```

Recognition input:

```
          <ESC>
NETGEN>Flow-count (for DN200 links is) 14<RET>

          <ESC>          <ESC>
NETGEN>Password (for) Receive (is) MARL<RET>

          <ESC>                                <ESC>
NETGEN>Save (configuration on) PS:<DN20-1>NETGEN-CONFIG.IMAGE<RET>
```

9.3.1.2 **NETGEN Terms and Concepts** - To configure and install your TOP-20 DECnet-20 Version 2.0 subsystem you need to be familiar with certain components you will normally be aware of indirectly if at all. This section is to familiarize you with the hardware and software terms used in NETGEN. The actual values to be entered will be explained later in the step-by-step procedure.

Hardware Terms:

All hardware terms you supply to NETGEN are DECnet-20 specific and must be directly related to the node you are configuring. You do not include hardware attached to the console front end or hardware attached to a communications front end running DN6x software.

DN20 — The term DN20 is a generic term and includes both the DN20 and the DN21, that is, DN20 refers to a DECnet-20 communications front end.

KDP — The term KDP is a DECnet term that refers to a combination of a KMC11 (controller) and one to four DUP11s. With the KMC the DUP11 functions as a direct memory access device (DMA). The interface is synchronous. The program prompts you for attached lines. DUP11s with a KMC are not specified separately with the INCLUDE command.

DUP11	This term is used in the NETGEN program to indicate DUP11s running as a character interrupt device, that is, without a KMC11. A DUP11 may be specified in an INCLUDE command only when it exists without a KMC.
DMC11	The DMC11 is a single line microprocessor-based interface to the network. The DMC11 is a synchronous DMA device.
DTE20	The DTE20 referred to in the program is the interface between the 2040, 2050, or 2060 processor and the DN20 front end. The parameter to be supplied to the program for the DTE20 is the number by which the software identifies the DTE, not the number of DTEs. Since DTE 0 is (and must be) the DTE that connects the console front end with the main processor, the parameter to be supplied will usually be DTE 1. The exception will be the case where your configuration has a front end for IBM Emulation/Termination. The two communication front end interfaces to the main processor would be DTE 1 and DTE 2. Provided the data base contains the valid number, it does not matter which is 1 and which is 2.
Links	This term appears in response to the INFORMATION command. The value is established by the program and not by a command you type. Links refers to the total number of physical links present in your DECnet-20 subsystem. If you have a KDP with two lines, for example, you have 3 links: one link controlled by DTE 1 to the main processor, one link with the identification KDP_0_0, and one link with the identification KDP_0_1.
Node	Used as a command, this term refers to your DECsystem-20 communications front end (or the DN200 if you are configuring for RJE). The NODE command takes several arguments. As used in the command NUMBER-OF-NODES, the reference is to the number of task-to-task links plus 2 (allowance for expansion of the network).
Line-printer Card-reader	Line printer and card-reader parameters are normally specified for the DN200 configuration only. (If you believe you have one or more of these devices on the DN20 when you do not have RJE, check with your manager.)

Software Terms:

The NETGEN program introduces software terms (several of which are acronyms) that may not be familiar to you. Or, you may be familiar with the terms but unaware of the significance of the parameters you enter associated with the terms.

The default values associated with the software terms will be acceptable in most instances. You can, however, alter the default values to tailor your DECnet system to your particular needs. Any changes you make in the values must fall within the limits set by the program. Do not change default values if you do not fully understand the implications of the change.

The software terms are explained in the order in which they would normally be entered or displayed. The format used is the command format unless the term is displayed only.

Node Name — This is the name for the node being generated. Within the network the node name must be unique. In the next release it will be a requirement that the node name include at least 1 alpha character. You may wish to meet this requirement during this generation.

Node Number — This term refers to the node identifying number. It may be any decimal number in the range 2 to 127 provided it does not duplicate a number already assigned to another node in the network. Node names and node numbers must be unique to establish the identify of a node in a unambiguous manner when the name or number is used in a command by the local or remote node.

Node Type — Node type must be either DN20 or DN200. DN20 is valid for task-to-task capability. DN200 is valid for RJE.

Software-ID — This term identifies the software you are including as your DECnet subsystem. Any text string from 1 to 32 characters is valid.

Password Transmit — This is the password sent to the remote node during an NSP initialization sequence. This is an optional parameter. If given, the transmit password must be a text string of 0-8 characters.

Password Receive — If specified, this password must be sent by the remote node during an NSP node initialization sequence. The receive password must be a text string of 0-8 characters. The receive password is optional. The default is null.

Logical-links — This term refers to the number of logical links NSP should support. Included are the logical links for user tasks and utilities. Be sure to include logical links for file transfer. The permitted range is 12-63; the default is 24. The cost of additional logical links is more memory for more speed.

CCBs — This term is the acronym for Communication Control Blocks. The CCBs command sets the number of CCBs. CCBs contain pointers to SDBs. The number of CCBs must be at least as large as the number of SDBs. The permitted range is 10-200; the default is 64.

RDBs

This term is the acronym for Receive Data Buffers. RDBs are used for both transmitting and receiving data segments. The RDBs command sets the number of RDBs. When the number of simultaneous exchanges between nodes exceeds the number of RDBs, a delay will occur before transmission can be made. This parameter depends on the configuration of your network and the extent and nature of your network activities. Factors to consider are: number of nodes; number of users; the mix of interactive jobs to file transfer; peak and average load; and the limits set by available memory. More RDBs improve throughput at the expense of memory. The permitted range is 6-200; the default is 64.

RDB-size

This term refers to the size of the RDBs in bytes. The size of the RDB determines the largest network data segment your system can transmit or receive via the physical line protocol (DDCMP). Large buffer size is best for file transfer service; small buffer size (and hence more buffers) is best for interactive traffic. The permitted range is 290-2082; the default is 290.

SDBs

This term is the acronym for Small Data Buffers. The SDBs command sets the number of SDBs. SDBs are used for internal storage of transmit data and temporary data structures. The number of SDBs that can be used at one time is limited to the number of CCBs you specify. The permitted range is 8-63; the default is 32.

SDB-size

This term refers to the size in bytes of the SDBs. The permitted range is 80-128. The default is 80. SDB size, within the given limits, will have no appreciable effect on performance.

Transmit-buffers

The number of Transmit buffers specified determines the number of transmit requests that NSP will try to keep outstanding. The permitted range is 2-16; the default is 2. The tradeoff is more memory for more speed. Double buffering is generally acceptable.

Number-of-nodes

This term refers to the number of nodes in the same network as the node you are configuring plus 2 to allow for expansion. The permitted range is 8-20; the default is 12. The number-of-nodes must be at least equal to the maximum number of lines plus 2.

NRM Pipes
NRM Flow Count

The defaults currently set are adequate. Allow these parameters to be defaulted.

Estimated-Pool-Size

This parameter is calculated by the program and appears only in response to the INFORMATION command. Pool-size is calculated on the basis of your specified configuration.

9.4 CONFIGURING FOR TASK-TO-TASK CAPABILITY

Files on the Task-to-Task Release Tape include all files that are needed to provide task-to-task capability for any DECSYSTEM-2020, 2040, 2050, or 2060. The DN20 files on tape are located in the directory PS:<DNGEN> and the subdirectory PS:<DNGEN.CMDS>. You create identical directories on your system and then copy the entire contents of the two directories to your system. These two directories should be saved as copied. In addition, if you are running a DECSYSTEM-2040, 2050, or 2060, you create a separate directory that will contain only those files specific to the DN20 system that you are generating. This directory is PS:<DN20-1>.

Once the directories are created and the files are restored from tape, you proceed with the several steps that build the DN20 front end. The parameters that appear in example commands have been chosen to ensure an efficient procedure: space parameters allow for expansion and actual and logical names permit inter-file referencing. You should not experience needless delays if you follow the directions exactly.

If you are upgrading from a previous version, use <NEW-SYSTEM> and <NEW-SUBSYS> for the directories <SYSTEM> and <SUBSYS> respectively. Files from the current version of the distribution tape should be copied to empty directories, that is, not mixed with files of a prior version. (The previous version should be saved on tape or disk until the current version runs successfully in a normal working environment.)

Because command and control files and logical definitions reference other files, you should name and locate files as given in the text and examples. If indicated by your site's conventions, file names and locations may be changed after the configuration and installation procedures have been successfully completed. Be aware, however, that should you later wish to reconfigure the system or run validation or certification tests for diagnostic purposes, references to files and structures will need to be resolved.

A fully installed TOPS-20 operating system with a monitor that is Version 4.0 or later is a prerequisite to on-site configuration of the DN20. Log in as a user with OPERATOR or WHEEL privileges and continue from Step 1 for DECSYSTEMs-2040/50/60, or from Step 4 for DECSYSTEM-2020.

Step 1: Create PS:<DNGEN>. (DECSYSTEMs-2040/50/60 only)

Enter:

```
@ENABLE (RET)
$^ECREATE PS:<DNGEN> (RET)
[New]
$$WORKING 1000 (RET)
$$PERMANENT 1000 (RET)
$$PROTECTION 774000 (RET)
$$PASSWORD DECNET (RET)
$$DIRECT 4747 (RET)
$$USER 4747 (RET)
$$MAXIMUM 1 (RET)
$$ (RET)
$DISABLE (RET)
@
```

Step 2: Create PS:<DNGEN.CMDS>. (DECSYSTEMs-2040/50/60 only)

Enter:

```
@ENABLE RET
$^ECREATE PS:<DNGEN.CMDS> RET
[New]
$$WORKING 150 RET
$$PERMANENT 150 RET
$$PROTECTION 774000 RET
$$DIRECT 4747 RET
$$FILES-ONLY RET
$$ RET
$DISABLE RET
@
```

Step 3: Create PS:<DN20-1>. (DECSYSTEMs-2040/50/60 only)

Enter:

```
@ENABLE RET
$^ECREATE PS:<DN20-1> RET
[New]
$$WORK 750 RET
$$PERM 750 RET
$$USER 4747 RET
$$PASSWORD XXX RET
$$ RET
$DISABLE RET
@
```

The password to the front-end directory (PS:<DN20-1>) to be built may be your own password for logging in to PS:.

Step 4: Create PS:<DECNET>

Enter:

```
@ENABLE RET
$^ECREATE PS:<DECNET> RET
[New]
$$WORK 2000 RET
$$PERM 2000 RET
$$PROTECTION 774000 RET
$$USER 4747 RET
$$PASSWORD XXX RET
$$ RET
$DISABLE RET
@
```

Step 5: Create PS:<UETP.DECNET>.

Enter:

```
@ENABLE (RET)
$^ECREATE PS:<UETP.DECNET> (RET)
[New]
$$WORK 100 (RET)
$$PERM 100 (RET)
$$FILES-ONLY (RET)
$$USER 100 (RET)
$$DIRECT 4747 (RET)
$$ (RET)
$DISABLE (RET)
@
```

Step 6: Use DUMPER to restore files from tape to disk. Mount the DECnet Version 2 distribution tape on an available drive. (MTA0 or MT0 is used in the examples.) If you are running a DECSYSTEM-2020, complete Step 6 and continue with Chapter 10.

NOTE

Restoring the save sets not related to configuring the front end is done at this time to avoid a second handling of the tape.

Use the INFORMATION SYSTEM-STATUS command to determine if tape allocation is enabled for your system.

If your installation does not include the tape allocation enabled feature:

Enter:

```
@ENABLE (RET)
$ASSIGN MTA0: (RET)
$DUMPER (RET)
DUMPER>TAPE MTA0: (RET)
DUMPER>REWIND (RET)
```

```
                ESC
                 ↓
DUMPER>ACCOUNT (OF RESTORED FILES FROM) SYSTEM-DEFAULT (RET)
```

```
                ESC             ESC
                 ↓               ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <DECNET> (RET)
(Restores Documentation files.)
```

```
                ESC             ESC
                 ↓               ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <SUBSYS> (RET)
(Restores Binary files.)
```

```
                ESC             ESC
                 ↓               ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <DECNET> (RET)
(Restores Source files.)
```

NOTE

The next RESTORE command is for DECSYSTEMs-2040/50/60 only.

```
                ESC
                 ↓
DUMPER>RESTORE (TAPE FILES) PS:<DNGEN>,PS:<DNGEN.CMDS>RET
(Restores DN20 Generation files)

                ESC                ESC
                 ↓                  ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <UETP.LIB>RET
(Restores the acceptance files.)

                ESC                ESC
                 ↓                  ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <UETP.DECNET>RET
(Restores the sample files for acceptance tests.)
DUMPER>REWINDRET
DUMPER>EXITRET
$DISABLERET
@
```

Or if your installation includes the tape allocation enabled feature:

Enter:

```
@ENABLERET
$MOUNT TAPE logical-name:RET
[logical name: assigned to MT0]
```

NOTE

If you are working standalone and there is no operator in attendance, go to the operator's console and enter the IDENTIFY command:

```
OPR>IDENTIFY tape-id REQ request-id
```

The request-id appears on the operator's console. The tape-id you type is MTAn. The system associates the physical device MTAn with your logical assignment to MTn.

```
$DUMPER (RET)
DUMPER>TAPE logical-name: (RET)
DUMPER>REWIND (RET)

            (ESC)
            ↓
DUMPER>ACCOUNT (OF RESTORED FILES FROM) SYSTEM-DEFAULT (RET)

            (ESC)           (ESC)
            ↓               ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <DECNET> (RET)
(Restores Documentation files.)

            (ESC)           (ESC)
            ↓               ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <SUBSYS> (RET)
(Restores Binary files.)

            (ESC)           (ESC)
            ↓               ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <DECNET> (RET)
(Restores Source files.)
```

NOTE

The next RESTORE command is for DECSYSTEMs-2040/50/60 only.

```
            (ESC)
            ↓
DUMPER>RESTORE (TAPE FILES) PS:<DNGEN>,PS:<DNGEN.CMDS> (RET)
(Restores DN20 Generation files)

            (ESC)           (ESC)
            ↓               ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <UETP.LIB> (RET)
(Restores the acceptance files.)

            (ESC)           (ESC)
            ↓               ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <UETP.DECNET> (RET)
(Restores the sample files for acceptance tests.)
DUMPER>REWIND (RET)
DUMPER>EXIT (RET)
$DISMOUNT TAPE logical name (RET)
$DISABLE (RET)
@
```

The logical name used above may be your first or last name, or any other valid string you will remember. If you are installing a DECSYSTEM-2020, go to Chapter 10 now.

Step 7: Set up the build files.

Enter:

```
@DEFINE GEN:  PS:<DNGEN>,PS:<DNGEN.CMDS>(RET)
```

This command defines the location of the release distribution areas.

```
@DEFINE DN20:  PS:<DN20-1>(RET)
```

This command defines the front-end directory created for site-specific files.

```
@DEFINE DSK:  DSK:,GEN:(RET)
```

This command defines the file search association of DSK:

Step 8: CONNECT to DN20: and establish group privileges by performing an ACCESS.

Enter:

```
@CONNECT DN20:(RET)
Password: XXX
@ACCESS DN20:(RET)
Password: XXX
```

The password is the same as that used in Step 3.

Step 9: Tell the system to process the command file appropriate for your configuration.

With this step you begin work on configuration. From this point on files will be put to the directory <DN20-1>. Therefore, a failure in generation procedures may require that all files be deleted from <DN20-1> and that you repeat all from Step 9.

If you are configuring a multiline task-to-task system with or without the RJE option or a single line task-to-task system with more than 32K of memory:

Enter:

```
@TAKE DN20BLD.CMD(RET)
```

If you are configuring a single line task-to-task system with only 32K of memory:

Enter:

```
@TAKE 32KBLD.CMD(RET)
```

The following files are now in the DN20: directory, PS:<DN20-1>:

BUILD-ALL.CTL.1	(control file for the Batch job that follows NETGEN)
RSX11S.STB	(symbol definition file)
RSX11S.TSK	(task image file)

If you use the wrong file in this step, simply repeat the TAKE command using the correct file.

Step 10: Run the NETGEN program to define the system configuration.

Enter:

```
@netgen(RET)
TOPS-20 Network Configuration for DN2x or DN200, Version:  V2.2-39
```

NOTE

If you are configuring a 32K DECnet-20 communications front end and have neglected to read the file DECNET.BWR.1, do so now before you continue. Within the beware file there is a memo with the subject title "Configuration of Release 4 32Kw DN20s". The beware file may be printed or typed from PS:<DECNET>DECNET.BWR.1 (unless you elected to put the documentation save set in a different directory).

Step 10.1: Give the identifying parameters for your DN20. The following is an example entry.

Enter:

```
NETGEN>node name mar1(RET)
NETGEN>node number 67(RET)
NETGEN>node type DN20(RET)
```

NOTE

You could have taken advantage of NETGEN features by using any of the following for help:

```
NETGEN> ?  Command name, one of the following:
 CCBs           Exclude         Exit            Finished
 Flow-count     Help            Include         Information
 Logical-links  Node            NRM-pipes       Number-of-nodes
 Password       RDB-size        RDBs            Restore
 Save           SDB-size        SDBs            Software-ID
 Transmit-buffers
NETGEN>node ? Parameter, one of the following:
 name        number    type
NETGEN>node name marl(RET)
NETGEN>node number 67(RET)
NETGEN>node type ? Node type, one of the following:
 DN20       DN200
NETGEN>node type DN20(RET)
NETGEN>
```

> You may need to use the <ESC> and ? in later commands. They will not always be given in examples. If you need more help with a parameter or command, refer back to Section 9.3.1.1.

You are now ready to examine the default values and accept them or change them to reflect your planned configuration.

Step 10.2: Check the current parameters.

```
NETGEN>info(RET)
   Node name is:      MARL
   Node number is:    67
   Software ID is: DECnet-20 V2
   Transmit Password is: DECNET20 (default)
   Receive Password is: (null) (default)
   Logical links:     24 (default)
   CCBs:              64 (default)
   RDBs:              64 (default)
   RDB-size:          290 (default)
   SDBs:              32 (default)
   SDB-size           80 (default)
   Transmit buffers: 2 (default)
   Node type is:      DN20
   Number of nodes:   12 (default)
   KDPs:              1 (default)
        KDP_0 has 1 lines,
   DUPlls:            0 (default)
   DMClls:            0 (default)
   Physical links:    2 (default)
    NSP link controlled by DTE_1, Task-to-Task
    NSP link controlled by KDP_0_0, Task-to-Task
   DTE20 number:      1 (default)
   Line printers:     0 (default)
   CRlls:             0 (default)
   Tasks:  (defaulted)
                72-Hour-Test
                Topology-Notifier
   NRM pipes:         3 (default)
   NRM Flow-count:    16 (default)
   Estimated Pool-size: 5504 bytes
NETGEN>
```

Step 10.3: Adjust the current hardware parameters as required by the hardware at your site.

Example #1. To include 3 lines controlled by the KDP, rather than the 1 indicated.

Enter:

```
NETGEN>exclude kdp 0 (RET)
NETGEN>include kdp 0 (RET)
Number of DUP11 lines on this KDP(1-4): 3
NETGEN>
```

The KDP is first excluded and then included to permit you to specify the changed number of lines and trigger the correct number of prompts for link usage.

Example #2. To include 3 DMC11s and no KDPs.

Enter:

```
NETGEN>exclude kdp 0 (RET)
NETGEN>include dmc11 0 (RET)
NETGEN>include dmc11 1 (RET)
NETGEN>include dmc11 2 (RET)
NETGEN>
```

Step 10.4: Check all current parameters. (The output shown assumes input as in Example 2.)

Enter:

```
NETGEN>information (RET)
   Node name is:      MARL
   Node number is:    67
   Software ID is: DECnet -20 V2
   Transmit Password is: DECNET20 (default)
   Receive Password is: (null) (default)
   Logical links:     24 (default)
   CCBs:              64 (default)
   RDBs:              64 (default)
   RDB-size:          290 (default)
   SDBs:              32 (default)
   SDB-size           80 (default)
   Transmit buffers:  2 (default)
   Node type is:      DN20
   Number of nodes:   12 (default)
   KDPs:              0
   DUP11s:            0 (default)
   DMC11s:            3
   Physical links:    4
    NSP link controlled by DTE_1, Task-to-Task
    NSP link controlled by DMC11_0, Task-to-Task
    NSP link controlled by DMC11_1, Task-to-Task
    NSP link controlled by DMC11_2, Task-to-Task
   DTE20 number:      1 (default)
   Line printers:     0 (default)
   CR11s:             0 (default)
   Tasks:  (defaulted)
                72-Hour-Test
                Topology-Notifier
   NRM pipes:         3 (default)
   NRM Flow-count:    16 (default)
   Estimated Pool-size: 5504 bytes
NETGEN>
```

Step 10.5: Make necessary additions and any desired changes in parameters. The default parameters for software should be adequate for the two examples. Space permitting, you may wish to increase the number of CCBs, SDBs, RDBs, and logical links for the configuration with the three higher speed DMC11s.

If you are configuring a system with 32K, you must exclude either the 72-Hour-Test or the Topology-Notifier.

Possible entries, depending on the network you plan, might be:

```
NETGEN>password (for) ? Password type, one of the following:
 Receive    Transmit

NETGEN>password (for) receive (is) ? Receive password, 0 to 8 characters, text string

NETGEN>password (for) receive (is) GUEST<RET>

NETGEN>password (for) transmit (is) MARL67<RET>

                                  *********

NETGEN>exclude (from Configuration) task (type) ? task type, one of the following:
 72-Hour-Test          Operator-Console        Topology-Notification

NETGEN>exclude (from Configuration) task (type) 72-Hour-Test<RET>
NETGEN>

                          *********
NETGEN>ccbs ? Number of CCBs, 10 to 200, decimal number
NETGEN>ccbs 96<RET>
NETGEN>sdbs ? Number of SDBs, 8 to 63, decimal number
NETGEN>sdbs 40<RET>
NETGEN>rdbs ? Number of RDBs, 6 to 200, decimal number
NETGEN>rdbs 96<RET>
NETGEN>
```

Step 10.6: Examine the current parameters once again if you made any entries since Step 10.4.

The example below assumes you have added MARL67 and GUEST as the transmit and receive passwords, respectively, rather than accept the defaults.

Entry and response:

```
NETGEN>information (RET)
   Node name is:      MARL
   Node number is:    67
   Software ID is:    DECnet -20 V2
   Transmit Password is: marl67
   Receive Password is: guest
   Logical links:     24 (default)
   CCBs:              64 (default)
   RDBs:              64 (default)
   RDB-size:          290 (default)
   SDBs:              32 (default)
   SDB-size:          80 (default)
   Transmit buffers:  2 (default)
   Node type is:      DN20
   Number of nodes:   12 (default)
   KDPs:              0
   DUP11s:            0 (default)
   DMC11s:            3
   Physical links:    4
    NSP link controlled by DTE_1, Task-to-Task
    NSP link controlled by DMC11_0, Task-to-Task
    NSP link controlled by DMC11_1, Task-to-Task
    NSP link controlled by DMC11_2, Task-to-Task
   DTE20 number:      1 (default)
   Line printers:     0 (default)
   CR11s:             0 (default)
   Tasks:  (defaulted)
                 72-Hour-Test
                 Topology-Notifier
   NRM pipes:         3 (default)
   NRM Flow-count:    16 (default)
   Estimated Pool-size: 5504 bytes
NETGEN>
```

Step 10.7: When the output from the INFORMATION command accurately reflects what you want your configuration to be, save the completed NETGEN procedure. Check very carefully: Once you have typed "FINISHED", correction procedures will include returning to Step 9 or less.

Enter:

```
NETGEN>save ? output filespec
```


```
NETGEN>save ps:<DN20-1>NETGEN-CONFIG.IMAGE (RET)
```

(Typing escape results in the output NETGEN-CONFIG.IMAGE as the default file name.)

Step 10.8: Type the commands FINISHED and EXIT.

Enter:

```
NETGEN>finished (RET)
%Have you issued a SAVE command for this configuration?
NETGEN>exit (RET)
@
```

(The reminder above, given after you type finished, is given ONLY if you did NOT save the completed NETGEN procedure. The reminder would not appear if Step 10.7 had been executed as given.)

The following files are now in your connected directory:

BUILD-ALL.CTL.1

CETAB.MAC.1

GBLDLX.MAC.1

GBLFLO.MAC.1

GBLNSP.MAC.1

NETGEN-CONFIG.IMAGE.1

NRMDAT.MAC.1

RSX11S.STB.1

RSX11S.TSK.1

Step 11: Ensure that the DN20 directory contains only the necessary files.

If the system notifies you that deleted files will be expunged, do NOT begin the following sequence. Wait until the expunge is completed. (The undelete will not function if the files have been expunged.)

Enter:

```
@delete *.* (RET)
@undel rsx11s.* (RET)
@undel *.image (RET)
@undel *.mac (RET)
@undel build-all.ctl (RET)
```

Step 12: Submit the BUILD-ALL.CTL file to the batch system.

Enter:

```
@SUBMIT BUILD-ALL.CTL/TIME:15/OUTPUT:NOLOG/NOTIFY:YES (RET)
[Job BUILD Queued, Request-ID #60 Limit 00:15:00]
```

The batch job should take from 10 to 15 minutes. When the job is completed, the system will output the equivalent of:

[From SYSTEM: Job BUILD Request # 60 Finished Executing at 14:47:31]

Step 13: Type or print the BUILD-ALL.LOG file and examine the file for errors.

Interpret Batch system messages as follows:

? at the start of the message indicates a fatal error.

% at the start of a message indicates a warning.

[at the beginning of a message indicates a comment.

Following successful completion of the Batch job, all files that are needed to construct the actual system have been created. Only jobs with fatal errors have failed to complete successfully. Warning messages may appear for any given configuration. Undefined global symbols, default symbols generated, and possibly multiply-defined global symbols may require no action on your part. All fatal errors require repetition of some previous steps. Interpret the fatal error message given to determine the steps to be repeated. (See Appendix G for a complete list of error messages for TKB20 and VNP20.) Consult your Software Support Specialist if you are in doubt as to how to proceed.

Step 14: Run the VNP20 program.

Enter:

@VNP20 (RET)

This program runs in approximately 2 minutes of CPU time.

When VNP20 completes successfully, the following output appears on your terminal:

- information on free space within the core image, the kernel pool, and the network pool
- a partition map of the created system front-end image

Your system now includes basic task-to-task capability. To include the RJE option, continue with Section 9.5. If you are generating a system without the RJE option, go now to Chapter 10.

IN CASE OF ERRORS:

If errors prevent successful completion of a step in the configuration process, check all steps prior to the point of failure. Specifically check if user groups were set correctly in Steps 1, 2, and 3, and if the access command was correctly executed in Step 6. If you have not found your error, check Step 9. Did you use the correct file for the TAKE command? If so, next check the files that should be in the DN20 directory at the end of Step 9 and at the end of Step 10.8. If an error has not been found, check the NETGEN procedure for omissions or errors and the BUILD-ALL.LOG file for error messages preceded by "?". If a fatal error is found in the log file, delete the log file. Go back to the point of error and repeat all from that point.

9.5 CONFIGURING FOR REMOTE JOB ENTRY

To build the DN200 image you follow the same logical steps you followed to build the DN20. However, because of the differences in hardware and functions, there will be some differences in files used and values you specify. These differences will be given under the step number when they occur.

In all instances the name DN20 becomes DN200.

Steps 1 and 2. (Creation of directories.)

You have already created the directories PS:<DNGEN> and PS:<DNGEN.CMDS>. The additional files needed for the DN200 configuration will be added to these directories.

Step 3: Create PS:<DN200>.

Enter:

```
@ENABLE (RET)
$^ECREATE PS:<DN200> (RET)
[New]
$$WORK 750 (RET)
$$PERM 750 (RET)
$$USER 4747 (RET)
$$PASSWORD XXX (RET)
$$ (RET)
$DISABLE (RET)
@
```

The password to the DN200 directory to be built may be your own password for logging in to PS:.

Step 4: Use DUMPER to restore files from tape to disk.

The input tape is the DN200 RJE Release Tape.

If your installation does not include the tape allocation feature:

Enter:

```
@ENABLE (RET)
$ASSIGN MTA0 (RET)
$DUMPER (RET)
DUMPER>TAPE MTA0 (RET)
DUMPER>REWIND (RET)

              (ESC)
              ↓
DUMPER>ACCOUNT (OF RESTORED FILES FROM) SYSTEM-DEFAULT (RET)

              (ESC)              (ESC)
              ↓                  ↓
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.* (TO) <DECNET> (RET)
(Restores Documentation files.)

              (ESC)
              ↓
DUMPER>RESTORE (TAPE FILES) PS:<DNGEN>,PS:<DNGEN.CMDS> (RET)
(Restores DN200 Generation files)
DUMPER>REWIND (RET)
DUMPER>EXIT (RET)
$DISABLE (RET)
@
```

Or, if your installation includes the tape allocation feature:

Enter:

```
@ENABLE (RET)
$MOUNT TAPE logical name: (RET)
[logical name: defined as MT0:]
```

NOTE

If you are working standalone and there is no operator in attendance, go to the operator's console and enter the IDENTIFY command:

OPR>IDENTIFY tape-id REQ request-id

The request-id appears on the operator's console. The tape-id you type is MTAn. The system associates the physical device MTAn with your logical assignment to MTn.

```
$DUMPER (RET)
DUMPER>TAPE logical name:(RET)
DUMPER>REWIND (RET)
```

(ESC)

```
DUMPER>ACCOUNT (OF RESTORED FILES FROM) SYSTEM-DEFAULT (RET)
```

(ESC) (ESC)

```
DUMPER>RESTORE (TAPE FILES) PS:<*>*.*.*(TO) <DECNET> (RET)
(Restores Documentation files.)
```

(ESC)

```
DUMPER>RESTORE (TAPE FILES) PS:<DNGEN>,PS:<DNGEN.CMDS> (RET)
(Restores DN200 Generation files)
DUMPER>REWIND (RET)
DUMPER>EXIT (RET)
$DISMOUNT TAPE logical name (RET)
$DISABLE (RET)
@
```

Step 5: Set up the build files.

Enter:

```
@DEFINE GEN: PS:<DNGEN>,PS:<DNGEN.CMDS> (RET)
@DEFINE DN200: PS:<DN200> (RET)
@DEFINE DSK: DSK:,GEN: (RET)
```

Step 6: CONNECT to DN200: and establish group privileges by performing an ACCESS.

Entry and response:

```
@CONNECT DN200:(RET)
Password: xxx
@ACCESS DN200:(RET)
Password: xxx
```

Step 7: Tell the system to process the appropriate command file.

Your work on configuration begins with this step. As was noted in configuring the DN20, a failure in procedures from this point on may require that existing files be deleted from <DN200> and that you repeat all from Step 7.

Enter:

```
@TAKE DN200BLD.CMD(RET)
```

Step 8: Run the NETGEN program to define the DN200 configuration.

Enter:

```
@NETGEN(RET)
TOPS-20 Network Configurator for DN2x or DN200, Version:  V2.2-39
```

Step 8.1: Give the identifying parameters for the system.

Enter:

```
NETGEN>node name mar200(RET)
NETGEN>node number 68(RET)
NETGEN>node type dn200(RET)
```

Step 8.2: Check the current parameters.

Enter:

```
NETGEN>info (RET)
   Node name is:      MAR200
   Node number is:    68
   Software ID is:    DECnet -20 V2
   Transmit Password is: DECNET20 (default)
   Receive Password is:  (null) (default)
   Logical links:     17 (default)
   CCBs:              30 (default)
   RDBs:              20 (default)
   RDB-size:          290 (default)
   SDBs:              15 (default)
   SDB-size:          80 (default)
   Transmit buffers:  2 (default)
   Node type is:      DN200
   Number of nodes:   12 (default)
   KDPs:              0 (default)
   DUPlls:            0 (default)
   DMClls:            1 (default)
   Physical links:    1 (default)
    NSP link controlled by DMC11_0, Task-to-Task
   No DTE20           (default)
   Line printers:     1 (default)
    Line Printer #0 is an  LP11
   CRlls:             0 (default)
   Tasks:  (defaulted)
                Operator-Console
   NRM pipes:         6 (default)
   NRM Flow-count:    4 (default)
   Estimated Pool-size: 3320 bytes
NETGEN>
```

Note that the program has adjusted the default parameters to values that probably will be most common to DN200 configurations.

Step 8.3: Adjust the hardware parameters as required by hardware at your site.

Possible additions/changes (ensure that your site requires or desires them) might be:

To include a card reader

Enter:

```
NETGEN>include card-reader (RET)
```

To exclude the printer

Enter:

```
NETGEN>exclude line-printer (RET)
```

To add a receive password, change the transmit password, or both

Enter:

```
                                ESC
                                 ↓
NETGEN> password transmit (is) MR200 RET
```

```
                                ESC
                                 ↓
NETGEN> password receive (is) TOMR RET
```

(Use of passwords in communication between the front end and the remote station is not a DECnet-20 requirement but is an option.)

Step 8.4: Check all current parameters if you made any changes or additions. (The output shown assumes a card reader was added and transmit and receive passwords specified.)

Enter:

```
NETGEN>info RET
   Node name is:      MAR200
   Node number is:    68
   Software ID is:    DECnet -20 V2
   Transmit Password is:  MR200
   Receive Password is:  TOMR
   Logical links:     17 (default)
   CCBs:              30 (default)
   RDBs:              20 (default)
   RDB-size:          290 (default)
   SDBs:              15 (default)
   SDB-size:          80 (default)
   Transmit buffer:   2 (default)
   Node type is:      DN200
   Number of nodes:   12 (default)
   KDPs:              0 (default)
   DUPlls:            0 (default)
   DMClls:            1 (default)
   Physical links:    1 (default)
    NSP link controlled by DMC11_0, Task-to-Task
   No DTE20           (default)
   Line printers:     1
   CRlls:             1
   Tasks:  (defaulted)
                 Operator-Console
   NRM pipes:         6 (default)
   NRM Flow-count:    4 (default)
   Estimated Pool-size: 3320 bytes
NETGEN>
```

Step 8.5: When the output from the INFORMATION command accurately reflects your configuration, save the completed NETGEN procedure.

Enter:

```
                                ESC
                                 ↓
NETGEN>save PS:<DN200>NETGEN-CONFIG.IMAGE RET
```

Step 8.6: Type the commands FINISHED and EXIT.

Enter:

```
NETGEN>finished (RET)
NETGEN>exit (RET)
@
```

Step 9: Ensure that the DN200 directory contains only the necessary files.

If the system notifies you that deleted files will be expunged, do NOT begin the following sequence. Wait until the expunge is completed. (The undelete will not function if the files have been expunged.)

Enter:

```
@delete *.* (RET)
@undel rsx11s.* (RET)
@undel *.image (RET)
@undel *.mac (RET)
@undel build-all.ctl (RET)
```

Step 10: Submit the BUILD-ALL.CTL file to the Batch system.

Entry and response:

```
@SUBMIT BUILD-ALL.CTL/TIME:15/OUTPUT:NOLOG/NOTIFY:YES (RET)
[Job BUILD Queued, Request-ID #61 Limit 15:00:00]
```

Step 11: Type or print the BUILD-ALL.LOG file and examine the file for errors.

Step 12: Run the VNP20 program.

NOTE

Be sure that you have configured ALL nodes. If, for example, you have 3 DN200s in your configuration, you run NETGEN once for the DN20 front end and once for each of the DN200s. (Each node requires its own communications data base.)

CHAPTER 10

INSTALLING DECnet-20

If you are installing a DECSYSTEM-2020, go to Section 10.1. If you are installing a DECSYSTEM-2040/50/60, continue with the next paragraph.

Following the configuration of the DECnet-20 Version 2 (2040/50/60) system, you have only a few critical but simple entries to make:

- Set up the DECnet Software by copying several files from PS:<DNGEN> to PS:<SUBSYS>, and one site-specific front-end file, nodename.SYS from PS:<DN20-1> to PS:<SUBSYS>nodename.SYS. (If you configured a DN200, also copy the PS:<DN200>nodename.SYS file to PS:<SUBSYS>.nodename.SYS.)

- Locate and check the file SYSJOB.RUN on PS:<SYSTEM>. Edit this file if there are missing entries.

- Locate and check the file 4-CONFIG.CMD on PS:<SYSTEM>. Edit this file to include the NODE nodename nodenumber command.

- Locate and check the PTYCON.ATO file on PS:<SYSTEM> to include starting the NETCON program.

- Create an NCP.CMD file.

- Create a DN200.CMD file (if your system includes a DN200).

10.1 INSTALLATION PROCEDURE

NOTE

If you are updating your system from TOPS-20 V3A DECnet V1 to TOPS-20 V4 DECnet V2, commands with structure/directory PS:<SYSTEM> and PS:<SUBSYS> should be interpreted as PS:<NEW-SYSTEM> and PS:<NEW-SUBSYS>. You follow the same procedure that you used to install TOPS-20 V4. When the new versions of TOPS-20 and DECnet are performing satisfactorily, the old versions can be saved and <NEW-SYSTEM> and <NEW-SUBSYS> can be copied to <SYSTEM> and <SUBSYS> respectively.

All text and examples assume that your directories conform to the following conventions regarding contents:

PS:<SYSTEM> for original installation

or PS:<NEW-SYSTEM> for update-installation prior to checkout of new hardware/software

Contents: monitor and data and program files used by the monitor in normal operation

PS:<SUBSYS>

or PS:<NEW-SUBSYS>

Contents: all the system programs a user would normally wish to run

If your installation chooses to locate files other than exactly as suggested in text and examples, it is your responsibility to correctly interpret procedures according to your installation's file-location scheme. To do this you must remember the structure and directory where files have been placed.

Responses illustrated as part of the procedures are examples only. Because system files are system-dependent, the output you see may be different. For example, your site may not have four printers or a card reader.

Figure 10-1 is a flowchart of the installation procedure. Step numbers on the chart correspond to step numbers in the text.

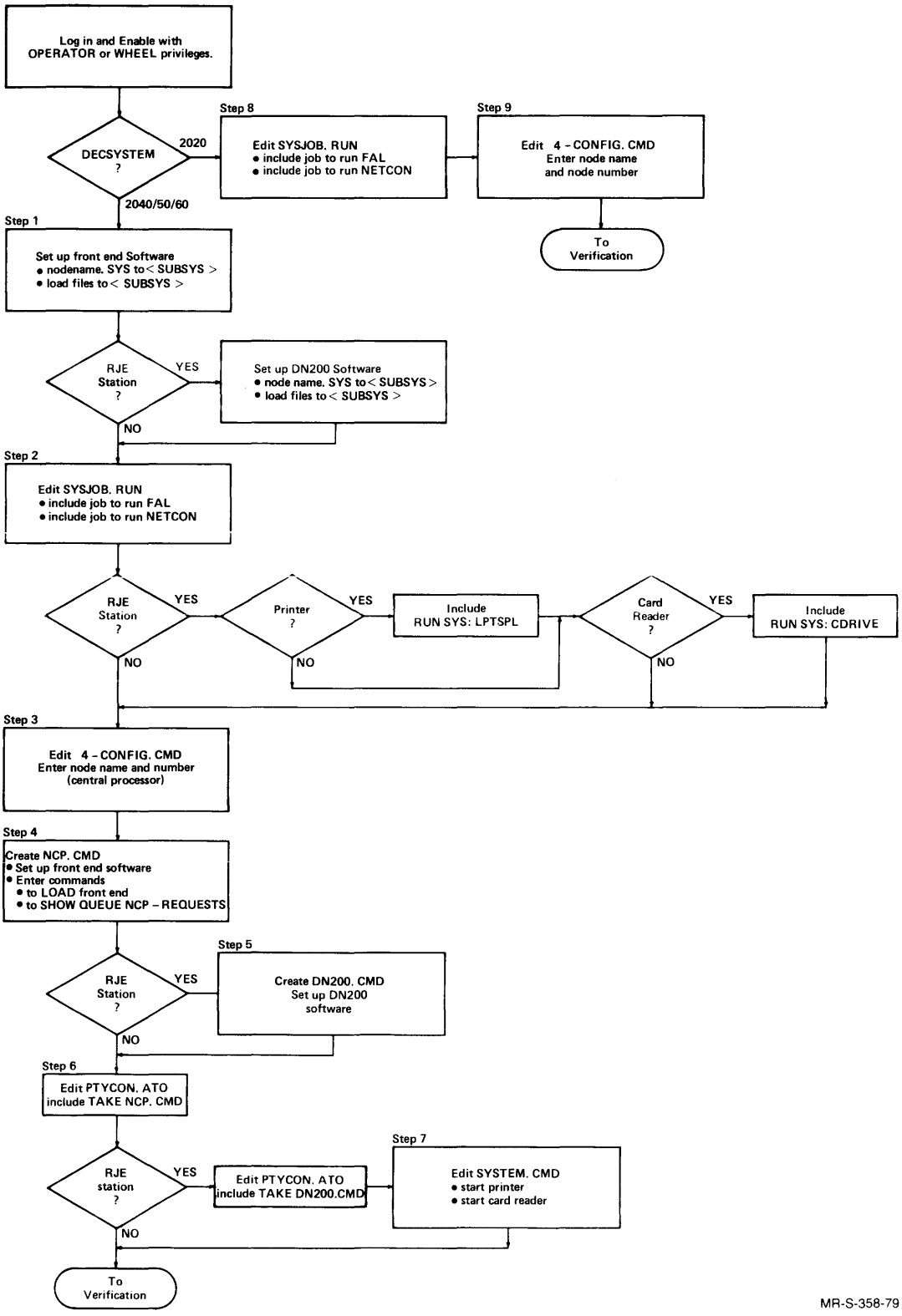


Figure 10-1 Installing DECnet-20 Software

When you are logged in and enabled with OPERATOR or WHEEL privileges:

- To install a DECSYSTEM-2040/50/60, perform Steps 1 through 7
- To install a DECSYSTEM-2020, perform Steps 8 and 9 only

Step 1: Set up the DECnet software.

If you have already executed any of the COPY commands listed below, do not repeat them. If you have not purchased the RJE option, do not give the DN200-related commands.

For the communications front end:

```
$COPY (FROM) PS:<DN20-1>nodename.SYS (TO) PS:<SUBSYS>nodename.SYS
```

The nodename parameter is the same as the response you gave to the NETGEN command NODE NAME for the communications front end.

```
$COPY (FROM) PS:<DNGEN>DTEMPS.BIN (TO) PS:<SUBSYS>DTEMPS.BIN

$COPY (FROM) PS:<DNGEN>DTEMPT.BIN (TO) PS:<SUBSYS>DTEMPT.BIN
```

For the DN200 remote station:

```
$COPY PS:<DN200>nodename.SYS (TO) PS:<SUBSYS>nodename.SYS
```

The nodename parameter is the same as the response you gave to the NETGEN command NODE NAME for the DN200.

If you are installing more than one DN200, each nodename.SYS file must be copied from the appropriate directory to the <SUBSYS> directory.

```
$COPY (FROM) PS:<DNGEN>SECDMC.SYS (TO) PS:<SUBSYS>SECDMC.SYS

$COPY (FROM) PS:<DNGEN>TERDMC.SYS (TO) PS<SUBSYS>TERDMC.SYS
```

Step 2: Edit SYSJOB.RUN.

In Steps 2 through 7 you are editing SYSTEM files. Any errors, such as duplication of commands, must be avoided. When you have completed each edit, use the EDIT command P^:* to check the entire file. Or, if you prefer, use the TYPE command to check the file before you edit it and again after you edit it.

Enter:

```
$DEFINE EDITOR: SYS:EDIT.EXE (RET)

        (ESC)
$CONNECT  (TO DIRECTORY) PS:<SYSTEM>(RET)

      (ESC)
$EDIT  (FILE) SYSJOB.RUN(RET)
```

The system responds:

```
EDIT:  SYSJOB.RUN.1
*
```

Enter:

*FJOB (ESC) (RET)

The system responds:

00800 JOB 0 /LOG OPERATOR XX OPERATOR

Enter:

*F (ESC) (RET)

The system responds:

01600 JOB 1 \ LOG OPERATOR XX OPERATOR

Enter:

*F (ESC) (RET)

Continue until the response is:

%Search fails

Enter:

This command is to find the line number to insert for your entry. You must judge by the line number of the last JOB number. The line number will probably be different from the example.

p1600: (RET)

The response will be the entire last JOB series. It will be similar to:

```
01600   JOB 1 \LOG OPERATOR XX OPERATOR
01700   ENA
01800   MFORK
01900   RUN SYS:ORION
02000   RUN SYS:QUASAR
02100   RUN SYS:LPTSPL/NAME:L1
02200   RUN SYS:LPTSPL/NAME:L2
02300   RUN SYS:LPTSPL/NAME:L3
02400   RUN SYS:LPTSPL/NAME:L4
02500   RUN SYS:CDRIVE
02600   RUN SYS:BATCON
02700   \
```

If you are entering at the end of the file, line numbers will appear automatically after you insert the first line; if you are not at the end, that is, there is at least one line following the last JOB series, use the command *I.!8.

If the last JOB series found is as shown above (lines 01600 to 02700),

Enter:

```
*i2800(RET)
02800   JOB 2 \LOG OPERATOR XX OPERATOR(RET)
02900   ENA(RET)
03000   RUN SYS:FAL(RET)
03100   \(RET)
03200   JOB 3 \LOG OPERATOR XX OPERATOR(RET)
03300   ENA(RET)
03400   RUN SYS:NETCON(RET)
03500   \(RET)
03600   $
*EU(RET)

[SYSJOB.RUN.3]
$
```

NOTE

If your configuration is to include RJE (DN200), check to ensure that there is a "RUN SYS:CDRIVE" command (if your remote station will have a card reader). Check to ensure that there are as many "RUN SYS:LPTSPL" commands as there are total printers (host site and RJE station combined). If SYSJOB.RUN does not meet these requirements, edit the file to include the necessary statements.

Step 3: Edit the 4-CONFIG.CMD.

Enter:

```
    (ESC)
     ↓
$EDIT (FILE) 4-CONFIG.CMD(RET)
```

The system responds:

```
EDIT:  4-CONFIG.CMD.9
*
```

Enter:

```
*I*(RET)
```

The system responds:

```
18200
(or whatever number follows the last line now in the file)
```

Enter:

```
18200   NODE nodename nodenumber(RET)
18300   $
```

This the node name and node number of the host or central processor.

```
*EU(RET)
```

Step 4: Create an NCP.CMD file.

Entries and responses:

```
$CREATE NCP.CMD(RET)
INPUT:  NCP.CMD.1
00100     ENTER NCP(RET)
00200     SET EXECUTOR servername(RET)
          (servername is the node name for the host or
          central processor)
00300     SET SECONDARY-LOAD-FILE DTE20 PDP-11 PS:<SUBSYS>DTEMPS.BIN(RET)
00400     SET TERTIARY-LOAD-FILE DTE20 PDP-11 PS:<SUBSYS>DTEMPT.BIN(RET)
00500     SET NODE nodename SERVER servername DTE20_1(RET)
          (servername is the node name for the host, nodename in this
          and subsequent lines is the front-end node name)
00600     SET NODE nodename LOAD-FILE PS:<SUBSYS>nodename.SYS(RET)
00700     SET NODE nodename DUMP-FILE PS:<SUBSYS>nodename.DMP(RET)
00800     SET NODE nodename PROTOCOL-TYPE NETWORK-SERVICES-PROTOCOL(RET)
00900     LOAD NODE nodename(RET)
01000     SHOW QUEUE NCP-REQUESTS(RET)
01100     RETURN(RET)
01200     $
*EU(RET)

[NCP.CMD.1]
$
```

NOTE

The severname (host node name) that you use in the above commands must be identical to the name used in editing the 4-CONFIG.CMD file in Step 3.

Step 5: Create a DN200.CMD file if your system includes a DN200.

Entries and responses:

```
$CREATE DN200.CMD(RET)
INPUT: DN200.CMD.1
00100   ENTER NCP(RET)
00200   SET EXECUTOR servername(RET)
```

NOTE

The servername above is the node name of the host or central processor.

```
00300   SET SECONDARY-LOAD-FILE DMC PDP-11 PS:<SUBSYS>SECDMC.SYS(RET)
00400   SET TERTIARY-LOAD-FILE DMC PDP-11 PS:<SUBSYS>TERDMC.SYS(RET)
```

NOTE

If more than one DN200 is in your configuration, repeat lines 500,600, and 700 which follow. Give specific parameters for each DN200.

```
00500   set node nodename SERVER servername KDP_0_2
```

NOTE

The servername above is the name of the DECnet communications front end. The controller number is the number of the front-end controller that controls the line to the DN200 (line 2 in the example). The nodename in the above and all subsequent lines is the nodename of the DN200.

```
00600   SET NODE nodename LOAD-FILE PS:<SUBSYS>nodename.SYS (RET)
00700   SET NODE nodename DUMP-FILE PS:<SUBSYS>nodename.DMP (RET)
00800   RETURN (RET)
00900   $
*EU (RET)
```

Step 6: Edit PTYCON.ATO

Entry and response:

```
$EDIT PTYCON.ATO (RET)
Edit:  PTYCON.ATO.5
*
```

Entry and response:

```
                          (ESC)
                            ↓
*FTAKE SYSTEM:SYSTEM.CMD (RET)
01100 TAKE SYSTEM:SYSTEM.CMD
*
```

Entry and response for DN20 without DN200:

```
*I.!1 (RET)
01150 TAKE SYSTEM:NCP.CMD (RET)
*EU (RET)
```

Entry and response for DN20 with DN200:

```
*I.!2 (RET)
01120 TAKE SYSTEM:NCP.CMD (RET)
01140 TAKE SYSTEM:DN200.CMD (RET)
*EU (RET)
```

NOTE

If your system does not include a DN200, you have completed the installation procedures. Go to Section 10.2. If your system does include a DN200, continue with Step 7.

Step 7: Edit SYSTEM.CMD.

Entry and response:

```
$EDIT PS:<SYSTEM>SYSTEM.CMD (RET)
Edit:  SYSTEM.CMD
```

Print the file using edit and locate a logical place to insert one or two commands. The edit procedures in the example given may need to be modified to fit the contents of your system's SYSTEM.CMD file.

The commands to be added are:

If the DN200 has a printer, enter:

```
START PRINTER 0/NODE:DN200::
```

If the DN200 has a card reader, enter:

```
START READER 0/NODE:DN200::
```

(The DN200:: given above represents the node name you specified for the DN200 at generation time.)

Example:

```
$edit system.cmd (RET)
Edit: SYSTEM.CMD.1
*p^:* (RET)
00100   MOUNT STRUCTURE SNARK:
00200   MOUNT STRUCTURE LANG:
00300   MOUNT STRUCTURE MISC:
00400   SET BATCH-STREAM 0 TIME-LIMIT 5
00500   SET BATCH-STREAM 1 TIME-LIMIT 10
00600   SET BATCH-STREAM 2 TIME-LIMIT 11000
00700   SET BATCH-STREAM 3 TIME-LIMIT 11000
00800   SET BATCH-STREAM 3 PRIORITY-LIMITS 20:63
00900   SET BATCH-STREAM 0:2 PRIORITY-LIMITS 1:19
01000   START BATCH-STREAM 0:3
01100   SET PRINTER 0 PAGE-LIMIT 20000
01200   SET PRINTER 1 PAGE-LIMIT 500
01300   START PRINTER 0:1
01400   START READER 0
01500   DISABLE OUTPUT-DISPLAY ALL-MESSAGES /JOB
*i.-1!2 (RET)
01420   START PRINTER 0/NODE:DN200::(RET)
01440    TART READER 0/NODE:DN200::(RET)
*p1300:* (RET)
01300   START PRINTER 0:1
01400   START READER 0
01420   START PRINTER 0/NODE:DN200::
01440   START READER 0/NODE:DN200::
01500   DISABLE OUTPUT-DISPLAY ALL-MESSAGES /JOB
*eu (RET)

[SYSTEM.CMD.2]
$
```

You have completed DECSYSTEM-2040/50/60 installation. Continue with Section 10.2.

Step 8: Edit SYSJOB.RUN (DECSYSTEM-2020)

In Steps 8 and 9, you are editing system files. Any errors, such as duplication of commands, must be avoided. When you have completed each edit, use the EDIT command p^:* to check the entire file. Or, if you prefer, use the TYPE command to check the file before and after you edit it.

Enter:

```
$DEFINE EDITOR: SYS:EDIT.EXE (RET)

        (ESC)
$CONNECT (TO DIRECTORY) PS:<SYSTEM> (RET)

      (ESC)
$EDIT (FILE) SYSJOB.RUN (RET)
```

The system responds:

```
EDIT:  SYSJOB.RUN.1
*
```

Enter:

*FJOB (ESC) (RET)

The system responds:

```
00800   JOB 0 /LOG OPERATOR XX OPERATOR
```

Enter:

*F (ESC) (RET)

Continue until the response is:

%Search fails

Enter:

This command is to find the line number to insert for your entry. You must judge by the line number of the last JOB number. It will probably be different from the example.

p800: (RET)

The response will be the entire last JOB series. It will be similar to:

```
00800  JOB 1 /LOG OPERATOR XX OPERATOR
00900  ENA
01000  ^ESET LOGIN PSEUDO
01100  ^ESET LOGIN CONSOLE
01200  ^ESET OPERATOR
01300  PTYCON
01400  GET SYSTEM:PTYCON.ATO
01500  /
```

If you are entering at the end of the file, line numbers will appear automatically after you insert the first line; if you are not at the end, that is, there is at least one line following the last JOB series, use the command *I.!8.

NOTE

Do NOT enter either of the jobs below if they have already been entered.

If the last JOB series found is as shown above (lines 00800 to 01500),

Enter:

```
*i1600(RET)
01600  JOB 2 \LOG OPERATOR XX OPERATOR(RET)
01700  ENA(RET)
01800  RUN SYS:FAL(RET)
01900  \(RET)
02000  JOB 3 \LOG OPERATOR XX OPERATOR(RET)
02100  ENA(RET)
02200  RUN SYS:NETCON(RET)
02300  \(RET)
02400  $
*EU(RET)

[SYSJOB.RUN.3]
$
```

Step 9: Edit the 4-CONFIG.CMD (DECSYSTEM-2020)

Enter:

```
      (ESC)
$EDIT (FILE) 4-CONFIG.CMD(RET)
```

The system responds:

```
EDIT:  4-CONFIG.CMD.9
*
```

Enter:

```
*I*(RET)
```

The system responds:

```
18200
(or whatever number follows the last line now in the file)
```

Enter:

```
18200   NODE nodename nodenumber(RET)
```

This is the node name and node number (decimal) of the host or central processor.

```
18300   $
*EU(RET)
```

10.2 INSTALLATION VERIFICATION PROCEDURE

The DECnet-20 installation verification procedure ensures that the appropriate DECnet-20 software modules have been installed. The verification procedure includes:

- Checking for the existence of each required software module
- Checking for the correct version of each required software module

The verification tests are run under the User Environmental Test Package (UETP). Before you begin the verification procedures, you must satisfy the requirements of UETP.

Examine your login-directory. The directory to which you login must include the following two entries:

```
IPCF
USER GROUPS nnnn,100
```

The nnnn represents your assigned user group for normal activity. The 100 that follows is required by UETP. Use the TOPS-20 command BUILD to add the two required subcommands.

Figure 10-2 is a flowchart of the verification procedure. Step numbers on the chart correspond to step numbers in the text.

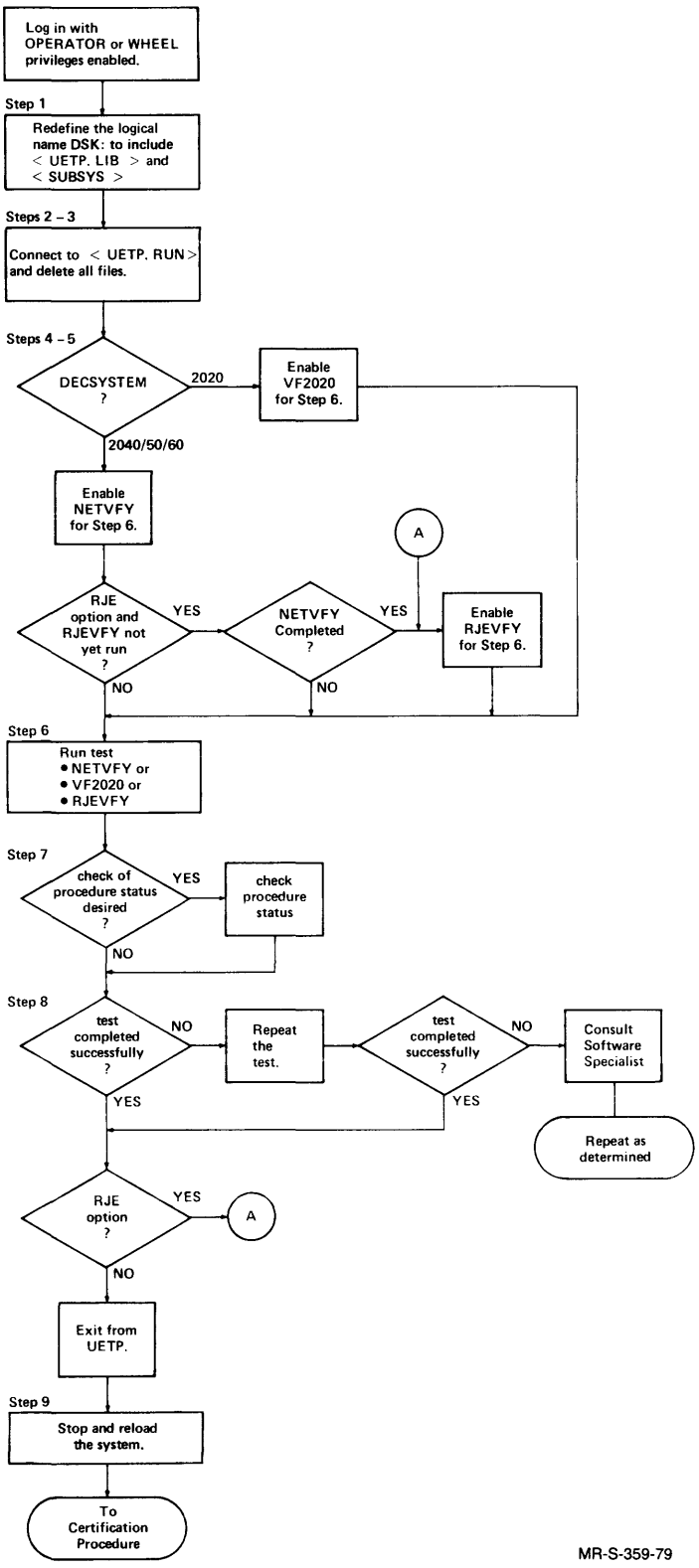


Figure 10-2 Verifying the DECnet-20 Software

Go to Step 1 when you are logged in with OPERATOR or WHEEL privileges enabled.

Step 1: Redefine the logical name DSK:.

Redefine the logical name DSK: to include the directories <UETP.LIB> and <SUBSYS> as part of the file search sequence. To verify the execution of the command, use the INFORMATION LOGICAL-NAME command.

Enter:

```
$DEFINE (LOGICAL NAME) DSK:  DSK:,<UETP.LIB>,<SUBSYS>
```

```
$INFORMATION (ABOUT) LOGICAL-NAMES
```

The system responds:

```
DSK:  => DSK:,<UETP.LIB>,<SUBSYS>
$
```

Step 2: Connect to directory <UETP.RUN>.

Enter:

```
$CONNECT (TO DIRECTORY) <UETP.RUN>
```

Step 3: Ensure that <UETP.RUN> is empty.

Enter:

```
$DELETE (FILES) *.*.*
```

The system responds:

```
filename1 [OK]
filename2 [OK]
     .
     .
     .
$
```

(if files were found and deleted)

or:

```
%NO FILES MATCH THIS SPECIFICATION- *.*.*
$
```

(if the directory was empty).

Step 4: RUN the UETP program.

Enter:

```
$RUN UETP (RET)
```

The system responds:

```
[da-mon-yr hh:mm:ss USER ENVIRONMENT TEST PACKAGE]
UETP>
```

Step 5: Enable the DECnet-20 verification test.

Enter:

```
          (ESC)
          ↓
UETP>ENABLE (TEST) prgnam (RET)
```

where:

prgnam = NETVFY for a DECSYSTEM-2040/50/60
= VF2020 for a DECSYSTEM-2020
= RJEVFY for DECSYSTEM 2040/50/60 with optional addition of remote station DN200

NOTE

If you are validating RJE, NETVFY must perform successfully before you run RJEVFY.

The system responds:

```
 hh:mm:ss [ENABLE COMPLETED]
UETP>
```

Step 6: Begin the verification procedure.

The system confirms the BEGIN command and displays the messages shown below as verification progresses.

Enter:

```
UETP>BEGIN (RET)
```

The system responds:

```
 hh:mm:ss [BEGIN COMPLETED]
UETP>

 START prgnam da-mon-yr hh:mm:ss message
 MAJOR prgnam da-mon-yr hh:mm:ss message
 END   prgnam da-mon-yr hh:mm:ss message
```

(prgnam is the name of the file enabled in Step 5.)

You may check the status of UETP at any time. Use the procedure shown in Step 7.

Step 7: Check status of verification procedure.

Enter:

STATUS (RET)

The system may respond:

```
[da-mon-yr hh:mm:ss]

 TEST NAME      STATUS TIMES TO TIMES ERROR    START
 (FILE NAME)           BE RUN   RUN   COUNT     TIME
 ===========   ===== ======== ===== ======    =====
 prgnam.SUP    queued       1     0      0  da-mon-yr hh:mm:ss

UETP>
```

The status QUEUED means that the verification process is waiting to be run. STATUS may also be given as RUNNING or ENDED. This status output will be given automatically when status is ENDED. When ENDED is displayed, continue with Step 8. If errors occur, refer to Section 10.2.1.

Step 8: Exit from UETP

Entry and response:

```
    (ESC)
     ↓
 EXIT (TO MONITOR) (RET)
$
```

Step 9: Stop and reload the system if test completed successfully.

Now that you have restored all the DECnet-20 software to the proper directories, modified the appropriate command files, and verified the software, you must first stop the system and reload it to activate DECnet-20.

Enter:

$(CTRL/^) (does not echo)

The system responds:

```
PAR> (for  DECSYSTEM 2040/50/60)
KS10> (for DECSYSTEM-2020)
```

Enter:

sh (RET)

The system shuts down with appropriate messages.

To reload the DECSYSTEM-2040/50/60:

Press the ENABLE and DISK switches simultaneously.

To reload the DECSYSTEM-2020:

Press the BOOT switch.

After the system completes printing loading and configuration messages, the system displays ENTER CURRENT DATE AND TIME:.

Enter:

day-mon-yr hhmm (followed by carriage return)

The system repeats the date and time you have entered and asks for confirmation. Answer with Y or N, and repeat if necessary.

The system then asks WHY RELOAD?

Type an abbreviation that indicates the reason for the reload, for example, CTF for certification. The answer you type here is stored in the system error file and reported by SYSERR. (If your site has a code or standard abbreviation, use the appropriate entry.)

The system asks:

RUN CHECKD?

Enter:

N (RET)

The system now executes all the command files that load and start the various system tasks. These will now include the DECnet-20 tasks.

10.2.1 **Verification Procedure Messages**

Verification tests pass confirmation, event, and error messages to the UETP program. The various UETP elements route these messages to one or more of the following:

- your terminal
- the file RUN.LOG
- the file EXCEPT.LOG

In addition, batch generates a Batch Log file for each test run under UETP. This file is named prgnam.LOG, where prgnam is the name of the DECnet-20 verification test. This log file is appended to the file prgnam.ERL if a test reports an error to UETP via the SENDER program.

Thus, if any test does not complete successfully, there are, in addition to the messages output to your terminal, three files that may be used to discover the problem. The table that follows summarizes message types and their destinations.

Table 10-1
SENDER Message Types and Destinations

MESSAGE TYPE	DESTINATION			
	TERMINAL	EXCEPT.LOG	RUN.LOG	prgnam.ERL
START	yes	no	yes	no
MAJOR	yes	no	yes	no
MINOR	no	no	yes	no
ERROR	yes	yes	yes	yes
END	yes	yes	yes	no
LOG	no	no	yes	no

If a test executes successfully, the following messages will be sent to your terminal in the order given:

START prgnam date time Start of verification

MAJOR prgnam date time Verifying file:<UETP.LIB>DECNET.VFY

END prgnam date time End of verification

If errors occur, one or more of the following will be sent to your terminal:

ERROR prgnam date time VERIFY program not available

The program VERIFY.EXE could not be found. This program is required to perform the verification. If you have made no errors, this probably indicates an error in the TOPS-20 installation procedures. YOU CANNOT CONTINUE UNTIL THE VERIFY PROGRAM IS IN <SUBSYS> or <NEW-SUBSYS>.

ERROR prgnam date time DECNET.VFY file (input to the VERIFY program) is not available.

This probably indicates an error in your DECnet installation procedures. YOU CANNOT CONTINUE UNTIL DECNET.VFY IS IN <UETP.LIB>.

ERROR prgnam date time ERRORS running VERIFY program

One or more files did not verify correctly. Either a required file is not present, a required file is in the wrong directory, or the data is incorrect (determined by noting that the file's checksum is different from that of the file shipped by DIGITAL). Return to the DECnet installation procedure and correctly restore the files that did not verify. If the files are not available, notify your DIGITAL Software Specialist.

ERROR prgnam date time Verification procedure exceeded CPU time limit

This message indicates a problem with the verification procedure itself. Notify the DIGITAL Software Specialist.

ERROR prgnam date time Unknown error in VERIFY test

This message indicates conditions for which no specific check could be made. Consult the DIGITAL Software Specialist.

10.3 INSTALLATION CERTIFICATION PROCEDURE

You have now determined that the correct version of each required DECnet-20 software module has been installed. In the certification procedures you will ensure that the various components of your DECnet-20 system function at a basic, minimal level. The verification and certification procedures are sometimes referred to as "acceptance tests". Together they confirm the acceptability of the installed hardware and software to the extent successfully tested.

The certification tests to be run include:

1. Local-NCP test
2. Line-loopback test
3. Remote-NCP test
4. Remote node file transfer test
5. DN200 test
6. Error and event logging test

The line-loopback test and the remote node file transfer test are run as batch jobs under UETP. For these two tests both general procedures and error messages are the same as for the verification procedures you have completed. The other four tests are run in production mode. That is, you type commands to a program or you tell the program to take the commands from a command file you create.

All files associated with installation verification and certification of DECnet-20 Version 2 are in one of two directories, as follows:

Directories and files for DECSYSTEM-2040/50/60 certification (and verification):

```
  PS:<UETP.LIB>
CHARS.TXT
DECNET.DAT
DNV2FT.APP
DNV2LN.CMP
DYNETS.EXE
NETVFY.VER
NULL.TXT
PICTUR.TXT
RJEVFY.VER

  PS:<UETP.DECNET>
BRIEF.TXT
DETAIL.TXT
LOCAL-NCP.CMD
LOCAL-NCP-KL-EXAMPLE.CMD
  .TXT
REMOTE-NCP.CMD
REMOTE-NCP-20-EXAMPLE.CMD
  .TXT
REMOTE-NCP-DN20-EXAMPLE.CMD
  .TXT
```

Directories and files for DECSYSTEM-2020 certification (and verification):

```
  PS:<UETP.LIB>
DECNET.DAT
DNV2FT.APP
DNV2LL.CMP
DNWAIT.EXE
DYNETS.EXE
NULL.TXT
TEST72.CMP
VF2020.VER

  PS:<UETP.DECNET>
BRIEF.TXT
DETAIL.TXT
LOCAL-NCP.CMD
LOCAL-NCP-KS-EXAMPLE.CMD
  .TXT
REMOTE-NCP.CMD
REMOTE-NCP-20-EXAMPLE.CMD
  .TXT
```

Note that the <UETP.LIB> directory contains all control files, command files to be edited to match your configuration, and original test data files. You will create several files to add to PS:<UETP.LIB> as you perform the various tests. These are the files that must be kept. They will be needed to repeat certification procedures if new DECnet-20 hardware/software is brought on line. They will be helpful in constructing user application tests to test production programs that normally run under batch. (For additional information on constructing your own tests, refer to the TOPS-20 User Environment Test Package Procedures/Reference Manual.)

The files on the <UETP.DECNET> directory contain sample command files and sample output files. They serve to help you understand what to do and how to do it. They may be saved or deleted, according to your needs, after the certification completes successfully.

Before you begin certification testing you must:

- Have a suitable loopback connector for each line to be tested (later you will be told at what point to install it)
- Create certain files to establish values specific to your configuration

10.3.1 Loopback Connector

In order to run the certification procedure for DECnet-20 without requiring another DECnet node to talk to, it is necessary to use some form of echoing device. A device known as a loopback connector may be attached to the line interface in place of the modem. All data directed to the line will then be echoed back to the sender.

Loopback connectors come in various models. DECSYSTEM-20s with communications hardware are supplied with the model loopback connector that is suitable for the particular communications device being used. Loopback connectors should be installed by DIGITAL Software Service or Field Service representatives. To test all DECnet lines in one run (the line loopback test), you will need one connector for each line to be tested.

Several of the acceptance tests may be run by either direct terminal input or by using a command file; some tests are run under UETP while others are not. There are also system dependencies. Use the flowcharts (Figure 10-3) as an overview and checklist as you perform the tests. (Connectors may span pages.)

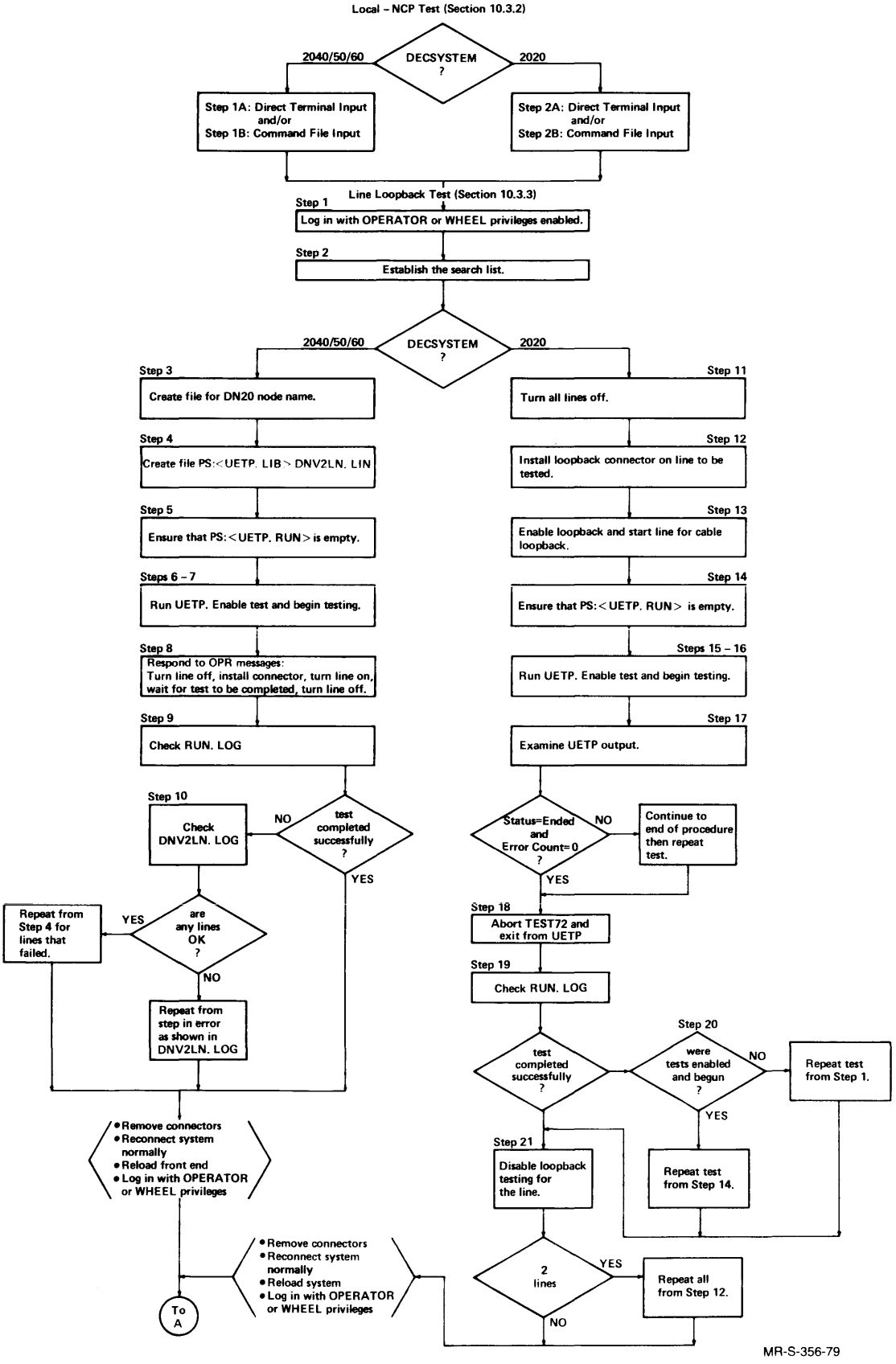


Figure 10-3 Certifying the DECnet-20 Software

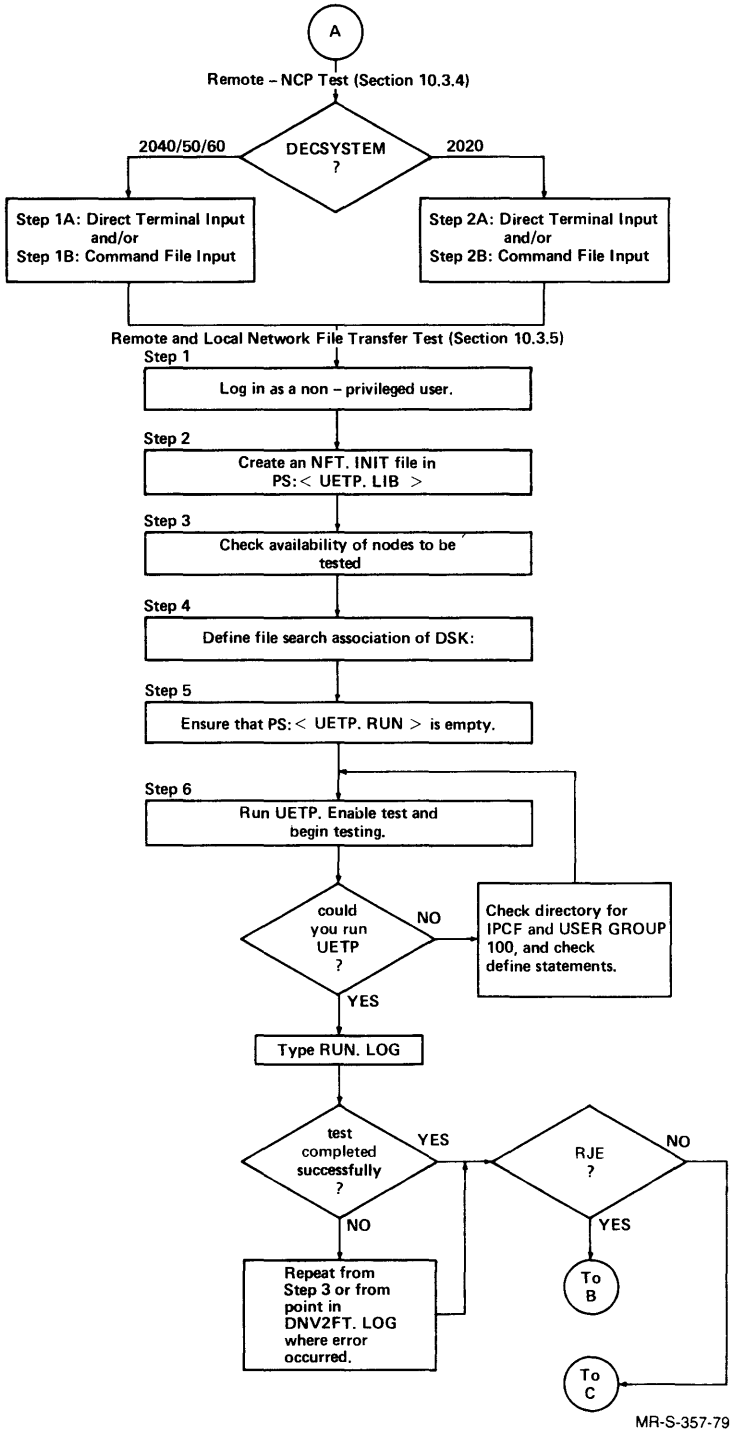


Figure 10-3 (Cont.) Certifying the DECnet-20 Software

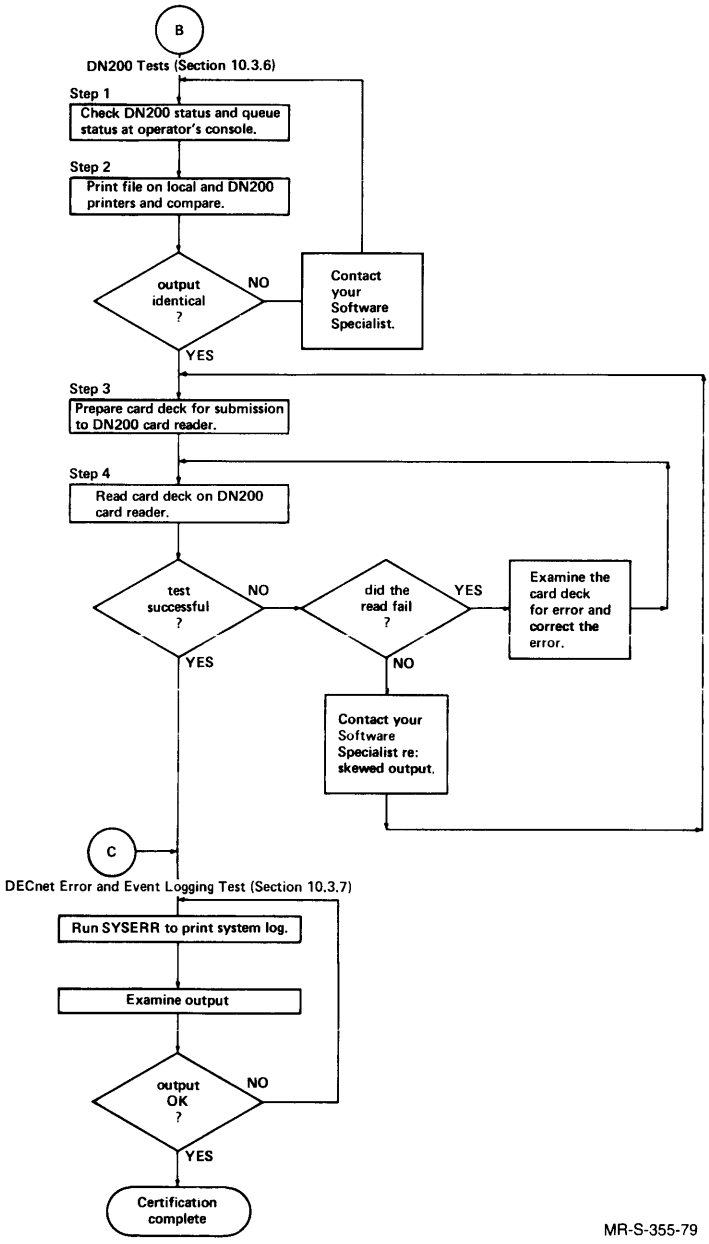


Figure 10-3 (Cont.) Certifying the DECnet-20 Software

10.3.2 Local-NCP Test

The commands to be executed for the LOCAL-NCP test may be typed directly at your terminal or entered by typing a TAKE command to NCP via OPR.

If your host node is a DECSYSTEM-2040/50/60, go to Step 1A if you wish to use direct input to your terminal; go to Step 1B if you wish to use a command file.

If your host node is a DECSYSTEM-2020, go to Step 2A if you wish to use direct input to your terminal; go to Step 2B if you wish to use a command file.

Step 1A. Local-NCP test using direct terminal input (DECSYSTEM-2040/50/60)

Enter:

```
ESC
@enaBLE (CAPABILITIES)RET
$type ps:<uetp.decnet>local-ncp-k1-example.cmdRET
```

Read the example file. Refer to the seven commands below the second line of asterisks. To test YOUR local NCP, certain changes are required:

The first command will be:

SET EXECUTOR your local host name

You entered the node name of your central processor in the 4-CONFIG.CMD file in Step 3 of the Installation Procedures (Section 10.1). The node name in the SET EXECUTOR command must agree with the node name in the 4-CONFIG.CMD file.

The SHOW STATUS LINE lineid command(s) must also be changed to reflect your configuration. You include the line identification of each DTE20 to be tested.

Follow the directions given in the example-file. Type the commands, making the changes indicated. If you wish to see what the output should look like, enable and type the file PS:<UETP.DECNET>LOCAL-NCP-EXAMPLE.TXT. This example is the result of a TAKE command and includes a list of outstanding NCP requests in response to the SHOW QUEUE NCP-REQUESTS command. If you enter the commands directly, you may not receive this list because your typing speed is slower than the response speed. An example of direct entry and response appears below. Your typed input will differ in the SET EXECUTOR and SHOW STATUS LINE commands.

Example of direct entry and response:

```
@ENABLE (CAPABILITIES)<RET>
$OPR<RET>
OPR>DISABLE outPUT-DISPLAY (of) all-MESSAGES<RET>
OPR>
12:56:17        --OUTPUT DISPLAY for OPR Modified--
OPR>enter ncp<RET>

NCP>set exeCUTOR 2102<RET>

NCP>show exeCUTOR<RET>
NCP>
12:56:49                Current EXECUTOR is node 2102

NCP>show status loCAL<RET>
NCP>
12:57:10          NCP request # 2 [SHOW STATUS LOCAL]

     Status of local node as of 16-Jul-79 12:57:10

     Node Name is 2102, # 64, System = 2102 Development System,
     TOPS-20 Routing Version = 3.0.0, Communications Version = 3.0.0
     State is On, Default Host = Unknown

     Function completed successfully

NCP>show sTATUS knoWN liNES<RET>
NCP>
12:57:54          NCP request # 3 [SHOW STATUS KNOWN LINES]

                Status as of 16-Jul-79 12:57:54

                Line ID         State           Adjacent Node

                DTE20_0_0_0     OFF
                DTE20_1_0_0     On              DN20A
                DTE20_2_0_0     Off
                DTE20_3_0_0     Off

                Function completed successfully
```

```
     (ESC) (ESC)    (ESC)
NCP>shOW stATUS liNE DTE20_1(RET)
NCP>
12:58:41          NCP request # 4 [SHOW STATUS LINE DTE20 _ 1]

                  Status as of 16-Jul-79 12:58:41

                  Line ID         State           Adjacent Node

                  DTE20_1_0_0     On              DN20A

                  Function completed successfully

     (ESC) (ESC)    (ESC)
NCP>shOW sTATUS liNE dte20_2(RET)
NCP>
12:59:20          NCP request # 5 [SHOW STATUS LINE DTE20 _ 2]

                  Status as of 16-Jul-79 12:59:20

                  Line ID         State           Adjacent Node

                  DTE20_2_0_0     Off

                  Function completed successfully


          (ESC)          (ESC)
NCP>show qUEUE (OF) ncP-REQUESTS(RET)
NCP>
12:59:52

                  There are no outstanding NCP requests.

NCP>exit(RET)

    (ESC)
$disaBLE (CAPABILITIES)(RET)
@
```

If you wish to repeat the LOCAL-NCP test using a command file, go to Step 1B; otherwise go to Section 10.3.3.

Step 1B. Local-NCP test using a .CMD file (DECSYSTEM-2040/50/60)

The file LOCAL-NCP.CMD in the directory PS:<UETP.DECNET> contains the commands to test the local NCP. One SET EXECUTOR command and two SHOW STATUS LINE commands are given with dummy arguments. These dummy arguments must be edited to reflect the node name and line identifications of YOUR configuration. The number of SHOW STATUS LINE commands must equal the number of DTE20 lines you wish to test. All lines you plan to use for network traffic should be tested. The host or central processor node name in the SET EXECUTOR command must be identical to the name you entered in the 4-CONFIG.CMD file in Section 10.1.

The edited file should be output to PS:<UETP.LIB> as shown in the example. Use the EDIT command EU to remove line numbers when you exit from EDIT. Including the host node name in the output file name will help identify it for future use.

Example entries and responses:

```
@ENA (RET)
$CONN PS:<UETP.DECNET> (RET)

                    (ESC)
$edit local-ncp.cmd.4 (OUTPUT AS) PS:<UETP.LIB>LOCAL-NCP-2102.CMD (RET)
Edit: LOCAL-NCP.CMD.4
*p^:* (RET)
00100   !************************************************************************!
00200   !                                                                        !
00300   !  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED       !
00400   !    OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.             !
00500   !                                                                        !
00600   !COPYRIGHT (C) 1978,1979 BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS!
00700   !                                                                        !
00800   !       DECnet-20 V2 Local NCP test                                      !
00900   !                                                                        !
01000   !                                                                        !
01100   !To run this test start OPR in a job enabled with wheel or operator     !
01200   !privileges and enter the NCP command subset:                           !
01300   !                                                                        !
01400   !       @ENABLE                                                          !
01500   !       $OPR                                                             !
01600   !       OPR>ENTER NCP                                                    !
01700   !       NCP>                                                             !
01800   !                                                                        !
01900   !Then either enter:                                                      !
02000   !                                                                        !
02100   !       NCP>TAKE REMOTE-NCP.CMD                                          !
02200   !                                                                        !
02300   !to execute the statements in the command file or                       !
02400   !type the following commands by hand.                                   !
02500   !                                                                        !
02600   !************************************************************************!
02700
02800   SET EXECUTOR host-node-name
02900   SHOW EXECUTOR
03000   SHOW STATUS LOCAL
03100   SHOW STATUS KNOWN LINE
03200   SHOW STATUS LINE line
03300   SHOW STATUS LINE line
03400   SHOW QUEUE NCP-REQUESTS
*shost-node-name$2102$2800 (RET)
02800   SET EXECUTOR 2102
*sline$DTE 20_0$3200,E (RET)
03200   SHOW STATUS LINE DTE20_0
*sline$DTE 20_1$3300,E (RET)
03300   SHOW STATUS LINE DTE20_1
*p2800:3400 (RET)
02800   SET EXECUTOR 2102
02900   SHOW EXECUTOR
03000   SHOW STATUS LOCAL
03100   SHOW STATUS KNOWN LINE
03200   SHOW STATUS LINE DTE20_0
03300   SHOW STATUS LINE DTE20_1
03400   SHOW QUEUE NCP-REQUESTS
*eu (RET)


[<UETP.LIB>LOCAL-NCP-2102.CMD.1]
```

When you have exited from EDIT,

Enter:

```
CONN PS:<UETP.LIB>(RET)
$OPR(RET)
OPR>ENTER NCP(RET)
NCP>TAKE LOCAL-NCP-xxxx.CMD(RET)
```

where xxxx is the node name of the local host

OUTPUT FOLLOWS

Enter:

```
NCP>EXIT(RET)
```

Check your output as the test is being performed. No error messages should be printed. The output should have the same format as the file LOCAL-NCP-KL-EXAMPLE.TXT in PS:<UETP.DECNET>. This example is for a DECSYSTEM-2060 node with the node name 2102. Type the example file to compare it with your output. The output from your .CMD file should be similar in format to the example typed. Node names and line identifications may differ.

When the test completes successfully, go to Section 10.3.3.

Step 2A. Local-NCP test using direct terminal input (DECSYSTEM-2020)

Enter:

```
(ESC)
@enaBLE (CAPABILITIES)(RET)
$type ps:<uetp.decnet>local-ncp-ks-example.cmd(RET)
```

Read the example file. Refer to the seven commands below the second line of asterisks. To test YOUR local NCP, certain changes are required:

The first command will be:

SET EXECUTOR your local host name

The node name in the SET EXECUTOR command must agree with the node name in the 4-CONFIG.CMD file in PS:<SYSTEM>.

The SHOW STATUS LINE lineid command(s) must also be changed to reflect your configuration. You include the line identification of each DECnet-20 line to be tested. For one line, your lineid will be KDP_0_0. For two lines, the lineids will be KDP_0_0 and KDP_0_1.

Now, type the commands (six or seven commands depending on the number of lines) shown in the example, making the changes indicated for node name and lineids. Check the output as the commands are being performed. There should be no error messages. When the response to the last command is complete, type EXIT following the NCP prompt. Compare your output with that of the example given below. The name of the 2020 node in the example is 4097. The 4097 has two DECnet-20 lines. The DISABLE OUTPUT-DISPLAY command in the example is needed only if you are testing under timesharing. If you fail to give this command, messages from OPR such as the one shown before the DISABLE OUTPUT-DISPLAY command was executed will be scattered throughout the responses to your commands.

Example:

```
@ENA(RET)
$opr(RET)
OPR>disable output-display all messages(RET)
OPR>
10:53:37          --MTA2: Volume DAVESS, ANSI labeled tape mounted--
OPR>
10:53:42        --OUTPUT DISPLAY for OPR Modified--
OPR>enter ncp(RET)
NCP>set executor 4097(RET)
NCP>show executor(RET)
NCP>
10:54:42               Current EXECUTOR is node 4097
NCP>show status known lines(RET)
NCP>
10:55:10          NCP request # 8 [SHOW STATUS KNOWN LINES]

                  Status as of 18-Jul-79 10:55:10

                  Line ID         State          Adjacent Node

                  KDP_0_0_0       On             2102
                  KDP_0_1_0       On

                  Function completed successfully

NCP>show status line kdp_0_0(RET)
NCP>
10:55:56          NCP request # 9 [SHOW STATUS LINE KDP _ 0 _ 0]

                  Status as of 18-Jul-79 10:55:56

                  Line ID         State          Adjacent Node

                  KDP_0_0_0       On             2102

                  Function completed successfully
```

```
NCP>show status line kdp_0_1(RET)
NCP>
10:56:44          NCP request # 10 [SHOW STATUS LINE KDP _ 0 _ 1]

                  Status as of 18-Jul-79 10:56:43

                  Line ID         State          Adjacent Node

                  KDP_0_1_0       On

                  Function completed successfully

NCP>show counts line kdp_0_0(RET)
NCP>
10:57:47          NCP request # 11 [SHOW COUNTS LINE KDP _ 0 _ 0]
        Counts for line KDP_0_0_0, as of 18-Jul-79 10:57:47
        Seconds since zeroed      27321
        Blocks received           11493
        Received line errors          0
        Blocks sent               13158
        Retran, line errors           0
        Retran, not line erros        0
        Resource errors               1

                  Function completed successfully

NCP>show counts line kdp_0_1(RET)
NCP>
10:58:29          NCP request # 12 [SHOW COUNTS LINE KDP _ 0 _ 1]
        Counts for line KDP_0_1_0, as of 18-Jul-79 10:58:29
        Seconds since zeroed      27361
        Blocks received               0
        Received line errors          0
        Blocks sent                   0
        Retran, line errors           0
        Retran, not line errors       0
        Resource errors               0

                  Function completed successfully

NCP>show queue ncp-requests(RET)
NCP>
10:59:13
                  There are no outstanding NCP requests.

NCP>exit(RET)
$
```

If you wish to repeat the LOCAL-NCP test using a command file, go to Step 2B; otherwise go to Section 10.3.3.

Step 2B. Local-NCP test using a .CMD file (DECSYSTEM-2020)

The file LOCAL-NCP.CMD in the directory PS:<UETP.DECNET> contains the commands to test the local NCP. One SET EXECUTOR command and two SHOW STATUS LINE commands are given with dummy arguments. These dummy arguments must be edited to reflect the node name and line identifications of YOUR configuration. The number of SHOW STATUS LINE commands must equal the number of KDP lines you wish to test. All lines you plan to use for network traffic should be tested. The host or central processor node name in the SET EXECUTOR command must be identical to the node name in the 4-CONFIG.CMD file.

The edited file should be output to PS:<UETP.LIB> as shown in the example. Use the EDIT command EU to remove line numbers when you exit from EDIT. Including the host node name in the output file name will help identify it for future use. In the example below, the node name is 4097. There are two DECnet-20 lines. You substitute the parameters that are correct for your node. In editing the command SHOW STATUS LINE line, be sure to include the ",E" as shown in the example. If you fail to do this, the lineid will be substituted two times.

Example entries and responses:

```
@ENA(RET)
$CONN PS:<UETP.DECNET>(RET)

                  (ESC)
$EDIT LOCAL-NCP.CMD.4 (OUTPUT AS) PS:<UETP.LIB>LOCAL-NCP-4097.CMD(RET)
Edit: LOCAL-NCP.CMD.4
*p^:*(RET)
00100   !************************************************************************!
00200   !                                                                        !
00300   ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED        !
00400   !   OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.              !
00500   !                                                                        !
00600   !COPYRIGHT (C) 1978,1979 BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS!
00700   !                                                                        !
00800   !       DECnet-20 V2 Local NCP test                                      !
00900   !                                                                        !
01000   !                                                                        !
01100   !To run this test start OPR in a job enabled with wheel or operator     !
01200   !privileges and enter the NCP command subset:                           !
01300   !                                                                        !
01400   !       @ENABLE                                                          !
01500   !       $OPR                                                             !
01600   !       OPR>ENTER NCP                                                    !
01700   !       NCP>                                                             !
01800   !                                                                        !
01900   !Then either enter:                                                      !
02000   !                                                                        !
02100   !       NCP>TAKE LOCAL-NCP.CMD                                           !
02200   !                                                                        !
02300   !to execute the statements in the command file or                       !
02400   !type the following commands by hand.                                   !
02500   !                                                                        !
02600   !************************************************************************!
02700
```

```
02800   SET EXECUTOR host-node-name
02900   SHOW EXECUTOR
03000   SHOW STATUS LOCAL
03100   SHOW STATUS KNOWN LINE
03200   SHOW STATUS LINE line
03300   SHOW STATUS LINE line
03400   SHOW QUEUE NCP-REQUESTS
*shost-node-name$4097$2800(RET)
02800   SET EXECUTOR 4097
*sline$KDP_0_0$3200,E(RET)
03200   SHOW STATUS LINE KDP_0_0
*sline$KDP_0_1$3300,E(RET)
03300   SHOW STATUS LINE KDP_0_1
*p2800:3400(RET)
02800   SET EXECUTOR 4097
02900   SHOW EXECUTOR
03000   SHOW STATUS LOCAL
03100   SHOW STATUS KNOWN LINE
03200   SHOW STATUS LINE KDP_0_0
03300   SHOW STATUS LINE KDP_0_1
03400   SHOW QUEUE NCP-REQUESTS
*eu(RET)

[<UETP.LIB>LOCAL-NCP-4097.CMD.1]
$
```

When you have exited from EDIT,

Enter:

```
$CONN PS:<UETP.LIB>(RET)
$OPR(RET)
OPR>ENTER NCP(RET)
NCP>TAKE LOCAL-NCP-xxxx.CMD(RET)
```

where xxxx is the node name of the local host

OUTPUT FOLLOWS

Enter:

```
NCP>EXIT(RET)
```

To compare the output from your command file with an example, enable and type the file PS:<UETP.DECNET>LOCAL-NCP-KS-EXAMPLE.TXT. This example is the result of a TAKE command and includes a list of outstanding NCP requests in response to the SHOW QUEUE NCP-REQUESTS command. When you enter the commands directly, you may not receive this list because your typing speed is slower than the response speed.

The output from your .CMD file should be similar in format to the example typed. Node names and line identifications may differ. There should be no error messages. When the test completes successfully, continue with the next section.

10.3.3 Line Loopback Test

You have now tested that NCP commands function on the local node. Before you test NCP commands to a remote node, the line loopback test is performed to ensure that the line(s) from your node out to the end of the cable are functioning.

Step 1: Log in and enable.

Log in as a user with WHEEL or OPERATOR privileges enabled. This test is run under UETP. Therefore, use the directory to which you added the IPCF and USER GROUPS 100 subcommands. You should now be familiar with the UETP responses. Only the entries are given for this test. If you need to review the UETP output, refer to Step 7 in Section 10.2.

Step 2: Establish the search list.

Enter:

(ESC)
↓
$DEFINE (LOGICAL NAME) DSK: DSK:,PS:<UETP.LIB>(RET)

NOTE

The line loopback test is system dependent. If you are testing a DECSYSTEM-2040/50/60, perform Steps 3 through 10. If you are testing a DECSYSTEM-2020, go to Step 11 and perform Steps 11 through 21.

Step 3: Create a file for the DN20 node name.

Create the file PS:<UETP.LIB>DNV2LN.NOD which specifies the name of the DN20 to be tested. The DN20 to be tested must be physically connected to the system on which this test is run. The name must be entered on the first line of the file being created and the name must not have terminating colons. Exit from EDIT with the EDIT command EU to remove the line numbers. An example is given. Substitute the name of your DECnet-20 communications front end for the DN20A in the example. Because the test cannot run without the file, both a DIRECTORY and a TYPE command are entered for checking purposes.

Example:

```
$conn ps:<uetp.lib>(RET)
$edit dnv2ln.nod(RET)

%No such file type, Creating New file
Input: DNV2LN.NOD.1
00100   dn20a(RET)
00200   $
*eu(RET)

[DNV2LN.NOD.1]
$dir dnv2ln.nod(RET)

   PS:<UETP.LIB>
DNV2LN.NOD.1
$type dnv2ln.nod(RET)
dn20a
$
```

Step 4: Create the file PS:<UETP.LIB>DNV2LN.LIN

Create the file PS:<UETP.LIB>DNV2LN.LIN which specifies the DN20 lines to be tested. This file must contain a separate line of text for each hardware line to be tested. The line of text is the line specification in the form DDD_n_m where DDD represents the device type, n represents the controller number, and m represents the line number. Valid device types are KDP and DMC. (Because the DMC has only one line, the form is DDD_n.) The device DUP is valid for 32K front ends only.

An example of a DNV2LN.LIN file creation follows. You must substitute the specifications of the hardware lines in your configuration. TEST72 in the front end requires that the line specifications be in uppercase as shown. Because the test will not run without the created file, the directory is checked and the file is typed.

Example:

```
$EDIT DNV2LN.LIN<RET>

%No such file type, Creating New File
Input: DNV2LN.LIN.1
00100   KDP_0_0<RET>
00200   KDP_0_1<RET>
00300   KDP_0_2<RET>
00400   KDP_0_3<RET>
00500   DMC_0<RET>
00600   DMC_1<RET>
00700   $
*EU<RET>

[DNV2LN.LIN.1]
$dir dnv2ln.lin<RET>

  PS:<UETP.LIB>
 DNV2LN.LIN.1
$type dnv2ln.lin<RET>
KDP_0_0
KDP_0_1
KDP_0_2
KDP_0_3
DMC_0
DMC_1
$
```

Step 5: Ensure PS:<UETP.RUN> is empty.

Enter:

```
$CONNECT PS:<UETP.RUN><RET>
$DELETE *.*<RET>
```

Step 6: Run the UETP program.

Enter:

```
$RUN UETP<RET>
UETP>
```

Step 7: Enable test and begin testing.

Enter:

```
UETP>ENABLE DNV2LN/DEPTH:COMPREHENSIVE(RET)
UETP>BEGIN(RET)
```

Step 8: Respond to OPR messages.

For each line to be tested, TWO messages are output to the operator's console. You will enter some commands and follow some given directions. Examples of the two messages follow. Not all numbers and names in the examples will duplicate what you see. Job name (X in the example) and line specification will reflect your testing parameters. The DN20A in the example should be replaced by the name of your DECnet-20 communications front end. As each line is tested, the two messages will be repeated until all lines to be tested have been addressed.

Example - first message:

```
OPR>
11:25:47  <3>   Batch-stream 0 for JOB #11   --Message from Batch User--
          Job X Req #17 for MIERSWA
          PLEASE TURN LINE OFF, CONNECT PLUG, TURN LINE ON: KDP_0_1

OPR>ENTER NCP(RET)
NCP>SET EXECUTOR DN20A(RET)
NCP>SET STATE LINE KDP_0_1 OFF(RET)
NCP>
11:26:27     NCP request # 2 [SET STATUS LINE KDP _ 0 _ 1 OFF]
             Function completed successfully


      *** ATTACH LOOPBACK CONNECTOR NOW ***

      *** RETURN TO CONSOLE AND CONTINUE ***


NCP>SET STATE LINE KDP_0_1 ON(RET)
NCP>
11:30:23     NCP request # 3 [SET STATUS LINE KDP _ 0 _ 1 ON]
             Function completed successfully
NCP>RETURN(RET)
OPR>RESPOND 3 OK(RET)
OPR>
             (Test takes approximately 2 minutes)
```

Example - second message:

```
OPR>
11:32:46  <4>   Batch-stream 0 JOB # 11 --Message from Batch
User--
                Job X Req # 17 for MIERSWA
                PLEASE TURN LINE OFF: KDP_0_1
OPR>ENTER NCP(RET)
NCP>SET EXECUTOR DN20A(RET)
NCP>SET STATE LINE KDP_0_1 OFF(RET)
NCP>
11:33:07     NCP request # 4 [SET STATUS KDP _ 0 _ 1 OFF]
             Function completed successfully
NCP>RETURN(RET)
OPR>RESPOND 4 OK(RET)
OPR>
```

Step 9: When the test ends, check the RUN.LOG file.

Enter:

```
$TYPE RUN.LOG(RET)
```

If the RUN.LOG file confirms the successful completion of the test (each of the lines you tested), you have completed this test. Go to Section 10.3.4. If there are errors, continue with the next step.

Step 10: Check the DNV2LN.LOG file.

Enter:

```
$TYPE DNV2LN.LOG(RET)
```

Examine the output to determine the cause of the error. If any of the lines were successfully tested, repeat from Step 4 but include only the line specifications for lines that failed. If the test shows no lines tested successfully, repeat all steps beginning with the step referenced as in error in DNV2LN.LOG.

Step 11: Turn lines off.

The DECSYSTEM-2020 can be configured with 1 or 2 KDP DECnet communication lines. This test is to be run once for each line in your configuration.

Use the NCP command subset of OPR to turn all lines off. For example, if your configuration includes 2 KDPs,

Enter:

```
$OPR(RET)
OPR>ENTER NCP(RET)
NCP>SET STATE LINE KDP_0_0 OFF(RET)
NCP>SET STATE LINE KDP_0_1 OFF(RET)
NCP>EXIT(RET)
$
```

Step 12: Install loopback connector.

Install a loopback connector on line to be tested. See Section 10.3.1 for description of loopback connector procedure.

Step 13: Enable loopback and start line for cable loopback.

Enter:

```
$OPR(RET)
OPR>ENTER NCP(RET)
NCP>SET LOCAL LOOPBACK ENABLED KDP_0_0(RET)
NCP>SET STATE LINE KDP_0_0 CABLE-LOOPBACK(RET)
NCP>EXIT(RET)
```

Step 14: Ensure that PS:<UETP.RUN> is empty.

Enter:

```
$CONNECT PS:<UETP.RUN>(RET)
$DELETE *.*(RET)
```

Step 15: Run the UETP program.

Enter:

```
$RUN UETP(RET)
UETP>
```

Step 16: Enable tests and begin testing.

Enter:

```
UETP>ENABLE TEST72/DEPTH:COMPREHENSIVE(RET)
UETP>ENABLE DNV2LL/DEPTH:COMPREHENSIVE(RET)
UETP>BEGIN(RET)
```

Step 17: Examine the output from UETP.

Only the DNV2LL test will end. TEST72 will not end by itself. If the STATUS reads Ended and the ERROR COUNT reads 0, the test was successful. If 1 or more errors are shown, continue to the end of the procedure, then repeat the test. If the test was successful, continue.

Step 18: Abort TEST72 and exit.

Enter:

```
UETP>ABORT TEST72/DEPTH:COMPREHENSIVE(RET)
UETP>EXIT(RET)
$
```

Step 19: Check the RUN.LOG file.

Enter:

```
$TYPE RUN.LOG(RET)
```

If the RUN.LOG file confirms the successful completion of the test, go to Step 21. If there are errors, continue with the next step.

Step 20: Check the DNV2LL.LOG file.

Enter:

```
$TYPE DNV2LL.LOG(RET)
```

Examine the output to determine the cause of the error. Do not go to the next step. Repeat the test from Step 1, or, if it is clear from DNV2LL.LOG that the tests were enabled and begun, repeat from Step 14.

Step 21: Disable loopback testing for the first line.

Enter:

```
$OPR(RET)
OPR>ENTER NCP(RET)
NCP>SET STATE LINE KDP_0_0 OFF(RET)
NCP>SET LOCAL LOOPBACK DISABLED KDP_0_0(RET)
NCP>EXIT(RET)
$
```

You have completed the line loopback testing for 1 line. If you have 2 lines, repeat all from Step 12 (Install Loopback Connector).

10.3.4 **Remote-NCP Test**

The Remote-NCP test is executed in much the same way as the Local-NCP test. You use the OPR/NCP/ORION interface and may enter commands directly to your terminal or indirectly with a command file.

In the previous two sections you have demonstrated that the Network Control Program functions on the local node (Section 10.3.2), and that the lines leading out of the local node are functioning (Section 10.3.3). The purpose of the Remote-NCP test is to demonstrate that the OPR/NCP/ORION interface can be used to obtain information from the device controllers on remote nodes.

Ensure that all loopback connectors are removed and the system is connected normally.

If your host node is a DECSYSTEM-2040/50/60, reload the DECnet-20 front end and log in with WHEEL or OPERATOR privileges enabled. Go to Step 1A if you wish to use direct input to your terminal; go to Step 1B if you wish to use a command file.

If your host node is a DECSYSTEM-2020, reload the system and log in with WHEEL or OPERATOR privileges enabled. Go to Step 2A if you wish to use direct input to your terminal; go to Step 2B if you wish to use a command file.

Step 1A. Remote-NCP test using direct terminal input (DECSYSTEM-2040/50/60)

Enter:

```
(ESC)
  ↓
@enaBLE(RET)
$conn ps:<uetp.decnet>(RET)
$type remote-ncp-dn20-example.cmd(RET)
```

Read the example file. Refer to the commands below the second line of asterisks. The "DN20" in the example SET EXECUTOR command is the node name of the DECnet-20 communications front end. You type the node name you assigned to the communications front end in the NETGEN procedures. For the SHOW STATUS LINE and SHOW COUNTS LINE commands, you type the line identification of each line in your configuration that you wish to test.

When you have executed the commands following the directions in the example, compare the output with the example below. The format should be similar. The node name and line identifications may differ.

The TAKE commands that precede the NCP commands in the example are given to suppress printing of system and OPR messages during output. These commands are helpful if you are testing under timesharing, but are not needed if you are testing standalone.

Example entry and responses:

```
$take refuse(RET)
OPR>take opr.cmd(RET)
OPR>
08:56:18        --OUTPUT DISPLAY for OPR Modified--
OPR>enter ncp(RET)
NCP>set executor dn20a(RET)
NCP>show executor(RET)
NCP>
08:56:51                Current EXECUTOR is node DN20A
NCP>show status local(RET)
NCP>
08:57:06          NCP request # 6 [SHOW STATUS LOCAL]

        Status of local node as of 6-Aug-79 08:57:06

        Node Name is DN20A, # 0, SYSTEM = REL 4 FIELDTEST 3.0 18-JUL-79
        Routine version = 3.1.0, Communications Version = 3.1.0
        State is On, Default Host = 2102

                Function completed successfully

NCP>show status known lines(RET)
NCP>
08:57:40        NCP request # 7 [SHOW STATUS KNOWN LINES]

                Status as of 6-Aug-79 08:57:40

                Line ID         State           Adjacent Node

                DTE20_1_0_0     On              2102
                DMC11_0_0_0     Off
                DMC11_1_0_0     On
                DMC11_2_0_0     On
                KDP_0_0_0       On              SYS880
                KDP_0_1_0       On
                KDP_0_2_0       On              DN200
                KDP_0_3_0       Off

                Function completed successfully
```

```
NCP>show status line dmc11_0_0(RET)
NCP>
08:58:42          NCP request # 8 [SHOW STATUS LINE DMC11 _ 0 _ 0]

                  Status as of 6-Aug-78 08:58:41

                  Line ID          State          Adjacent Node

                  DMC11_0_0_0      Off

                  Function completed successfully

NCP>show status line kdp_0_0(RET)
NCP>
08:59:18          NCP request # 9 [SHOW STATUS LINE KDP _ 0 _ 0]

                  Status as of 6-Aug-79 08:59:18

                  Line ID          State          Adjacent Node

                  KDP_0_0_0        On             SYS880

                  Function completed successfully

NCP>show counts line dmc11_0_0(RET)
NCP>
08:59:52          NCP request # 10 [SHOW COUNTS LINE DMC11 _ 0 _ 0]
        Counts for line DMC11_0_0_0, as of 6-Aug-79 08:59:52
        Seconds since zeroed       4296
        Blocks received               0
        Blocks sent                   0
        Retran, line errors           0
        Received line errors          0
        Retran, not line errors       0
        Receive timeouts              0

                  Function completed successfully

NCP>show counts line kdp_0_0(RET)
NCP>
09:00:29          NCP request # 11 [SHOW COUNTS LINE KDP _ 0 _ 0]

        Counts for line KDP_0_0_0, as of 6-Aug-79 09:00:29
        Seconds since zeroed       4334
        Blocks received               1
        Blocks sent                   1
        Retran, line errors           0
        Received line errors          0
        Retran, not line errors       0
        Receive timeouts              0

                  Function completed successfully

NCP>show queue ncp-requests(RET)
NCP>
09:01:04

                  There are no outstanding NCP requests.

NCP>exit(RET)
 End of REFUSE.CMD.1
$
```

If you wish to see an example of the same commands entered with a command file, type the file PS:<UETP.DECNET>REMOTE-NCP-DN20-EXAMPLE.TXT. You will see one major difference between direct and indirect entry. When commands are given using a command file, there will be a response to the SHOW QUEUE NCP-REQUESTS command. Because direct entry is at typing speed, the responses follow the commands and there are no commands waiting in the queue.

If you wish to repeat the REMOTE-NCP test using a command file, continue with Step 1B; otherwise go to Section 10.3.5.

Step 1B. Remote-NCP test using a command file (DECSYSTEM-2040/50/60)

The file REMOTE-NCP.CMD in the directory PS:<UETP.DECNET> contains the commands for the REMOTE-NCP test. One SET EXECUTOR command, two SHOW STATUS LINE commands, and two SHOW COUNTS LINE commands are given with dummy arguments. These arguments must be edited to reflect the node name and line identifications of YOUR configuration. The number of both SHOW STATUS LINE and SHOW COUNTS LINE commands must equal the number of lines you plan to use for network traffic. The node name in the SET EXECUTOR command must be identical to the node name assigned to the DECnet-20 communications front end in the NETGEN procedures.

The edited file should be output to PS:<UETP.LIB> as shown in the example. Use the EDIT command EU to remove line numbers when you exit from EDIT. Including the DN20 node name in the output file will help identify it for future use.

Example entries and responses:

```
$edit remote-ncp.cmd.3 (ESC)(OUTPUT AS) ps:<uetp.lib>remote-ncp-dn20a.cmd(RET)
Edit: REMOTE-NCP.CMD.3
*p^:*(RET)
00100   !*****************************************************************************!
00200   !                                                                             !
00300   ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED             !
00400   !   OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.                   !
00500   !                                                                             !
00600   !COPYRIGHT (C) 1978,1979 BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS!
00700   !                                                                             !
00800   !       DECnet-20 V2 Remote NCP test                                          !
00900   !                                                                             !
01000   !                                                                             !
01100   !To run this test start OPR in a job enabled with wheel or operator          !
01200   !privileges and enter the NCP command subset:                                 !
01300   !                                                                             !
01400   !       @ENABLE                                                               !
01500   !       $OPR                                                                  !
01600   !       OPR>ENTER NCP                                                         !
01700   !       NCP>                                                                  !
01800   !                                                                             !
01900   !Then either enter:                                                           !
02000   !                                                                             !
02100   !       NCP>TAKE REMOTE-NCP.CMD                                               !
02200   !                                                                             !
02300   !to execute the statements in the command file or                            !
02400   !type the following commands by hand.                                         !
02500   !                                                                             !
02600   !*****************************************************************************!
02700
02800   SET EXECUTOR node-name
02900   SHOW EXECUTOR
03000   SHOW STATUS LOCAL
03100   SHOW STATUS KNOWN LINE
03200   SHOW STATUS LINE line
03300   SHOW STATUS LINE line
03400   SHOW COUNTS LINE line
03500   SHOW COUNTS LINE line
03600   SHOW QUEUE NCP-REQUESTS
*snode-name$DN20A$2800(RET)
02800   SET EXECUTOR DN20A
*sline$DMC11 0 0$3200,E(RET)
03200   SHOW STATUS LINE DMC11_0_0
*sline$KDP 0 0$3300,E(RET)
03300   SHOW STATUS LINE KDP_0_0
*sline$DMC11 0 0$3400,E(RET)
03400   SHOW COUNTS LINE DMC11_0_0
*sline$KDP 0 0$3500,E(RET)
03500   SHOW COUNTS LINE KDP_0_0
*p2800:3600(RET)
02800   SET EXECUTOR DN20A
02900   SHOW EXECUTOR
03000   SHOW STATUS LOCAL
03100   SHOW STATUS KNOWN LINE
03200   SHOW STATUS LINE DMC11_0_0
03300   SHOW STATUS LINE KDP_0_0
03400   SHOW COUNTS LINE DMC11_0_0
03500   SHOW COUNTS LINE KDP_0_0
03600   SHOW QUEUE NCP-REQUESTS
*eu(RET)

[REMOTE-NCP-DN20A.CMD.1]
```

When you have exited from EDIT,

Enter:

```
CONN PS:<UETP.LIB>(RET)
$OPR(RET)
OPR>ENTER NCP(RET)
NCP>TAKE REMOTE-NCP-xxxx.CMD(RET)
```

where xxxx is the node name of the DN20

OUTPUT FOLLOWS

Enter:

```
NCP>EXIT(RET)
$
```

Check your output as the test is being performed. No error messages should be printed. The output should have the same format as the file REMOTE-NCP-DN20-EXAMPLE.TXT in PS:<UETP.DECNET>. This example is for a DECSYSTEM-2060 node with a DN20 node name of DN20A. Type the example file to compare it with your output.

When the test completes successfully, go to Section 10.3.5.

Step 2A. Remote-NCP test using direct terminal input (DECSYSTEM-2020)

Enter:

```
(ESC)
@enaBLE (CAPABILTIES)(RET)
$type ps:<uetp.decnet>remote-ncp-20-example.cmd(RET)
```

Read the example file. Refer to the nine commands below the second line of asterisks. To test YOUR local NCP, certain changes are required:

The first command will be:

SET EXECUTOR your local host name

The node name in the SET EXECUTOR command must agree with the node name in the 4-CONFIG.CMD file in PS:<SYSTEM>.

For both the SHOW STATUS LINE and SHOW COUNTS LINE commands, type the line identification(s) of the line(s) in your configuration that you wish to test. For one line, the lineid will be KDP_0_0. For two lines, the lineids will be KDP_0_0 and KDP_0_1.

Now, type the commands (seven or nine commands depending on the number of lines) shown in the example, making the changes indicated for node name and lineids. Check the output as the commands are being performed. There should be no error messages. When the response to the last command is complete, type EXIT following the NCP prompt.

To compare your output with an example, type the file PS:<UETP.DECNET>REMOTE-NCP-20-EXAMPLE.TXT. This example is the result of a TAKE command. If you wish to repeat the LOCAL-NCP test using a command file, go to Step 2B; otherwise go to Section 10.3.5.

Step 2B: Remote-NCP test using a command file (DECSYSTEM-2020)

The file REMOTE-NCP.CMD in the directory PS:<UETP.DECNET> contains the commands for the REMOTE-NCP test. One SET EXECUTOR command, two SHOW STATUS LINE commands, and two SHOW COUNTS LINE commands are given with dummy arguments. These arguments must be edited to reflect the node name and line identifications of YOUR configuration. The number of both SHOW STATUS LINE and SHOW COUNTS LINE commands must equal the number of lines you plan to use for network traffic. The node name in the SET EXECUTOR command must be identical to the node name specified in the 4-CONFIG.CMD file in the installation procedures.

The edited file should be output to PS:<UETP.LIB> as shown in the example. Use the EDIT command EU to remove line numbers when you exit from EDIT.

Example entries and responses:

```
                    (ESC)
$edit remote-ncp.cmd.2 (OUTPUT AS) ps:<uetp.lib>remote-ncp.cmd(RET)
Edit: REMOTE-NCP.CMD.2
*p^:*(RET)
00100   !*************************************************************************!
00200   !                                                                         !
00300   ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED         !
00400   !   OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.               !
00500   !                                                                         !
00600   !COPYRIGHT (C) 1978,1979 BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS!
00700   !                                                                         !
00800   !       DECnet-20 V2 Remote NCP test                                      !
00900   !                                                                         !
01000   !                                                                         !
01100   !To run this test start OPR in a job enabled with wheel or operator      !
01200   !privileges and enter the NCP command subset:                            !
01300   !                                                                         !
01400   !       @ENABLE                                                           !
01500   !       $OPR                                                              !
01600   !       OPR>ENTER NCP                                                     !
01700   !       NCP>                                                              !
01800   !                                                                         !
01900   !Then either enter:                                                       !
02000   !                                                                         !
02100   !       NCP>TAKE REMOTE-NCP.CMD                                           !
02200   !                                                                         !
02300   !to execute the statements in the command file or                        !
02400   !type the following commands by hand.                                    !
02500   !                                                                         !
02600   !*************************************************************************!
02700
```

```
02800   SET EXECUTOR node-name
02900   SHOW EXECUTOR
03000   SHOW STATUS LOCAL
03100   SHOW STATUS KNOWN LINE
03200   SHOW STATUS LINE line
03300   SHOW STATUS LINE line
03400   SHOW COUNTS LINE line
03500   SHOW COUNTS LINE line
03600   SHOW QUEUE NCP-REQUESTS
*snode-name$4097$2800(RET)
02800   SET EXECUTOR 4097
*sline$KDP_0_0$3200,E(RET)
03200   SHOW STATUS LINE KDP_0_0
*sline$KDP_0_1$3300,E(RET)
03300   SHOW STATUS LINE KDP_0_1
*sline$KDP_0_0$3400,E(RET)
03400   SHOW COUNTS LINE KDP_0_0
*sline$KDP_0_1$3500,E(RET)
03500   SHOW COUNTS LINE KDP_0_1
*p2800:3600(RET)
2800    SET EXECUTOR 4097
2900    SHOW EXECUTOR
3000    SHOW STATUS LOCAL
3100    SHOW STATUS KNOWN LINE
3200    SHOW STATUS LINE KDP_0_0
3300    SHOW STATUS LINE KDP_0_1
3400    SHOW COUNTS LINE KDP_0_0
3500    SHOW COUNTS LINE KDP_0_1
3600    SHOW QUEUE NCP-REQUESTS
*eu(RET)

[<UETP.LIB>REMOTE-NCP.CMD.1]
$
```

When you have exited from EDIT,

Enter:

```
$CONN PS:<UETP.LIB>(RET)
$OPR(RET)
OPR>ENTER NCP(RET)
NCP>TAKE REMOTE-NCP.CMD(RET)

     OUTPUT FOLLOWS
```

Enter:

```
NCP>EXIT(RET)
$
```

Check your output as the test is being performed. No error messages should be printed. The output should have the same format as the file REMOTE-NCP-20-EXAMPLE.CMD in PS:<UETP.DECNET>. This example is for a DECSYSTEM-2020. Type the example file to compare it with your output.

When the test completes successfully, go to Section 10.3.5.

10.3.5 Remote and Local Node Network File Transfer Test

This test ensures that the Network File Transfer (NFT) utility is functioning. The test is run under the control of the UETP program.

Step 1: Log in as a nonprivileged user.

NOTE

If you are logging in to a directory other than the one you used for the verification procedures, remember to add the IPCF and USER GROUPS 100 parameters to the login directory. See Section 10.2 for help.

Step 2: Create an NFT.INIT file in PS:<UETP.LIB>.

The following example is for your guidance only. The parameters you use in the NFT command SET DEFAULTS must describe each node to which transfers will be made. You must have access to at least one account on each node to be tested. USER, ACCOUNT, and PASSWORD must have the values and format agreed upon by the remote node when the account was opened.

The example assumes your file transfers will be between two nodes: 2102, a DECSYSTEM-2060, and 4097, a DECSYSTEM-2020.

Example:

```
@conn ps:<uetp.lib>(RET)
@edit nft.init(RET)

%File not found, Creating New file
Input: NFT.INIT.1
00100   set defaults 2102::/user:gunn/account:341/ostype:tops20/pass:zlinfm(RET)
00200   set defaults 4097::/user:gunn/account:341/ostype:tops20/pass:zlinfm(RET)
00300   $
*eu(RET)

[NFT.INIT.1]

@type ps:<uetp.lib>nft.init(RET)
set defaults 2102::/user:gunn/account:341/ostype:tops20/pass:zlinfm
set defaults 4097::/user:gunn/account:341/ostype:tops20/pass:zlinfm
@
```

See Chapter 8 if you need additional help in creating the NFT.INIT file.

Step 3: Check availability of nodes to be tested.

Entry and response:

```
@info decnet (RET)
 Accessible DECNET nodes are:  1031, 2102,  4097,  DN200,  DN20A,
DN20L, SYS880
```

NOTE

If the NFT test does not complete successfully the first time, it may be necessary to return to this step. (A node may have lost its "accessible" state.)

Step 4: Define the file search association of DSK:.

Entries and responses:

```
@define utp:  ps:<uetp.lib>(RET)
@define dsk:  dsk:,utp:(RET)
```

Step 5: Connect to PS:<UETP.RUN> and ensure that this directory is empty.

Entries and responses:

```
@connect ps:<uetp.run>(RET)
@delete *.*(RET)
 DNV2FT.LOG.1 [OK]
 EXCEPT.LOG.1 [OK]
 RUN.LOG.1 [OK]
 SEND.DAT.1 [OK]
@expunge <uetp.run>(RET)
 PS:<UETP.RUN> [13 pages freed]
```

Step 6: Run UETP.

Entries and responses:

```
@run uetp(RET)
[27-Sep-79 12:05:16    User Environment Test Package ]


UETP>enable dnv2ft/depth:application(RET)
[12:06:01  ENABLE COMPLETED]
UETP>begin(RET)
[12:06:09  BEGIN COMPLETED]
UETP>
    START DNV2FT   12:06:35
status

[27-Sep-79 12:09:06]

   Test   Depth   Status    Cycle   Times   Error   Error   Start
   name                             run     count   limit   time
   =====  ====    ======    ====    ====    =====   =====   ====
   DNV2FT APP     Running        1       0       0       0   27-Sep-79 12:06:09

     END  DNV2FT  12:12:07

[All tests complete on processor # 2102]

[27-Sep-79 12:13:25]

   Test   Depth   Status    Cycle   Times   Error   Error   Start
   name                             run     count   limit   time
   =====  ====    ======    =====   =====   =====   =====   =====
   DNV2FT APP     Ended          1       1       0       0   27-Sep-79 12:06:09


UEPT>exit(RET)
@dir(RET)

  PS:<UETP.RUN>
DNV2FT.LOG.1
EXCEPT.LOG.1
RUN.LOG.1

TOTAL OF 3 FILES
@type run.log(RET)
   START DNV2FT   27-Sep-79 12:06:35   INIT  0:00:025:25:51DNV2FT started
   MINOR DNV2FT   27-Sep-79 12:06:51   INIT  0:00:059:52:01DNV2FT finished step INIT
   MINOR DNV2FT   27-Sep-79 12:07:03   STP1  0:00:0813:07:19DNV2FT finished step STP1
   MINOR DNV2FT   27-Sep-79 12:08:57   STP2  0:00:1544:55:22DNV2FT finished step STP2
   MINOR DNV2FT   27-Sep-79 12:09:42   STP3  0:00:2157:25:28DNV2FT finished step STP3
   MINOR DNV2FT   27-Sep-79 12:10:22   STP1  0:00:2368:30:31DNV2FT finished step STP1
   MINOR DNV2FT   27-Sep-79 12:11:51   STP2  0:00:3193:04:20DNV2FT finished step STP2
   MINOR DNV2FT   27-Sep-79 12:12:05   STP3  0:00:3697:01:03DNV2FT finished step STP3
   END   DNV2FT   27-Sep-79 12:12:07   FIN   0:00:3797:39:22DNV2FT ended

@
```

If the test ends with 0 errors, typing the RUN.LOG file displays for each node tested:

```
finished STP1
finished STP2
finished STP3
```

In the example, two nodes were tested. The six lines confirm the successful completion of the test. If the error count is 1 or more, type the DNV2FT.LOG file. All steps and errors are displayed in detail. This file will also be helpful if you plan to create your own NFT test. If you could not run UETP, check your directory for the IPCF and USER GROUP 100 entries and check your define statements.

10.3.6 DN200 Tests (2040/50/60 with Remote Job Entry Station Only)

This test demonstrates the use of Operator Commands (OPR) at the DN200 remote station console. The test includes queueing output to be printed by the printer at the remote station and executing a batch job from the remote station. The batch job includes reading the card deck, executing the batch program, and printing the batch log.

Prior to this test, the DN200 must have been configured as part of the local DECnet-20 system. The DN200 must have been verified, powered on and loaded.

Step 1: Check the DN200 status and queue status at operator's console.

Entries and Responses:

```
2102::OPR>SHOW STATUS(RET)
2102::OPR>
2102::
9:13:05                 -- System Device Status --

Printer Status:
  Unit     Node         Status
  ----  ----------- ---------------
     0       DN200  Idle

Reader Status:

  Unit     Node         Status
  ----  ----------  ---------------
     0       DN200  Idle

2102::OPR>SHOW Q(RET)
2102::OPR>
2102::
9:13:40                 --The Queues are Empty--

2102::OPR>
```

Step 2: Print file on local and remote (DN200) printers and compare. (First ensure the DN200 is available.)

Entries and responses:

```
@inFORMATION (ABOUT) decNET<RET>
 Accessible DECNET nodes are: 1031, 2102, 4097, DN200, DN20A, DN20L
@print ps:<uetp.lib>pictur.txt<RET>
[Job PICTUR Queued, Request-ID 89, Limit 54]
@print ps:<uetp.lib>pictur.txt/destination-node:dn200::<RET>
[Job PICTUR Queued, Request-ID 90, Limit 54]

@iNFORMATION (ABOUT) oUTPUT-REQUESTS<RET>

Printer Queue:
 Job Name    Req#  Limit            User
 --------    ----  -----  ------------------------
 * PICTUR      90     54  CIRINO                      On Unit:0  /Dest:DN200
    Started at 10:55:21, printed 0 of 54 pages
   6201DP        6    810  HARAMUNDANIS                /Lower  /Forms:NARROW
   DN6A          2    108  LNEFF                       /Lower  /Forms:NARROW
 There are 3 Jobs in the Queue (1 in Progress)
```

The output is a geometric pattern. Any errors in transmission will be easily seen.

Step 3: Prepare a card deck for submission to the DN200 card reader.

Keypunch entries:

```
$JOB username
$PASSWORD password
$TOPS20
@DEFINE DSK: DSK:,<UETP.RUN>,<UETP.LIB>
@PRINT CHARS.TXT
$EOJ
```

Step 4: Read the deck on the DN200 card reader.

Output on DN200 console:

```
2102::OPR>SHOW STATUS (RET)
2102::OPR>
2102::
 9:27:13                -- System Device Status --

Printer Status:
  Unit     Node          Status
  ----  ----------  ---------------
     0       DN200  Idle

Reader Status:
  Unit     Node          Status
  ----  ----------  ---------------
     0       DN200  Idle


2102::OPR>
2102::
 9:29:34        Reader 0 [DN200]  -- 1 Job: 7 Cards Spooled --

2102::
 9:29:37        Batch-Stream 2  --Begin--
                Job JB3047 Req #38 for CIRINO

2102::OPR>
2102::
 9:29:49        Batch-Stream 2  --End--
                Job JB3047 Req #38 for CIRINO

2102::
 9:29:51        Printer 0 [DN200]  --Started--

2102::
 9:29:51        Printer 0 [DN200]  --Begin--
                Job CHARS Req #39 for CIRINO

2102::OPR>
2102::
 9:31:54        Printer 0 [DN200]  --End--
                Job CHARS Req #39 for CIRINO

2102::
 9:31:54        Printer 0 [DN200]  --Begin--
                Job JB3047 Req #40 for CIRINO

2102::OPR>
2102::
 9:32:56        Printer 0 [DN200]  --End--
                Job JB3047 Req #40 for CIRINO

2102::OPR>
```

The SHOW STATUS command is given to check the immediate availability of the reader and printer. All of the output that follows is displayed without further action by the operator. Req #38 is for the read; Req #39 is for the printing of the CHARS.TXT file; Req #40 is for the printing of the log file.

Output on the DN200 printer is a five-page printout of the complete character set extending margin to margin with each successive line offset one space to the left. The printout of the batch log file completes the job. The log file printout should resemble the following example:

Example:

```
 9:29:35 STDAT  SPRINT version 104(4072) 2102 Development System,
                TOPS-20 Monitor 4(3117)
 9:29:35 STMSG  [Job Input from Reader 0 [DN200]]
 9:29:35 STCRD  $JOB CIRINO
 9:29:35 STCRD  $PASSWORD
 9:29:35 STCRD  $TOPS20
 9:29:35 STCRD  $EOJ
 9:29:35 STSUM  End of job encountered
 9:29:35 STSUM  7 Cards Read
 9:29:35 STSUM  Batch job submitted


                        26-Sep-79  9:29:37

 BATCON Version  104(4126)                        GLXLIB Version  1(515)

           Job JB3047 Reg #38 for CIRINO in Stream 2

        OUTPUT:  Log                              TIME-LIMIT: 0:05:00
        UNIQUE:  Yes                              BATCH-LOG:  Append
        RESTART: No                               ASSISTANCE: Yes
                                                  SEQUENCE:   1624

        Input from => PS:<CIRINO>JB304.CTL.1
        Output to  => PS:<CIRINO>JB3047.LOG.1


 9:29:38 USER    2102 Development System, TOPS-20 Monitor 4(3117)
 9:29:38 MONTR  @SET TIME-LIMIT 300
 9:29:38 MONTR  @LOGIN CIRINO 341
 9:29:42 MONTR   Job 50 on TTY222 26-Sep-79 09:29:42
 9:29:42 MONTR  @
 9:29:42 MONTR  [PS Mounted]
 9:29:42 MONTR
 9:29:42 MONTR  [CONNECTED TO PS:<CIRINO>]
 9:29:42 MONTR  @DEFINE DSK: DSK:,<UETP.LIB>
 9:29:44 MONTR  @@CONN <UETP.RUN>
 9:29:44 MONTR  @@PRINT CHARS.TXT
 9:29:46 MONTR  [Job CHARS Queued, Request-ID 39, Limit 54]
 9:29:47 MONTR  @
 9:29:47 BLABL  %FIN::
 9:29:47 MONTR
 9:29:48 MONTR  Killed Job 50, User CIRINO, Account 341, TTY 222,
 9:29:48 MONTR    at 26-Sep-79 09:29:47, Used 0:00:00 in 0:00:05
```

10.3.7 DECnet Error and Event Logging Test

This test demonstrates the error and event logging features of DECnet. While you were performing the previous tests, DECnet was writing entries into the system log. SYSERR can be used to examine these entries. Refer to Appendix C for a detailed description of SYSERR entries.

The following commands output to the printer a summary of all entries concerning your network. The /BRIEF switch creates a listing containing the sequence number, the time of the occurrence, and a maximum of three lines of the most usable information for each entry. Output may be directed to your terminal by typing TTY:= in place of LPT:=.

Enter:

```
@SYSERR(RET)
*LPT:=/BRIEF/NETALL(RET)
```

Examine the output. If you need further help in interpreting the entries, the following sources will be helpful:

- Appendix C of this manual
- TOPS-10 and TOPS-20 SYSERR Manual
- HELP file for SYSERR

To obtain a detailed listing of the entries summarized in the first listing,

Enter:

```
@SYSERR(RET)
*LPT:=/DETAIL/NETALL(RET)
```

Examples of SYSERR output may be printed from PS:<UETP.DECNET>. The example files are BRIEF.TXT and DETAIL.TXT.

The HELP file for SYSERR contains information on switches for use with SYSERR. For example, you can request listings for specific sequence numbers or specific dates.

PART VI

APPENDIXES

APPENDIX A

DISCONNECT OR REJECT CODES

The disconnect or reject codes in Table A-1 are defined by NSP and are sent and retrieved by network tasks via the network functions of the MTOPR monitor call.

Table A-1
Disconnect or Reject Codes

Symbol	Value	Meaning
.DCX0	0	No special error
.DCX1	1	Resource allocation failure
.DCX2	2	Destination node does not exist
.DCX3	3	Node shutting down
.DCX4	4	Destination process does not exist
.DCX5	5	Invalid name field
.DCX6	6	Destination process queue overflow
.DCX7	7	Unspecified error
.DCX8	8	Third party aborted the logical link
.DCX9	9	User abort (asynchronous disconnect)
.DCX11	11	Undefined error code
.DCX21	21	Connect initiate (CI) with illegal destination address
.DCX24	24	Flow control violation
.DCX32	32	Too many connections to node
.DCX33	33	Too many connections to destination process
.DCX34	34	Access not permitted
.DCX35	35	Logical link services mismatch
.DCX36	36	Invalid account
.DCX37	37	Segment size too small
.DCX38	38	Process aborted
.DCX39	39	No path to destination node
.DCX40	40	Link aborted due to data loss
.DCX41	41	Destination logical link address does not exist
.DCX42	42	Confirmation of disconnect initiate (DI)
.DCX43	43	Image data field too long

APPENDIX B

DECnet OBJECT TYPES

The object types listed in Table B-1 are taken from the Network Services Protocol, Version 3.1 documentation. Object type codes are expressed in decimal. DECnet-20 will, in addition, recognize a number of object names in place of object types. Object names that are currently supported are shown in Table B-1.

Object type 0 (TASK) can only be used by a source task in order to address a target task. Object types 1 through 127 can be used by any system task; however, the task must have WHEEL or OPERATOR privileges enabled. Object types 128 through 255 are available to all network tasks.

Table B-1
DECnet Object Types

Object Type Code	Object Name	Function
0	TASK	General task, user process
1		File Access (FAL/DAP-Version 1)
2		Unit Record Services
3-4		Reserved for DECnet use
5		RSX-11M Task Control-Version 1
6		Reserved for DECnet use
7	NRM	Node Resource Manager
8-14		Reserved for DECnet use
15		RSX-11M Task Control-Version 2
16		System TALK Utility
17	FAL	File Access (FAL/DAP-Version 4)
18		RSX-11S Remote Task Loader
19	NCU	NICE process
20-62		Reserved for DECnet use
63		DECnet Test Tool (DTR)
64-127		Reserved for DECnet control
128-255		Reserved for customer extensions

APPENDIX C

NETWORK EVENT AND ERROR LOGGING

C.1 OVERVIEW

Error and event logging as described in this appendix is available only for DECSYSTEMs-2040/50/60 with a minimum of 128K words in the DN20 DECnet communications front end. (Significant events on a DECSYSTEM-2020 DECnet system are displayed as BUGCHKs or BUGINFs on the operator's console and are logged in the SYSERR file.)

One of the functions performed by the network control task (NETCON) is to record pertinent network events as SYSERR entries to the system's ERROR.SYS file. The following events are recorded each time they occur:

- Load and start NETCON
- Load a node
- Dump a node

SYSERR entries are also created for:

- hardware detected failures
- CHK11 diagnostic messages

In addition, you can optionally request NETCON to periodically log line-counter values and error statistics for any communications line terminating on any node in the network. See the Operator Commands in Section 6.3 for a description of the following commands:

INITIATE LOGGING LINE-COUNTERS lineid nodename

TERMINATE LOGGING LINE-COUNTERS lineid nodename

SET LOGGING-INTERVAL LINE-COUNTERS n (MINUTES)

Note that this feature is a DECnet-20 function; all SYSERR entries, regardless of which line or node they are recorded for, are entered into the ERROR.SYS file residing at the DECnet-20 node that processes the INITIATE LOGGING command.

To obtain a formatted printout of all or selected portions of the ERROR.SYS file, use the SYSERR program described in the TOPS-10 and TOPS-20 SYSERR Manual.

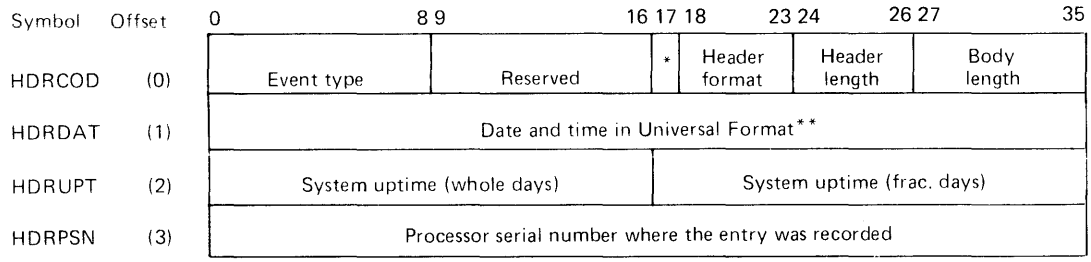
C.2 SYSERR ENTRIES

Each of the SYSERR entries described below consists of two parts:

1. A 4-word header section
2. A variable-length body section

The format of the header section is the same for all entries; the format of the body section varies by the type of event being logged.

The header section is comprised of four words that provide general information about the system, the time and type of event, and the length of the entry. The format and contents are shown in Figure C-1.



*This entry recorded by TOPS-20

** A 36-bit quantity with a binary point between the left and right halves. The left half is the number of days since 17 November 1858; the right half is the fraction of a day in Greenwich Meridian Time.

MR-S-603-80

Figure C-1 Common Header Section for SYSERR Entry

C.2.1 NETCON Load Entry (event 201)

Whenever NETCON is loaded and started, a SYSERR entry is made to record the version number and the node on which NETCON is running. The format and contents are shown in Figure C-2.

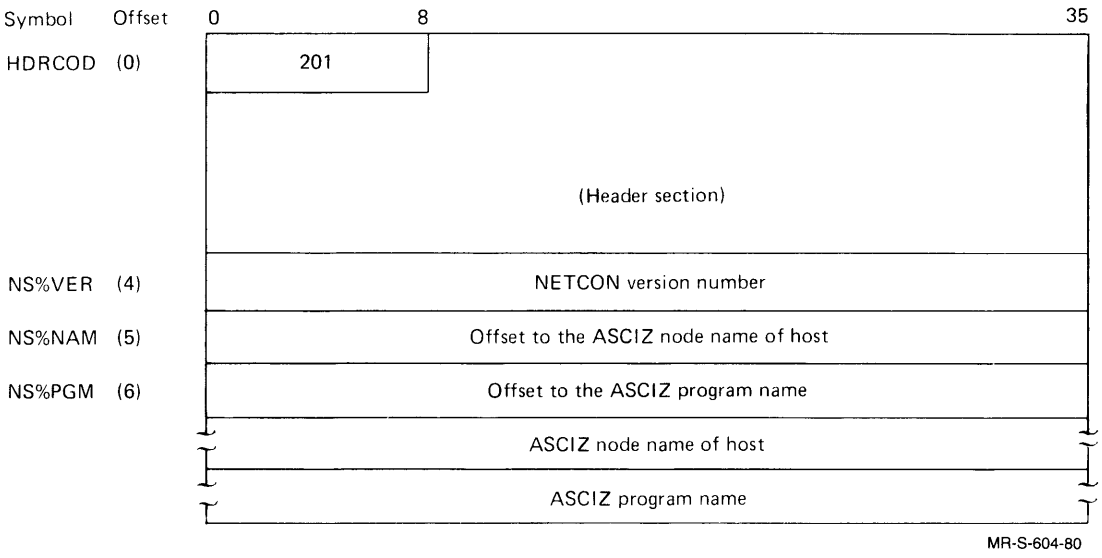


Figure C-2 SYSERR Entry for a NETCON Load

C.2.2 Node Load Entry (event 202)

Whenever NETCON loads a node, a SYSERR entry is made to record information such as the nodes involved, the line used, the load file specification, and any return code. The format and contents are shown in Figure C-3.

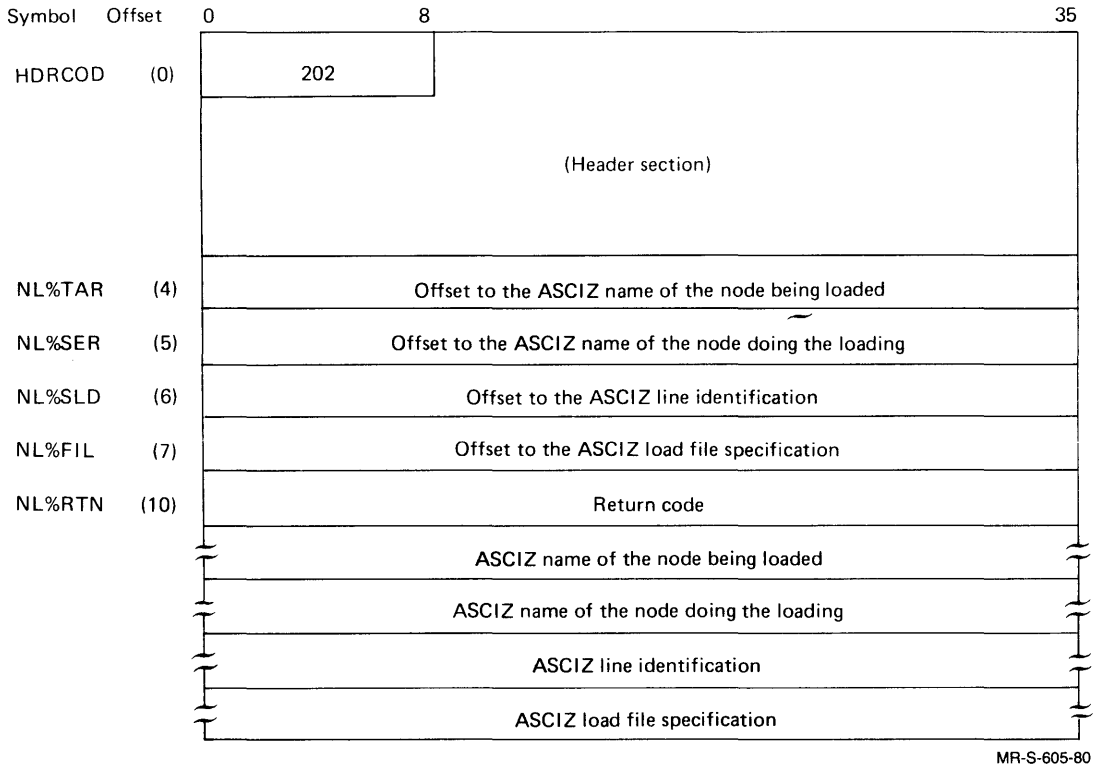


Figure C-3 SYSERR Entry for a Node Load

C.2.3 Node Dump Entry (event 203)

Whenever NETCON dumps a node, a SYSERR entry is made to record information such as the nodes involved, the line used, the dump file specification, and any return code. The format and contents are shown in Figure C-4.

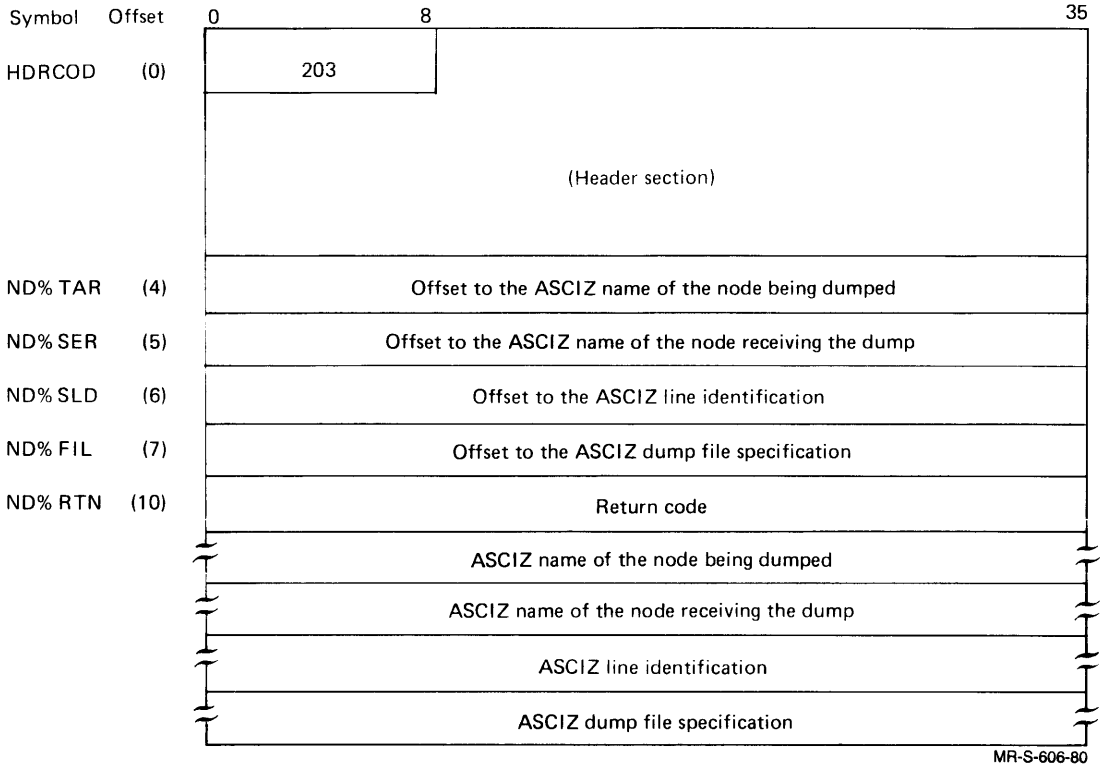


Figure C-4 SYSERR Entry for a Node Dump

C.2.4 Hardware Error Entry (event 210)

Whenever a hardware option detects a failure, a SYSERR entry is made to assist in the network maintenance function. The information recorded includes the environment of the entry, the hardware option that detected the error, the software running in the node where the error is detected, and a detailed log data description of the hardware option reporting the failure. The format and contents of the hardware error entry are shown in Figure C-5.

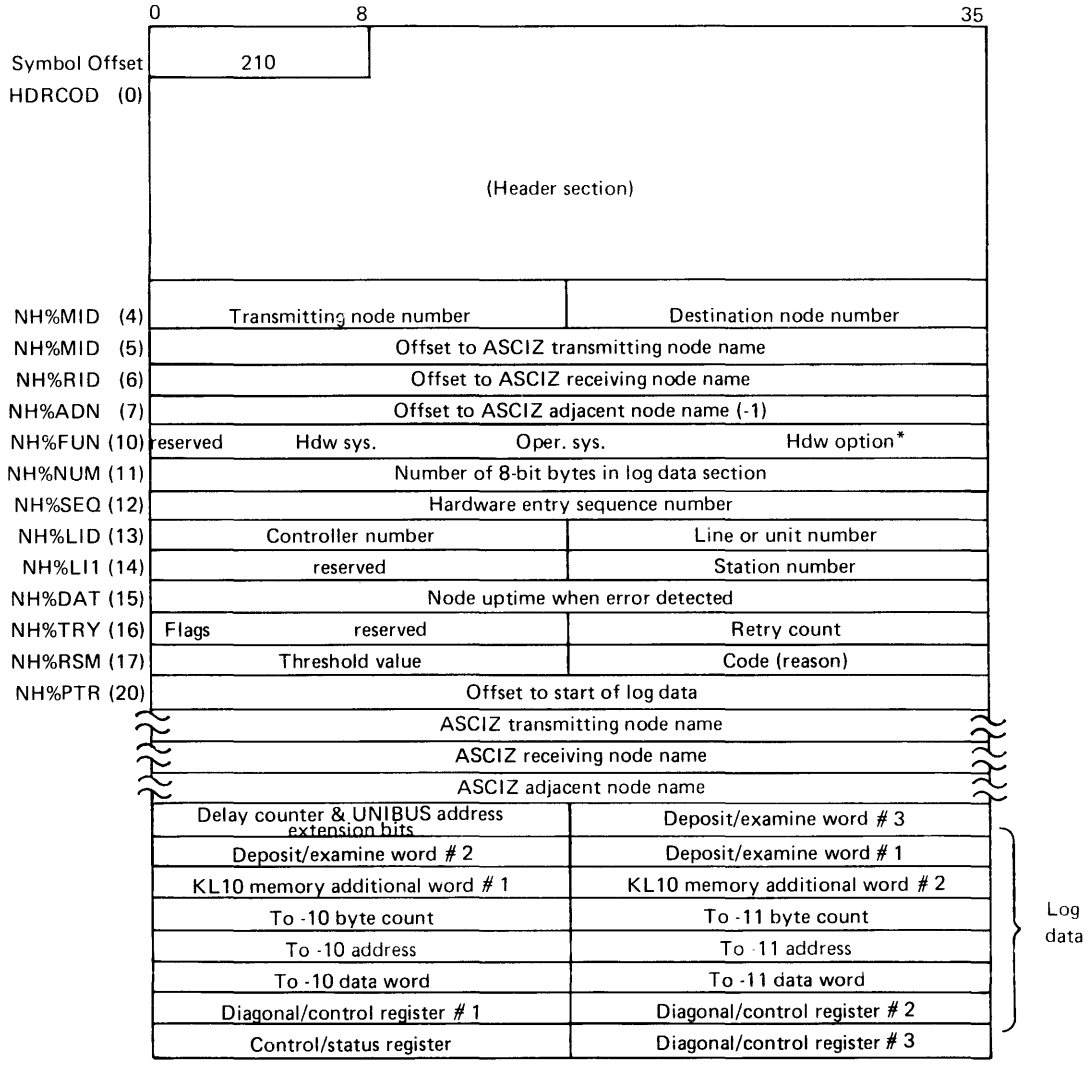


Figure C-5 SYSERR Entry for Hardware Detected Failures

Log data for hardware option KMC11/DUP11 (003). The initial 21 bytes for this hardware option type follows.

<table>
<tr><td colspan="4">Offset to first word of sub-device table</td></tr>
<tr><td colspan="2">Input transfer & maintenance control/status register</td><td colspan="2">Output transfer control/status register</td></tr>
<tr><td colspan="2">1st half of data port</td><td colspan="2">2nd half of data port</td></tr>
<tr><td>0</td><td>Microcode type</td><td>0</td><td>Microcode version</td></tr>
<tr><td colspan="2">0</td><td colspan="2">0</td></tr>
<tr><td colspan="2">0</td><td colspan="2"></td></tr>
</table>

MR-S-608-80

Figure C-5.1 Log Data for KMC11/DUP11

Receiver status register	Receiver data buffer register
Transmitter status register	Transmitter data buffer register

MR-S-609-80

Figure C-5.2 Sub-device Table for Hardware Option DUP11 (003)

C.2.5 CHK11 Diagnostic Entry (event 220)

CHK11, a hardware test module, is started when the DN20 or DN200 is loaded. All messages from CHK11, messages concerning available memory, and number and status of devices as well as detected errors, become part of the CHK11 diagnostic entry.

The format of the environment and reporting device section that follows the header and precedes the log data section is the same as that for event 210 (hardware error entry.) Symbols NH%MID through NH%PTR become NO%MID through NO%PTR. The log data is an ASCIZ CHK11 message of arbitrary length. The format and contents are shown in Figure C-6.

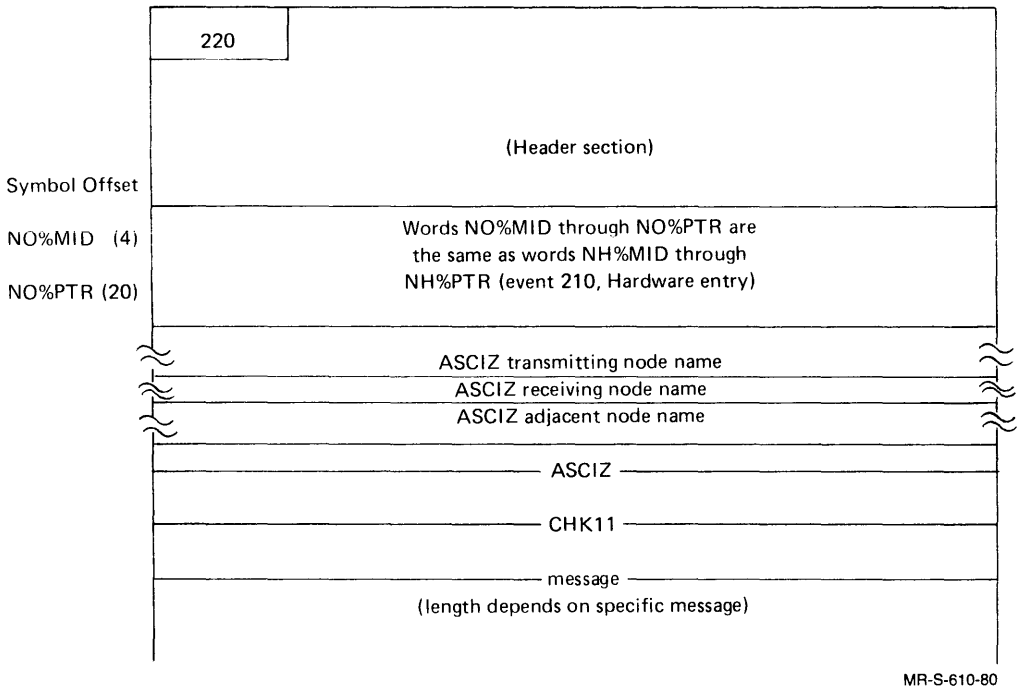


Figure C-6 SYSERR Entry for CHK11 Diagnostic Entry

C.2.6 Line Status Entry (event 230)

Periodically, NETCON records the status of each communications line via a SYSERR entry. The information that is logged includes the node name, line identification, and a variable number of line counters. The format and contents are shown in Figure C-7.

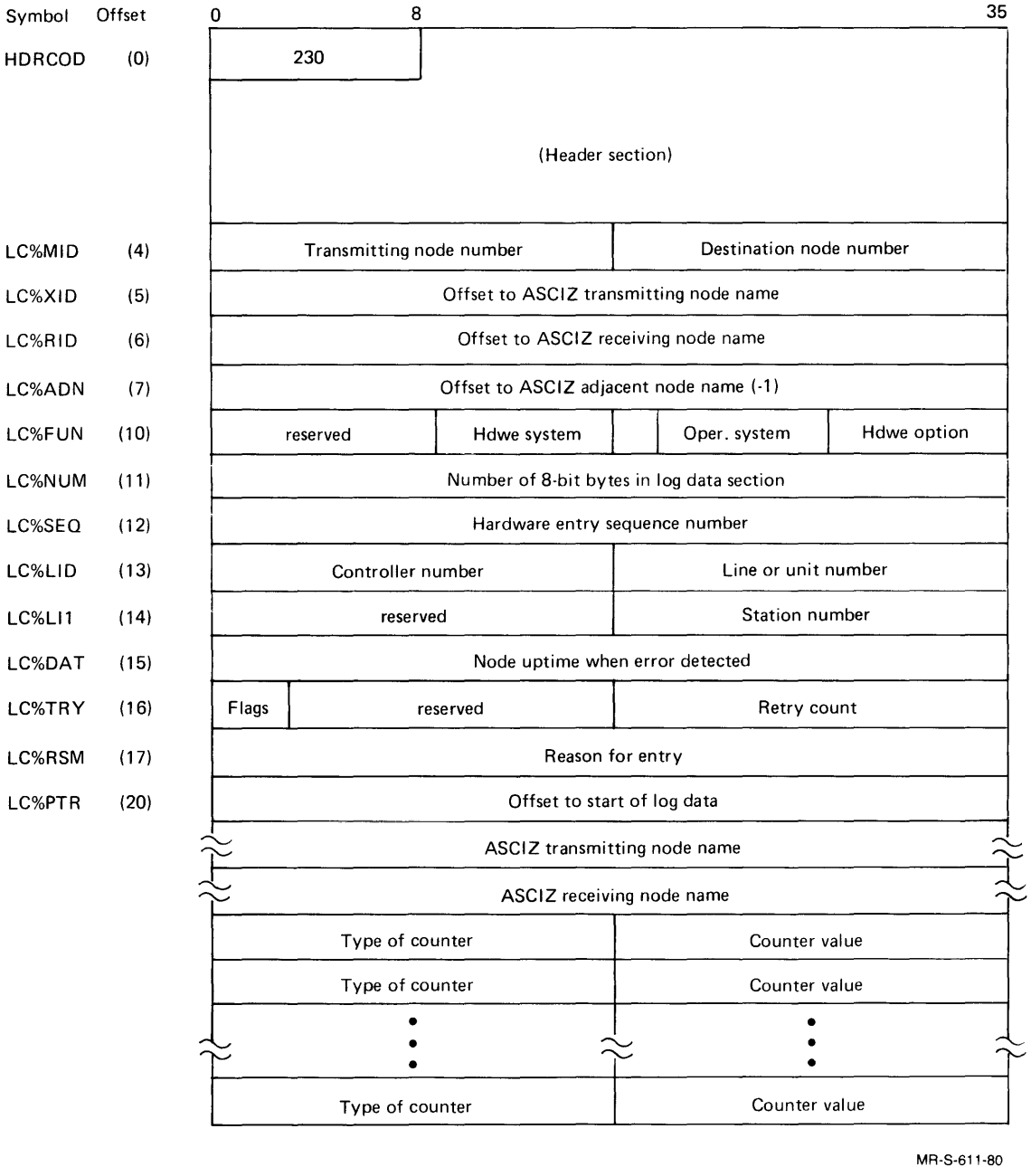


Figure C-7 SYSERR Entry for Line Statistics

The line counter entries are in half-word format, with the line counter type (0-10 octal) in the left half and the item count in the right half. The current line counter types are described in Table C-1. The number of line counter values that are actually recorded depends on the type of controller on the line.

Table C-1
Line Counter Types

Type (Octal)	Count of	Meaning
0	seconds since last zeroed	This count is incremented once a second. It is cleared when the other counters are cleared and thus provides a time frame for their values.
1	blocks received	This is the number of blocks successfully received over the line.
2	blocks sent	This is the number of blocks successfully sent over the line.
3	retransmissions, line errors	This is the number of blocks retransmitted because of CRC or parity errors.
4	received line errors	This is the number of blocks received with CRC or parity errors.
5	retransmissions, other than errors or parity	This is the number of blocks retransmitted for reasons other than CRC or parity errors.
6	receive timeouts	This is the number of times the receive timer has expired.
7	receive selection errors	This is the number of times blocks have been received with invalid selection data.
10	resource errors	This is the number of times such things as buffers have not been available.

APPENDIX D

BIBLIOGRAPHY

The following DIGITAL documents contain detailed information on the protocols used in DECnet.

DECnet DIGITAL Network Architecture:

Digital Data Communications Message Protocol (DDCMP)	AA-D599A-TC
Network Services Protocol (NSP)	AA-D600A-TC
Data Access Protocol (DAP)	AA-D601A-TC
Maintenance Operation Protocol (MOP)	AA-D602A-TC

The following published books are a source of general network information. The list is both arbitrary and abbreviated and is intended to be a catalyst to stimulate the reader's interest.

Title	Author	Publisher
Introduction to Data Communications	Murphy and Kalis	DIGITAL
Introduction to Mini-Computer Networks	--	DIGITAL
Technical Aspects of Data Communications	McNamara	DIGITAL
Introduction to Tele-Processing (and others)	Martin	Prentice-Hall
Basics of Data Communications	Karp, ed.	McGraw-Hill
Communications Networks for Computers	Davies and Barker	Wiley and Sons
Handbook of Data Communications	Editors	National Computing Centre Publications (United Kingdom)
Data Communications	Doll	Wiley Interscience

APPENDIX E

GLOSSARY

acceptance tests	Acceptance tests refer to the verification and certification procedures.
asynchronous transmission	Transmission in which the time intervals between transmitted characters may be of unequal length because each character contains its own start and stop bits. This is also known as start/stop transmission.
certification procedure	The DECnet certification procedure ensures that the various components of the DECnet-20 system function at a basic, minimal level. The DECnet certification procedure is performed after the DECnet configuration, installation, and verification procedures are performed.
computer network	An interconnection of computer systems, I/O devices, and communications facilities.
configuration	The process of customizing the DECSYSTEM-2040/50/60. Using the configuration tools, the DECnet user establishes the network parameters specific to the DECnet communications front end (and RJE station, if included) being configured.
configuration tools	The programs NETGEN, TKB20, VNP20, and DNMAC. These are the programs needed for on-site configuration of DECnet-20 on a DECSYSTEM-2040/50/60.
connect	The process of creating a logical link.
connect password	A 1- to 8-character password used to validate access privileges between tasks on a network.

DAP (Data Access Protocol)	A set of standardized formats and procedures that facilitate the creation, deletion, transfer, and access of files between a user process and a file system existing in a network environment.
data transmission	The sending of data from one computer to another over a physical link, or from one task to another over a logical link.
DDCMP	Digital Data Communications Message Protocol. A formal set of conventions designed to provide error-free, sequential transmission of data over physical links.
disconnect	The process of closing a logical link.
DMC11	A single line microprocessor-based interface to the network. The DMC11 is a synchronous DMA device.
DN20	A DECnet-20 communications front end. As used in this manual, DN20 is a generic term and includes both the DN20 and the DN21.
DNMAC	The DNMAC program is the cross assembler for PDP-11 macro source files.
down-line loading	Transmitting a program's memory image over a logical link and loading and starting the program on a computer at another node.
DTE	The hardware interface between the main processor in a DECSYSTEM-2040, 2050, or 2060 and the PDP-11 processor in the DN20 communications front end.
duplex	Simultaneous independent transmission in both directions. Also referred to as full-duplex or two-way simultaneous.
FAL (File Access Listener)	The FAL program resides on a DECnet host and acts as the target for requests made by the NFT programs residing on remote DECnet hosts. The FAL program is responsible for determining a user's access privileges to a requested file and the subsequent honoring or rejecting of the request.
full-duplex	See duplex.
half-duplex	Transmission in either direction, but not in both directions simultaneously. Also referred to as two-way alternate.
host computer	A computer at a network node that primarily provides services such as computation, data base access, special programs, or programming languages to other nodes in the network.

installation procedure	DECnet installation is the process of setting up the DECnet software and modifying several system files to include DECnet-related jobs. DECnet installation occurs after DECnet configuration.
interrupt message	A high-priority message used to inform another task of some significant event.
local node	A relative term indicating the node at which your task is executing or at which your terminal is logged in.
local NSP	NSP executing in the local node.
local task	A task executing at a local node.
logical link	A virtual data path between two tasks in a network that permits them to communicate.
logical node	The logical node is the node to which the system sends a user's queued output. At login time, the logical node is the same as the physical node. The user may specify a logical node by using the SET LOCATION command from TOPS-20 command level.
loop-back	A mode of operation where data transmitted by a network task is reflected at some point along the communication path and is returned to the originating task.
modem	In networks, a device that makes computer signals compatible with communications facilities.
MOP	Maintenance Operation Protocol. A formal set of messages and rules used to load and dump computer memory as well as test a communications link between two adjacent nodes.
NCP	NCP is the name of the network control program that processes DECnet-20 network control commands. NCP refers to one of the three major programs of NETCON.
NCU	NCU is the name of the network control utility that executes network control commands. NCU implements the Network Information and Exchange (NICE) protocol. NCU refers to one of the three major programs of NETCON.

NETCON	NETCON is a specialized network control task that accepts requests via operator commands and command files. NETCON processes network control requests such as the loading or dumping of communications front end software, the loading or dumping of synchronous line controllers, the displaying of line statistics for any line, etc. See Sections 1.4 and 1.6 in Chapter 1 for details.
NETGEN	NETGEN is the interactive program used to describe the hardware and software during the configuration procedure for a DECnet node.
network	An interconnected or interrelated group of nodes. In this manual, network is synonymous with computer network.
network dialogue	An exchange of information between two tasks in a network.
network task	A task engaged in, or willing to engage in, a network dialogue. In NSP documentation, a network task is also referred to as a network object.
NFT (Network File Transfer)	A program that allows you to access or delete files residing on DECnet hosts that provide network file access capabilities. NFT initiates the service requests that will be carried out by the FAL program.
NICE protocol	NICE is the acronym for the Network Information and Control Exchange protocol that enables various DIGITAL computers to access the information and control facilities of remote nodes on the same network.
node	An endpoint of any branch of a network, or a junction common to two or more branches of a network. In this manual, the DECSYSTEM-20 and any communications front ends are all considered nodes.
node name	A 1- to 6-character name uniquely identifying a node within a network. Node names can be any combination of the characters A through Z, and 0 through 9.
node number	A number uniquely identifying a node within a network. Although NSP allows node numbers of 2 through 240, DECnet-20 nodes may only be assigned numbers 2 through 127.

NSP	Network Services Protocol. A formal set of conventions used in DECnet to perform network management and to exchange messages over logical links. NSP also refers to the program that implements the NSP protocol. (In the text, NSP refers to the program; NSP protocol is used to refer to the protocol.)
OPR	OPR is the Operator Command Language program that provides the operator with one command language to communicate with several TOPS-20 components. OPR processes commands for syntax and passes syntactically correct commands to ORION. (See the ORION definition.)
ORION	ORION is the operator control program that accepts commands from OPR and forwards the commands to the proper program for execution. ORION forwards NCP commands to NETCON.
packet	A group of bits, comprising data and control information, which is transmitted as a composite whole over a physical link. The data, control, and possibly error control information are arranged in a specified format. (The basic transmission unit of DDCMP).
physical link	A communications path between two adjacent nodes. This can be in the form of a dial-up line, leased line, radio, satellite link, or a channel-to-channel connector such as a DTE.
physical node	A physical node consists of the hardware comprising a node. See also the definition of a node.
protocol	A formal set of conventions or rules governing the format and relative timing of message exchange.
remote node	A node in a network that is not your local node.
remote NSP	NSP executing in a remote node.
remote task	A task executing in a remote node.
server task	An alternate designation for a task that has declared itself willing to accept a network connection, usually to provide some system service.
source node	The node at which the request for a connection is initiated or from which a message is transmitted.
source task	The task in which the request for a connection is initiated or from which a message is transmitted.

synchronous transmission	Transmission in which time intervals between transmitted characters are of equal length. Multiple characters can be transmitted without start or stop bits following an initial synchronizing sequence.
target node	The node at which the request for a connection is accepted or rejected or to which a message is transmitted.
target task	Any task that has declared itself willing to accept a network connection.
TKB20	TKB20 is the program that constructs PDP-11 formatted task images from object files during the configuration procedure for a DECnet node.
UETP	UETP is the acronym for User Environment Test Package. UETP is a collection of programs, data files, and batch control files designed to allow the user to verify the integrity of the operating system. UETP acts as an interface between the user and the batch system by allowing the user to selectively access, enable, and run tests in the form of batch control files. In this manual, UETP is used during the installation, verification, and certification procedures.
up-line dumping	Transmitting a copy of a computer's memory over a logical link and storing it in a file on another node.
verification procedure	The DECnet verification procedure ensures that the appropriate DECnet-20 software modules have been installed. The verification process checks for the existence of each required module and for the correct version of each required module.
VNP20	VNP20 is the program that creates the communication front end system image during the configuration procedure for a DECnet node.

APPENDIX F

SAMPLE PROGRAMS FOR REMOTE LOGIN

F.1 INTRODUCTION

This appendix describes a set of three programs that use DECnet-20 task-to-task communications capabilities to allow a terminal at a local host to log in to a remote host. The two environmental requirements for running these programs are that a DECnet connection must exist between the two hosts and that both hosts must be running the TOPS-20 operating system at or above Release 3A level.

The three programs that implement the remote login function are NRT20, STNRT, and NRTSRV. The function of each program is defined briefly below:

NRT20	provides a network source task at the source terminal's local node, accepts data from the terminal, sends the data to the remote host via a network connection, and passes output from the remote host to the source terminal.
NRTSRV	provides a server task at the remote host to accept the network connection from NRT20, passes the received data to a pseudo-TTY, and returns output from the remote host to the network connection. The output is read by NRT20 and sent to the source terminal.
STNRT	creates a job that runs NRTSRV on the remote host.

F.2 NRT20 PROGRAM

NRT20 allows a user at a terminal on a TOPS-20 system to log in to a remote host in a DECnet network. Note that the type of operating system on the remote host is immaterial to the NRT20 program. However, this appendix describes the use of NRT20 in conjunction with the NRTSRV program which sets up a server task on a TOPS-20 host.

NRT20 defines a network source task and establishes a task-to-task network connection between the source task and a previously defined target task at the remote host. NRT20 passes source terminal input to the network connection and passes the remote host's output to the source terminal. The program also provides for a special escape character by which the user can exit normally from NRT20 and return to TOPS-20 (EXEC) command level at the local node. Finally, NRT20 monitors the network connection and handles any unexpected break in the connection.

The NRT20 source listing provided in this appendix contains code that implements the functions described immediately below. The order in which the functions are coded in the listing parallels the order in which the functions are presented in this description.

F.2.1 NRT20 Initialization Routines

NRT20 performs the following actions to create the proper environment for executing its major functions:

- Creates an inferior process to read input from the source terminal and to send the source terminal's input to the remote host.
- Enables the job to trap ^C interrupts and send the ^C characters to the remote host.
- Initializes, activates, and enables the software interrupt system to handle input of a special escape character from the terminal. This permits the user to exit from the remote host and return to EXEC level at the local node.
- Permits the user to define the special escape character either by default or by explicit specification. This action occurs as part of the initial NRT20-user dialogue, as shown in Section F.6 "Using the Remote Login Capability".
- Disables echoing and sets binary data mode for the terminal. This is done so that the monitor at the remote host can assume responsibility for echoing and character handling and translation for the terminal.
- Opens a network connection to the server task at the remote host. NRT20 uses the file specification:

  ```
  DCN:node-200
  ```

  where node is the remote host's node name as specified by the user in the NRT20-user dialogue and 200 is the object type for the server task NRTSRV. If the object type 200 is already in use, another object type should be selected instead. (Refer to Appendix B for a list of available object types.) If another object type is selected, the user must modify both the DCN: specification in the NRT20 program and the SRV: specification in the NRTSRV program.

F.2.2 NRT20 Operations

NRT20 handles data transmission between the terminal and the remote host by performing the following actions:

- Starts an inferior process that reads terminal input and sends the input to the remote host. The inferior process sends all characters (except the special escape sequence) to the remote host and passes all data received from the remote host to the terminal.

- If the network connection is broken unexpectedly, NRT20 prints the following message:

  ?Connection broken. Reason: nn: reasontext

  where nn is the DECnet disconnect reason code and reasontext is the explanation for the disconnect. Following the printing of the message, NRT20 returns to EXEC command level. If the user enters CONTINUE at this point, NRT20 restarts processing from the beginning, as though the user had entered NRT20 in response to the @ monitor prompt.

- When the user enters the special escape sequence, NRT20 freezes the inferior process and prints

  [Connection broken, back at node node-name,
  Type CONTINUE to resume connection]

  where node-name is the node name of the local node to which the terminal is attached. NRT20 returns to EXEC level where the user may resume NRT20 processing at the point of the interrupt by entering CONTINUE. If the user enters CONTINUE, NRT20 prints:

  %Reconnecting to remote node...

  and unfreezes the inferior process. At that point, the user is connected to the remote system and processing is resumed. (Section F.6.3 contains an example of this situation.)

F.3 STNRT PROGRAM

The server task NRTSRV permits remote users to log in to the host on which NRTSRV is running. For security reasons, NRTSRV must not be run from a logged in job. If NRTSRV were run from a logged in job, any user could log in to NRTSRV with the same user name and privileges as specified for the job under which NRTSRV is running without supplying a password. The use of the program STNRT prevents such a potential breach of system security.

STNRT creates a job with the following characteristics:

- The job is not logged in.
- The job has the file SYS:NRTSRV.EXE as its top-level process.
- The job is not owned by the user running STNRT.

STNRT must be included in the PTYCON.ATO file to initialize the NRTSRV program at system startup time. (Inclusion of the STNRT job in the PTYCON file is described in Section F.5 "Installation", below.) At system initialization time, STNRT creates a job with the characteristics described above and also:

- Starts the job.
- Detaches the job.
- Halts execution and returns to EXEC.

F.4 NRTSRV PROGRAM

NRTSRV is a server task that allows users at remote terminals to log in to the system on which NRTSRV is running. The number of users that may run concurrently under NRTSRV is set by using the MAXFRK parameter in the NRTSRV.MAC source file.

NRTSRV implements remote logins by connecting to pseudo-terminals on its local host. Therefore, the maximum number of users cannot exceed the number of pseudo-terminals on the host on which NRTSRV is running. Before setting the value of MAXFRK, the user should take into consideration other processes that require the use of PTYs, such as BATCH and PTYCON. Other factors that affect the maximum number of users of the remote login capability are the maximum number of forks permitted by the remote system and the system manager's evaluation of the demand for the remote login capability. The value of MAXFRK is set to 7 on the distribution tape.

When NRTSRV receives a network connection, it opens a pseudo-terminal (PTY). As NRT20 sends messages over the network connection to NRTSRV, NRTSRV passes the messages to the PTY. Conversely, as NRTSRV's host system sends messages to the PTY, NRTSRV passes the messages over the network connection to NRT20.

The NRTSRV source listing provided in this appendix contains code that implements the functions described immediately below. The order in which the functions are coded in the listing parallels the order in which the functions are presented in this description.

F.4.1 NRTSRV Initialization Routines

NRTSRV performs the following actions to create the proper environment for executing its major functions:

- Sets the private name of the program being run by the job set up by STNRT to NRTSRV.
- Verifies that the DECnet server device SRV: exists on the system. If SRV: does not exist on the system, the job is logged out.
- Enables the system-permitted capabilities for the job.
- Creates and starts n inferior processes, each of which is a copy of NRTSRV.
- Disables the system-permitted capabilities for the top-level process.
- Sets up a routine to handle the abnormal termination of any inferior process.

The job capabilities are originally enabled for the top-level process so that they may be passed on to the inferior processes. The NRTSRV program ensures that no logins occur between the time that the capabilites are enabled for the top-level process and the time that they are disabled after having been passed on to the inferior processes. A login during that time could constitute a breach of system security.

There are two situations in which NRTSRV must guard against undesired logins: (1) during the time that the job capabilities are originally passed on to the inferior processes and (2) during the time that a new inferior process is created to replace an inferior process that terminated abnormally. In both cases, NRTSRV prohibits logins by freezing the inferior processes until the job capabilties have been passed on.

If you intend to modify the NRTSRV source for your system, it is recommended that you preserve this "no login" feature.

F.4.2 NRTSRV Operations

NRTSRV accepts input from the terminals and transmits PTY outputs in the manner described below:

- Each inferior process performs the following actions:

  - Clears, then sets up and initializes the software interrupt system.
  - Removes any PTY or server JFN assignments.
  - Opens a DECnet server connection with the filespec SRV:200 which specifies a DECnet generic server with object type 200.
  - Assigns and enables software interrupt channels for connect event notification and data available events for the server JFN.
  - Awaits a software interrupt signalling an incoming connection.

  When the inferior process receives a connect request, it requests a PTY to be used to log in the remote terminal. If a PTY is not available, the inferior process sends the following message to NRT20 to be printed at the user's terminal:

  ```
  ?No more PTYs on remote host
  ```

  Processing is then restarted as described above for inferior processes. If a PTY connection is available, the NRTSRV inferior process does the following:

  - Issues a ^C to the PTY. The system then issues its standard banner message.
  - Accepts the connection from the network and sets up a routine run at background level to read data from the PTY and pass it down the network logical link to the NRT20 program.
  - When a network connection data available is signalled, the inferior process reads the data and passes it to the PTY

Data processing continues until the link is broken. When the link is broken, processing is restarted as described above for inferior processes.

- In the event of an abnormal termination of an inferior process, NRTSRV

  - Freezes all inferior processes.
  - Scans to identify the inferior process that terminated.
  - Kills the inferior process.
  - Creates and starts another inferior process.
  - Resumes all inferior processes.

F.5 INSTALLATION PROCEDURES

The programs required to implement the remote login capability are included on the distribution tape as part of the release of TOPS-20 Version 4. The following files are relevant:

CMD.UNV	STNRT.REL
CMD.REL	STNRT.EXE
NRT20.MAC	STNRT.LST
NRT20.REL	NRTSRV.MAC
NRT20.EXE	NRTSRV.REL
NRT20.LST	NRTSRV.EXE
NRT20.MAC	NRTSRV.LST

CMD.UNV and CMD.REL are associated with the NRT20 files and are relevant only if modifications to the NRT20.MAC source code are contemplated.

Installation of the remote login capability involves two steps: restoring the necessary executable files from the distribution tape and modifying the PTYCON.ATO file **on each host on which NRTSRV is to run** to automatically start NRTSRV at system initialization time.

F.5.1 Restoring Files from the Distribution Tape

Follow the steps below to restore the required files from the distribution tape:

1. Log in as OPERATOR (or as any user with wheel or operator privileges) and enable capabilities.

2. Connect to the directory in which your installation's unsupported software is maintained. (The directory should be in the system search list SYS:.) For example:

   $connect ps:<unsupported>(RET)

   connects to the directory PS:<UNSUPPORTED> which contains the system's unsupported software. You should replace PS:<UNSUPPORTED> with the name of the directory that maintains your installation's unsupported software.

3. Mount the distribution tape and run DUMPER to restore the following executable files: NRT20.EXE, NRTSRV.EXE, and STNRT.EXE. Also, you may optionally restore any other supporting files that you want for your installation.

F.5.2 Editing the PTYCON.ATO File

Prior to using the remote login capability, you must edit your installation's PYTCON.ATO file which is processed at system start-up time. Add a line to run the STNRT program at a convenient place in one of the operator jobs running under PTYCON. For example, the following sequence shows the editing of a PTYCON.ATO file. The operator edits the file to insert the STNRT command immediately above the PUSH command.

```
$type ptycon.ato(RET)
SILENCE
LOG
DEFINE ^$OPR
CONN OPR
LOG OPERATOR FOO OPERATOR
ENA
!NEW OPERATOR INTERFACE PARSER
OPR
TAKE SYSTEM:NCP.CMD
TAKE SYSTEM:DN200.CMD
PUSH
ENA
^ESET LOGIN ANY
^ESEND * SYSTEM IN OPERATION
APPEND PS:<SPOOL>ORION-SYSTEM-LOG.002.* PS:<SPOOL>ORION-OLD.LOG
DELETE PS:<SPOOL>ORION-SYSTEM-LOG.002.*
POP
^X
NO SILENCE
W ALL
CONN OPR

$edit pytcon.ato(RET)
Edit: PTYCON.ATO.1
*fPUSH$^:*(RET)
01100   PUSH
*i.-1(RET)
01050   STNRT(RET)
*eu(RET)

[PTYCON.ATO.2]
$
```

F.6 USING THE REMOTE LOGIN CAPABILITY

Each time the system is brought up, the edited PTYCON.ATO file runs the STNRT program. STNRT loads, starts, and then detaches the server task NRTSRV which automatically sets up the inferior processes that wait for a connection request from a remote terminal. Users at remote terminals may then run NRT20 to access the host on which NRTSRV is running.

F.6.1 Logging in to a Remote Host Using NRT20

You invoke NRT20 by entering some form of the NRT20 command in response to the TOPS-20 prompt at your terminal. NRT20 expects you to specify the remote host's node-name and a special escape character to be used to exit from NRT20. If you do not specify a special escape character, NRT20 uses ^Y by default.

To specify the remote host's node-name and use ^Y as the special escape character, use the following NRT20 command format:

```
@NRT20 node-name(RET)
```

NRT20 sets ^Y as the special escape character and prints the following message:

```
[Type ^Y to return to node node-name]
```

where node-name is the name of the terminal's local host. If you enter an invalid node name (or if no physical connection exists), NRT20 prints:

```
Connection broken. Reason: 39: Network failure. No path to
remote node.
```

and terminates processing. You are returned to EXEC command level.

To specify a different special escape character, use one of the following NRT20 command formats:

```
@R NRT20(RET)
```

or

```
@NRT20(RET)
```

NRT20 then prompts for the special escape character:

```
Escape character (^Y):
```

Press the control key and enter any ASCII character between A and Z except for the following system-recognized control characters: ^C, ^F, ^G, ^I, ^J, ^M, ^O, ^Q, ^R, ^S, ^T, ^U, and ^W. In addition, if you have enabled any other control characters, they are also invalid as the special escape character for NRT20. If you enter a carriage return, NRT20 uses ^Y as the special escape sequence.

If you did not enter a node name, as shown in the two NRT20 command lines above, NRT20 prompts you for one as follows:

```
Host name:
```

Enter the remote host's node name. If you enter a carriage return, NRT20 reissues the prompt until you enter a valid node name or a ^C. If you enter an invalid node name, NRT20 responds in the manner previously described by printing an error message and terminating processing.

After a successful connection to the remote host, NRT20 prints the remote host's standard banner message on your terminal. After the banner message is printed, you may perform any function permitted by the remote host.

F.6.2 Exiting from a Remote Host Using NRT20

To exit normally from the remote host, type the special escape sequence selected (or defaulted) in the initial NRT20 dialogue. (See Section F.2.2 for NRT20's response and your possible actions when an abnormal disconnection occurs.) When you enter the special escape sequence, NRT20 prints the following message on your terminal:

```
[Connection broken, back at node node-name,
Type CONTINUE to resume connection.]
```

You may continue NRT20 execution from the point of the interrupt by entering the CONTINUE command. NRT20 responds:

```
%Reconnecting to remote node...
```

At this point, the connection is restored and the terminal is again connected to the remote host.

NOTE

It is strongly recommended that you log off the remote system before breaking the network connection between the local system and the remote system. If you do not log off before breaking the connection, the job will exist on the remote system in a DETACHED state and may require operator intervention to kill it.

F.6.3 Sample Terminal Sessions Using NRT20

Note that, when used within the context of the remote login capability, ^S, and ^Q do not stop the scrolling of pages on the user's terminal. This occurs because NRT20 does not trap these control characters and also because one cannot use the TERMINAL PAGE command to set the printing mode of a PTY.

Example 1.

The following example shows user KAMANITZ at a terminal on system 2102. He first logs in to 2102, then uses NRT20 to connect to the remote system 4097. By entering the remote system's node name on the NRT20 command line, KAMANITZ allows NRT20 to use ^Y as the special escape character. He then logs in to system 4097 as user CRUGNOLA, performs functions on system 4097, and logs off 4097. After logging off of system 4097, he presses ^Y to return to system 2102. From 2102, he issues a LOGOUT command to log off system 2102.

```
 2102 Development System, TOPS-20 Monitor 4(3117)
@LOGIN KAMANITZ password 341(RET)
 Job 16 on TTY106 19-Sep-79 14:41:15
@NRT20 4097(RET)
[Type ^Y to return to node 2102]

 4097 Load-Test System, TOPS-20 Monitor 4(3100)
@LOGIN CRUGNOLA password 341(RET)
 Job 12 on TTY50 19-Sep-79 14:41:27
    .
    . (User performs functions on system 4097.)
    .
@LOGO(RET)
KILLED JOB 12, USER CRUGNOLA, ACCOUNT 341, TTY 50,
  AT 19-SEP-79 14:41:49, USED 0:00:02 IN 0:00:21

(User enters ^Y here which is not echoed.)

[Connection broken, back at node 2102,
Type CONTINUE to resume connection]
@LOGO(RET)
KILLED JOB 16, USER KAMANITZ, ACCOUNT 341, TTY 106,
  AT 19-SEP-79 14:46:53, USED 0:00:01 IN 0:05:37
```

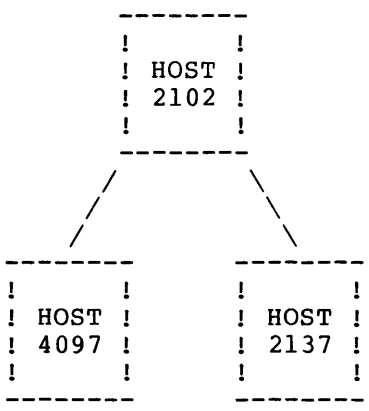
Example 2.

In the following example, user KAMANITZ logs in to system 2102. Then, he uses NRT20 to connect to system 4097 and specifies ^B as the special escape character. Once he is connected to 4097, he logs on as user CRUGNOLA and begins listing his directory. While system 4097 is printing information about his directory, he presses ^B to gain the attention of NRT20 on system 2102. From 2102, he enters CONTINUE in response to the EXEC prompt and NRT20 reconnects to system 4097. System 4097 resumes printing information about his directory. (Note that the directory listing does not contain the lines that would have printed during the time used to escape from and return to system 4097.) User KAMANITZ then logs off system 4097, returns to system 2102, and logs off system 2102.

```
 2102 Development System, TOPS-20 Monitor 4(3117)
@LOGIN KAMANITZ password 341(RET)
 Job 62 on TTY106 19-Sep-79 14:50:59
@NRT20(RET)
Escape character (^Y):(CTRL/B)
Host name:4097(RET)

4097 Load-Test System, TOPS-20 Monitor 4(3100)
@LOGIN CRUGNOLA password 341(RET)
 JOB 13 ON TTY50 19-Sep-79 14:53:09
@VDIR(RET)

  PS:<CRUGNOLA>
CALEND.EXE.1;P777700       5 (User enters CTRL/B. It does not
echo.)
[Connection broken, back at node 2102,
Type CONTINUE to resume connection]
@CONTINUE(RET)
Reconnecting to node...
10-6-AUG-79 15:51:41
 LA36.CMD.1;P777700      1 74(7)  13-MAR-78 15:39:48
 LOGIN.CMD.2;P777700     1 21(7)  13-MAR-78 15:36:23
 MAIL.TXT.1;P770404      1 175(7) 26-JUL-79 13:01:46
 SWITCH.INI;2;P777700    1 39(7)  18-MAY-78 15:33:10
 VT50.CMD;1;P777700      1 28(7)  13-MAR-78 15:37:44
 VT52.CMD.1;P777700      1 60(7)  13-MAR-78 15:38:49
 ZIP.Q;1;P777700         1 45(7)  14-JUN-78 17:06:25

 TOTAL OF 13 PAGES IN 9 FILES
@LOGO(RET)
KILLED JOB 13, USER CRUGNOLA, ACCOUNT 341, TTY 50
  AT 19-SEP-79 14:55:45, USED 0:00:03 IN 0:02:36

(User enters CTRL/B.  It does not echo.)

[Connection broken, back at node 2102,
Type CONTINUE to resume connection]
@LOGO(RET)
KILLED JOB 62, USER KAMANITZ, ACCOUNT 341, TTY 106,
  AT 19-SEP-79 14:56:53, USED 0:00:01 IN 0:05:37
```

Example 3.

This example shows the use of NRT20 to log in to a remote host that is not an adjacent node in the network. The network is configured according to the following diagram:



User CRUGNOLA on host 4097 wants to log in to host 2137. To do so, he first logs in to host 4097. Then, he uses NRT20 to establish a network connection to host 2102 using the default special escape character. Once the connection to host 2102 is established, user CRUGNOLA logs in to host 2102 as user KAMANITZ. From host 2102, KAMANITZ uses NRT20 to establish a network connection to host 2137, specifying ^B as the special escape character.

Note that a different special escape character must be used for the connection between hosts 2102 and 2137. If the same special escape character were used for both network connections, NRT20 on 4097 would trap the special escape character and return control of the user's terminal to host 4097. This would interrupt the connection between hosts 4097 and 2102. Once this occurred, no means would exist to gain access to host 2102 on the existing connection.

After the network connection is established between hosts 2102 and 2137, user KAMANITZ logs in to host 2137 as user OSMAN. OSMAN performs some functions on host 2137, then logs off and enters ^B to return to host 2102. Once at host 2102, user KAMANITZ logs off and enters ^Y to return to host 4097. Once there, user CRUGNOLA logs off.

```
 4097 Load-Test System, TOPS-20 Monitor 4(3100)
@LOG CRUGNOLA password 341(RET)
 Job 7 on TTY33 25-Sep-79 10:18:05
@NRT20 2102(RET)
[TYPE ^Y TO RETURN TO NODE 4097]

 2102 Development System, TOPS-20 Monitor 4(3117)
 LOG KAMANITZ password 341(RET)
 Job 42 on TTY217 25-Sep-79 10:19:17
@NRT20 (RET)
Escape character (^Y):(CTRL/B)
Host name:2137(RET)

 Gus - The Languages System (2137), TOPS-20 Monitor 4(3046)
@LOG OSMAN password(RET)
 Job 20 on TTY214 25-Sep-79 10:20:58
    .
    .  (User performs functions on host 2137)
    .
@LOGO(RET)
Killed Job 20, User OSMAN, Account MONITOR, TTY 214
  at 25-Sep-79 10:21:30, Used 0:00:02 in 0:00:31

(User enters CTRL/B.  It does not echo.)

[Connection broken, back at node 2102,
Type CONTINUE to resume connection]
@LOGO(RET)
Killed Job 42, User KAMANITZ, Account 341, TTY 217
  at 25-Sep-79 10:23:18, Used 00:00:01 in 0:05:00

(User enters CTRL/Y. It does not echo.)

[Connection broken, back at node 4097,
Type CONTINUE to resume connection]
@LOGO(RET)
Killed Job 7, User CRUGNOLA, Account 341, TTY 33,
  at 25-Sep-79 10:26:31, Used 0:00:08 in 0:07:26
```

```
        TITLE   NRT20 - Program to permit logins to remote network host


;COPYRIGHT (C) 1979 BY
;DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
;
;
;THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
;ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH LICENSE AND WITH THE
;INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY  OTHER
;COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
;OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE  SOFTWARE  IS  HEREBY
;TRANSFERRED.
;
;
;THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT  NOTICE
;AND  SHOULD  NOT  BE  CONSTRUED  AS A COMMITMENT BY DIGITAL EQUIPMENT
;CORPORATION.
;
;DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY  OF  ITS
;SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.


        SALL                    ; Keep listing uncluttered, no macro expansion
        SEARCH  MONSYM,MACSYM,CMD ; Universals to search
        .REQUIRE SYS:MACREL,SYS:CMD ; Make LINK load these automatically

        SUBTTL  DECLARED SYMBOLS

        T1==1                   ; AC definitions
        T2==2
        T3==3
        T4==4
        CX==16
        P==17

        NPDL==20                ; Size of pushdown list
        INSIZ==50               ; Terminal input buffer size, in words
        MAXINP==INSIZ*5         ; Maximum number of terminal input characters
        DBUFFR==100000          ; Data buffer on page 100
```

```
        SUBTTL  IMPURE STORAGE

STIMSK: BLOCK 1                 ; Terminal interrupt word mask for STIW
LOCAL:  BLOCK 2                 ; Local node name
DCNJFN: BLOCK 1                 ; JFN of logical link to NRTSRV server
INSTR:  BLOCK INSIZ             ; Terminal input stream buffer
SAVACE: BLOCK 4                 ; Register save area for AC's 1 - 4
NAME:   BLOCK 10                ; File spec name of logical link to NRTSRV
FORK:   BLOCK 1                 ; Handle of inferior process to read from TTY
VIRGIN: EXP -1                  ; Flag to indicate fresh start, initially -1.
PC:     BLOCK 3                 ; Storage for PC on interrupts
LEVTAB: PC                      ; Software interrupt level table
        PC+1
        PC+2
CHNTAB: 0                       ; Software interrupt channel table
        1,,ESC                  ; Panic character (default ^Y)
        0
PDL:    BLOCK NPDL              ; Pushdown list (STACK)

CMDSTG                          ; Command parser storage (from CMDS)
```

```
        SUBTTL  START  -  Start of NRT20 program

START:  RESET                       ; Intialize ourself
        MOVE P,[IOWD NPDL,PDL]      ; and set up pushdown list
        AOS VIRGIN                  ; Bump count of times we come through here
        CALL CMDINI                 ; Initialize command parser
        MOVX T1,CR%MAP              ; Create process to read from TTY
        SETZM T2
        CFORK                       ; Make the process, same address map
         JRST [ JSHLT                     ; on error... halt and allow continue
                JRST START]
        MOVEM T1,FORK               ; Save the process handle obtained
        MOVEI T1,.FHSLF             ; Our process handle
        RPCAP                       ; Get our capabilities
        TXO T3,SC%CTC               ; allow ^C trapping for job
        EPCAP                       ; Enable new set of capabilities
        MOVE T2,[LEVTAB,,CHNTAB]    ; Address of software interrupt system tables
        SIR                         ; Give them to monitor
        MOVX T2,1B1                 ; Channel number
        AIC                         ; Activate software interrupt channels
        EIR                         ; Enable software interrupt system
        CALL ALLON                  ; Turn on all TTY interrupts
        CALL PRESCN                 ; Prescan EXEC line for host name...
         JRST PREHST                ; Got it - skip the prompting
        JRST ESCHAR                 ; Otherwise get it now
```

```
        SUBTTL  ESCHAR - Obtain the escape character user wants to use

ESCHAR: PROMPT <Escape character(^Y):> ; Put out prompt to users terminal
        MOVEI T1,[FLDDB. (.CMTXT,CM%SDH,,<Character for interrupting connection>,<^Y>)]
        CALL CFIELD             ; Read the escape character
        LDB T1,[POINT 7,ATMBUF,6] ; Get character itself
        CAILE T1,.TICCZ         ; Is it a valid character? (^A to ^Z)
         JRST [ HRROI T1,[ASCIZ /Invalid character/] ; No, tell user
                ESOUT
                JRST ESCHAR]    ; and try again
        MOVNI T2,0(T1)          ; Save the character
        HRLS T1                 ; Character value is terminal code
        HRRI T1,1               ; Channel number
        ATI                     ; Assign terminal code to software interrupt channel
        MOVX T1,1B0             ; Get a bit
        LSH T1,0(T2)            ; Make terminal interrupt word mask for character
        TXO T1,1B<.TICTI>       ; Allow typein
        MOVEM T1,STIMSK         ; Save for use later
        JRST HSTNAM             ; Get the host name
```

```
        SUBTTL  HSTNAM - Get the host name user wants to login on

HSTNAM: PROMPT <Host name: >     ; Put out prompt to users terminal
PREHST: MOVEI T1,[FLDDB. (.CMTXT,CM%SDH,,<Name of system to connect to>)]
        CALL CFIELD              ; Read host name
        HRROI T1,INSTR           ; Pointer to destination buffer
        HRROI T2,ATMBUF          ; Pointer to source buffer
        MOVEI T3,0               ; An ASCIZ string
        SOUT                     ; Copy the host name
        MOVX T1,177B6            ; See if real host name given
        TDNN T1,INSTR            ; Was one?
        JRST [  HRROI T1,[ASCIZ /
[CONNECTING TO LOCAL HOST- /]            ; No, tell user what's happening
                PSOUT
                MOVEI T1,.NDGLN          ; Function code to get local node name
                MOVEI T2,T3
                HRROI T3,INSTR
                NODE                     ; Obtain local node name
                HRROI T1,INSTR
                PSOUT                    ; Output to user's TTY
                HRROI T1,[ASCIZ /]
/]
                PSOUT                    ; Finish the message nicely
                JRST .+1]
        JRST SETTMD              ; Have name. Now set TTY modes
```

```
        SUBTTL  SETTMD - Set TTY modes

SETTMD: MOVEI T1,.PRIIN         ; Primary input
        RFMOD                   ; Get current JFN mode word
        TXO T2,TT%WAK           ; Wake on all
        TXZ T2,TT%ECO!TT%DAM    ; Echoes off and binary data mode
        SFMOD                   ; Set new JFN mode word
        MOVX T1,.FHJOB          ; Set terminal interrupt word for entire job
        MOVE T2,STIMSK          ; Get terminal interrupt word saved earlier
        STIW                    ; Set it to allow only panic interrupts
        HRROI T1,NAME           ; Build network file spec to NRTSRV
        HRROI T2,[ASCIZ /DCN:/] ; The DECnet device for making connection
        SETZM T3
        SOUT
        HRROI T2,INSTR          ; The node name to connect to
        SOUT
        HRROI T2,[ASCIZ /-200/] ; NRTSRV's object type
        SOUT
        MOVX T1,GJ%SHT          ; Do a short form GTJFN
        HRROI T2,NAME           ; Using the spec just built
        GTJFN                   ; Get DCN connection
         JRST [ JSERR                   ; If error, tell user
                CALL ALLON              ; Reset TTY to normal mode
                JRST HSTNAM]            ; And go back to get host name
        MOVEM T1,DCNJFN         ; Save JFN for the connection
        MOVEI T1,.PRIOU         ; Check if parity OK on this TTY
        GDSTS                   ; Get device status
        ERJMP [SETZM T2                 ; If error, set all status off
               JRST .+1]                ; And continue along
        MOVE T1,DCNJFN          ; Restore the JFN for network connection
        TXNE T2,GD%PAR          ; Does terminal allow parity?
        SKIPA T2,[FLD(^D8,OF%BSZ)!OF%RD!OF%WR] ; Yes, 8 bit bytes and read/write access
        MOVE T2,[FLD(^D7,OF%BSZ)!OF%RD!OF%WR] ; No, then 7 bit bytes and read/write access
        OPENF                   ; Open the network connection
         JRST [ JSERR                   ; If failed, tell user
                MOVE T1,DCNJFN          ; And release the JFN
                RLJFN
                 JFCL
                CALL ALLON              ; Reset TTY to normal mode
                JRST HSTNAM]            ; And go back to get host name
        MOVE T1,FORK            ; Restore the process handle obtained
        MOVEI T2,DOINP0         ; The start address
        SFORK                   ; Start the process to read input from TTY
         ERJMP [RESET                   ; If failed, reset ourself
                JSHLT                   ; Tell user and halt
                JRST START]             ; Go back to beginning if continued
        JRST GOTDA1             ; Connection now opened. Wait for input
```

```
        SUBTTL  GOTDA1 - Read data from network connection

GOTDA1: MOVE T1,DCNJFN          ; Get the network connection JFN
        MOVE T2,[POINT ^D8,DBUFFR] ; Pointer to buffer
        MOVNI T3,1              ; One byte only
        SIN                     ; Read from logical link
         ERJMP BADCON                   ; If error, connection broken
        MOVE T3,T2              ; Save byte pointer
        SIBE                    ; Any more data to read now ?
         SKIPA                          ; Yes, read the rest
        JRST [  SETZM T4        ; No, then go type what there is to user
                JRST GOTDAX]
        EXCH T2,T3              ; Restore byte pointer
        MOVE T4,T3              ; Save count
        MOVNS T3                ; Read the exact number available
        SIN                     ; Read remaining bytes
         ERJMP BADCON                   ; If error, connection broken

GOTDAX: MOVEI T1,.PRIOU         ; Output to users terminal
        MOVE T2,[POINT ^D8,DBUFFR] ; Pointer to data from remote host
        MOVNI T3,1(T4)          ; Output count
        SOUT                    ; Type data on users terminal
        JRST GOTDA1             ; And continue
```

```
        SUBTTL  ALLON - Restore echoing and interrupts on terminal

ALLON:  MOVX T1,.FHJOB          ; Set terminal interrupt word for entire job
        SETOM T2                ; Terminal interrupt word mask for all codes
        STIW                    ; Set terminal interrupt word
        MOVEI T1,.PRIIN         ; Primary input
        RFMOD                   ; Read JFN mode word
        TXO T2,TT%ECO!TT%DAM    ; Echo on, ASCII data mode, output translation disabled
        SFMOD                   ; Set new JFN mode word
        RET                     ; Done, return to caller
```

```
        SUBTTL  PRESCN - Prescan EXEC command for host name

PRESCN: SKIPE   VIRGIN          ; Starting out fresh?
        RETSKP                  ; No..., so return +2 to force prompt
        MOVEI   T1,.RSINI       ; Make rescan buffer available
        RSCAN                   ; as command line to process reading from CTTY
         ERJMP PRESC1
        MOVX    T1,.RSCNT       ; Get count of characters
        RSCAN                   ; in rescan buffer
         ERJMP PRESC1
        JUMPE   T1,[RETSKP]     ; None there, so must prompt
        MOVEI   T1,.PRIIN       ; Primary input
        BIN                     ; Read a character
        CAIE    T2,"R"          ; RUN or R (program name can't start with "R")
        CAIN    T2,"r"          ; Check lowercase also
        JRST PRESC0             ; Yes, no prescanning, go eat rest of line
PRES1:  CAIE    T2," "          ; Have we hit a space yet?
        CAIN    T2,"    "       ; Also accept a tab
        JRST    PRES2           ; Yes, read the node name
        CAIN    T2,.CHLFD       ; Have we hit EOL yet?
        RETSKP                  ; Yes, no node name, skip return to prompt
        BIN                     ; Read the next character
        JRST    PRES1           ; And loop to test
PRES2:  MOVSI   T1,.TICCY       ; Default to ^Y for interrupt character
        HRRI    T1,1            ; And use channel one
        ATI                     ; Assign terminal code to interrupt channel
        MOVX    T1,1B0          ; Get a hi order bit
        LSH     T1,-.TICCY      ; Shift bit by value of ^Y code
        TXO     T1,1B<.TICTI>   ; Allow typein
        MOVEM   T1,STIMSK       ; Save for use later
        TMSG    <[Type ^Y to return to node > ; Tell user how to get back
        MOVX T1,.NDGLN
        MOVX T2,T3
        MOVE T3,[POINT 7,LOCAL]
        NODE                    ; And name of local node
         JFCL
        HRROI T1,LOCAL
        PSOUT
        TMSG    <]
>                               ; Finish off nicely
        PROMPT  <>              ; Null prompt
        RET                     ; Done, return no skip to not prompt

;HERE TO FLUSH REST OF EXEC COMMAND LINE

PRESC0: BIN                     ; Get next byte
        CAIE T2,.CHLFD          ; The end of line?
        JRST PRESC0             ; No, Try again
        RETSKP                  ; Yes, return +2 to prompt for node

; Here if rescan attempt does not succeed

PRESC1: HRROI T1,[ASCIZ/Unexpected error in scanning command line/]; If error,
        ESOUT                   ; tell user
        HALTF                   ; and halt
        JRST START              ; Start over if continued...
```

```
        SUBTTL BADCON - Handle a broken network connection

BADCON: CIS                         ; Prevent further interruption
        CALL ALLON                  ; Restore TTY modes
        MOVE T1,FORK                ; Restore the process handle
        KFORK                       ; Kill the terminal input process
        HRROI T1,[ASCIZ /
?Connection broken. Reason: /]      ; Tell user bad news
        PSOUT
        MOVE T1,DCNJFN              ; Restore network connection JFN
        MOVEI T2,.MORLS             ; Get current link status
        MTOPR
        HRRZ T2,T3                  ; Get DECnet error reason code
        MOVEI T1,.PRIOU             ; Output to users terminal
        MOVEI T3,12                 ; In decimal
        NOUT                        ; Type it
         JFCL
        CAIG T2,MAXMSG              ; Know about this error?
        SKIPN T1,MSGTBL(T2)         ; Yes, have one to type?
        SKIPA                       ; No
        PSOUT                       ; Yes, type it then
        HRROI T1,[ASCIZ /
/]
        PSOUT                       ; Pretty it
        MOVE T1,DCNJFN              ; Get the JFN for the link
        TXO T1,CZ%ABT               ; Abort any operations
        CLOSF                       ; Close it
        HALTF                       ; Halt here
        JRST START                  ; And start at beginning if continued
```

```
        SUBTTL  ESC - Handle input of panic escape character from terminal


ESC:    DMOVEM T1,SAVACE        ; Save AC's we're going to use
        DMOVEM T3,SAVACE+2
        MOVE T1,FORK            ; Restore process handle
        FFORK                   ; Freeze the terminal input process
        MOVEI T1,.PRIOU         ; Users terminal
        RFMOD                   ; Get current TTY modes
        PUSH P,T2               ; Save them
        TXO T2,TT%DAM           ; ASCII data mode, output translation disabled
        SFMOD                   ; Set new JFN mode word
        CFOBF                   ; Flush any pending terminal output
        HRROI T1,[ASCIZ /
[Connection broken, back at node /] ; Tell user he's now talking to us
        PSOUT
        MOVX T1,.NDGLN
        MOVX T2,T3
        MOVE T3,[POINT 7,LOCAL]
        NODE                    ; Get our node name
         JFCL
        HRROI T1,LOCAL
        PSOUT                   ; Type node name to user
        HRROI T1,[ASCIZ/,
Type CONTINUE to resume connection]
/]                              ; Let him know how to restart
        PSOUT
        HALTF                   ; Halt here until continued
        HRROI T1,[ASCIZ/%Reconnecting to remote node...
/]                              ; Let user know we are restarting
        PSOUT
        MOVEI T1,.PRIOU         ; Users terminal
        POP P,T2                ; Desired TTY modes
        SFMOD                   ; Set new JFN mode word
        MOVE T1,FORK            ; Restore process handle
        RFORK                   ; Resume the terminal input process
        DMOVE T1,SAVACE         ; Restore the AC's
        DMOVE T3,SAVACE+2
        DEBRK                   ; And finish interrupt processing
```

```
        SUBTTL  DOINP0 - Read terminal input and send to remote host

DOINP0: MOVEI T4,MAXINP          ; Maximum input characters
        MOVE T3,[POINT 7,INSTR]  ; Where they are going
        MOVEI T1,.PRIIN          ; Primary input
        BIN                      ; Wait for a byte
        JRST DOIN0               ; Got one, go handle it...

DOINP:  SIBE                     ; Any bytes?
         SKIPA                   ; Yes
        JRST EMPTY               ; No
        BIN                      ; Get another byte
DOIN0:  IDPB T2,T3               ; Stash it in input stream buffer
        SOJG T4,DOINP            ; Do it all
EMPTY:  MOVNI T3,MAXINP          ; Maximum characters
        ADD T3,T4                ; Compute number to send
        JUMPE T3,DOINP0          ; If none, go wait
        MOVE T1,DCNJFN           ; The network JFN
        HRROI T2,INSTR           ; Pointer to input stream buffer
        SOUTR                    ; Send this buffer to remote NRTSRV
        JRST DOINP0              ; And continue
```

```
        SUBTTL MSGTBL - Table of DECnet disconnect/abort reason text

MSGTBL: 0
        0
        -1,,[ASCIZ /: Destination node does not exist/]
        0
        -1,,[ASCIZ /: Destination process does not exist/]
        0
        0
        0
        0
        -1,,[ASCIZ /: Connection aborted by remote node/]
        BLOCK ^D22
        -1,,[ASCIZ /: No more NRTs at remote node/]
        -1,,[ASCIZ /: No more NRTs at remote node/]
        BLOCK 4
        -1,,[ASCIZ /: Remote process aborted link/]
        -1,,[ASCIZ /: Network failure. No path to remote node/]
MAXMSG==.-MSGTBL-1

        END START
```

```
        TITLE   NRTSRV - Program to provide remote network login service


;COPYRIGHT (C) 1979 BY
;DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
;
;
;THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
;ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH LICENSE AND WITH THE
;INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY  OTHER
;COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
;OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE  SOFTWARE  IS  HEREBY
;TRANSFERRED.
;
;
;THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT  NOTICE
;AND  SHOULD  NOT  BE  CONSTRUED  AS A COMMITMENT BY DIGITAL EQUIPMENT
;CORPORATION.
;
;DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY  OF  ITS
;SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.


        SALL                    ; Keep listing uncluttered, no macro expansion
        SEARCH  MONSYM,MACSYM   ; Universals to search
        .REQUIRE SYS:MACREL     ; Make LINK load these automatically


        SUBTTL  DECLARED SYMBOLS

        T1==1                   ; AC definitions
        T2==2
        T3==3
        T4==4
        CX==16
        P==17

        MAXFRK==7               ; Maximum number of server sub processes
        SRVOSZ==50              ; Output buffer size in words
        MAXOUT==SRVOSZ*4        ; Maximum number of output characters
        SRVSIZ==10              ; Size of server buffer in words
        NPDL==40                ; Size of pushdown list
```

```
        SUBTTL  IMPURE STORAGE

SAVAC:  BLOCK 4                 ; AC save area
DEVNAM: BLOCK 2                 ; For building PTY names
PTYJFN: BLOCK 1                 ; The JFN of the PTY
TTYJFN: BLOCK 1                 ; TTY designator
SRVJFN: BLOCK 1                 ; The JFN of the network connection
SRVINP: BLOCK SRVSIZ            ; Holds input characters
OUTPUT: BLOCK 1                 ; Output buffer pointer
OUCNT:  BLOCK 1
PC:     BLOCK 1                 ; The software interrupt PC save area
SRVOUP: BLOCK SRVOSZ            ; Holds output data
FORKS:  BLOCK MAXFRK            ; Save process ID's here
HSTNAM: BLOCK 2                 ; Host name string
PDL:    BLOCK NPDL              ; Pushdown list (stack)

        RELOC 1000-140          ; Start on page boundary

LEVTAB: PC                      ; Where to stash the PC
CHNTAB: 1,,GOTCI                ; Network connect received interrupt routine
        1,,GOTDAT               ; Network data available interrupt routine
```

```
        SUBTTL  BEGIN - Start of NRTSRV program

BEGIN:  RESET                       ; Initialize ourselves
        MOVE T1,[SIXBIT /NRTSRV/]   ; Our name in SIXBIT
        SETNM                       ; Set jobs private name
        HRROI T1,[ASCIZ /SRV/]      ; See if network exists
        STDEV                       ; By checking for SRV device
         JRST [ SETOM T1                    ; Doesn't exist, so go away
                LGOUT                       ; By logging out this job
                 JFCL                       ; Should always work, but...
                HALTF]                      ; Either way, go away
        MOVEI T1,.FHSLF             ; This process
        RPCAP                       ; Get our capabilities
        MOVE T3,T2                  ; Copy them
        EPCAP                       ; Enable all capabilities we have
        MOVSI T4,-MAXFRK            ; Maximum processes to create
FRKDO:  MOVX T1,CR%CAP              ; Give them the same capabilities we have
        SETZM T2                    ; No AC's
        CFORK                       ; Create a process
         ERJMP WTFRK                        ; If no more, done
        MOVEM T1,FORKS(T4)          ; Save process handle
        HRLZ T2,T1                  ; Process handle of destination
        HRRI T2,1                   ; Start at page 1 in destination process
        MOVE T1,[.FHSLF,,1]         ; Map this process to it
        MOVEI T3,ENDPRG-1           ; Last address in use
        LSH T3,-11                  ; Make a repetition count
        TXO T3,PM%CNT!PM%RD!PM%WR   ; Make PMAP argument
        PMAP                        ; Map the process' address space
        HLRZ T1,T2                  ; Get process handle
        MOVEI T2,START              ; Start address
        SFORK                       ; Start the process up
FRKLOP: AOBJN T4,FRKDO              ; Do all of them
WTFRK:  MOVEI T1,.FHSLF             ; Our process
        SETZM T3                    ; No capabilities
        EPCAP                       ; Disable all capabilities
        MOVEI T1,.FHINF             ; All the inferior processes
        RFORK                       ; Resume them
        WFORK                       ; Wait for any inferior to halt
        ; ..
```

```
        SUBTTL  RSPROC - Restart inferior process after it halted

RSPROC: MOVEI T1,.FHINF         ; All the inferior processes
        FFORK                   ; Freeze them
        MOVSI T4,-MAXFRK        ; Scan process table
DEDLOP: SKIPN T1,FORKS(T4)      ; This one exist?
        JRST DEDLO1             ; No, go on
        RFSTS                   ; Yes, get its status
        LOAD T1,RF%STS,T1       ; Get its status
        CAIE T1,.RFHLT          ; Halted?
        CAIN T1,.RFFPT          ; Or error?
        JRST [  MOVE T1,FORKS(T4)       ; Yes, get process handle again
                SETZM FORKS(T4)         ; Clear entry
                KFORK                   ; Kill the process
                HRRZS T4                ; Create one more process
                MOVEI T1,.FHSLF         ; Our process
                RPCAP                   ; Get our capabilities
                MOVE T3,T2              ; Copy them
                EPCAP                   ; Enable all capabilities we have
                JRST FRKDO]             ; Go back and create another process
DEDLO1: AOBJN T4,DEDLOP         ; Look at them all
        JRST WTFRK              ; Wait some more
```

```
        SUBTTL  START - Start of code for inferior process'


START:  CIS                     ; Clear interrupt system
        SKIPN T1,SRVJFN         ; Have a server JFN yet?
        JRST START1             ; No...
        TXO T1,CZ%ABT           ; Yes, abort it
        CLOSF                   ; Close and abort the network connection
         JFCL                           ; Ignore errors
        SETZM SRVJFN            ; Clear the server JFN
START1: SKIPN T1,PTYJFN         ; Have a PTY yet?
        JRST START2             ; No...
        TXO T1,CZ%ABT           ; Yes, abort it
        CLOSF                   ; Close the PTY
         JFCL                           ; Ignore errors
START2: MOVE P,[IOWD NPDL,PDL]  ; Initialize the stack pointer
        MOVEI T1,.FHSLF         ; Our process
        MOVE T2,[LEVTAB,,CHNTAB] ; Address of level and channel table
        SIR                     ; Set software interrupt table addresses
        MOVX T2,3B1             ; Channel 0 and 1
        AIC                     ; Activate software interrupt channels
        EIR                     ; Enable software interrupt system
        MOVE T1,[POINT ^D8,SRVOUP] ; Initial output pointer
        MOVEM T1,OUTPUT         ; Initialize the pointer
        MOVEI T1,MAXOUT         ; Maximum output characters
        MOVEM T1,OUCNT          ; Initialize the output count
        MOVEI T1,.FHSLF         ; Our process
        RPCAP                   ; Get our capabilities
        MOVE T3,T2              ; Copy them
        EPCAP                   ; Enable all capabilities we have
        MOVX T1,GJ%SHT          ; Short form GTJFN
        HRROI T2,[ASCIZ /SRV:200/] ; The NRT server file spec
        GTJFN                   ; Get a JFN for it
         JRST TRYAGN                    ; Failed, go try to recover
        MOVEM T1,SRVJFN         ; Save the server JFN
        MOVX T2,<FLD(^D8,OF%BSZ)!OF%RD!OF%WR> ; 8 bit bytes, read/write access
        OPENF                   ; Open the server JFN
         JRST [ MOVE T1,SRVJFN          ; Failed, get server JFN
                RLJFN                   ; Release the JFN
                 JFCL                   ; Ignore errors
                SETZM SRVJFN            ; Clear the server JFN
                JRST TRYAGN]            ; And go try to recover
        MOVEI T1,.FHSLF         ; Our process
        SETZB T2,T3             ; No capabilities
        EPCAP                   ; Disable all capabilities
        MOVE T1,SRVJFN          ; Get the server JFN
        MOVEI T2,.MOACN         ; Assign interrupt system channel numbers
        MOVX T3,<FLD(0,MO%CDN)!FLD(1,MO%DAV)!FLD(.MOCIA,MO%INA)> ; Connect event on 0,
                                ; Data available on 1
        MTOPR                   ; Perform the device operation
        WAIT                    ; Wait for something to happen
```

```
        SUBTTL  TRYAGN - Wait before trying to become a server again

TRYAGN: MOVX T1,^D<1000*60*2>   ; Two minutes in milliseconds
        DISMS                   ; Dismiss the process until time expires
        JRST START              ; And go back to try again
```

```
        SUBTTL  GOTCI - Got a connect from the network


GOTCI:  CALL GETPTY             ; Get a PTY to use
        JRST [  CIS                     ; Failed, clear the software interrupt system
                HRROI T1,[ASCIZ /
?No more PTYs on remote host
/]                                      ; Point to message to tell what happened
                MOVE T2,SRVJFN          ; Get the server JFN
                SETZM T3                ; All of the string
                SOUTR                   ; Send message on network link
                 ERJMP .+1              ; Ignore errors
                MOVE T1,SRVJFN          ; Get the server JFN
                CLOSF                   ; Close the network connection
                 JRST START                     ; Failed, go back to recover
                SETZM SRVJFN            ; Clear the server JFN
                JRST START]             ; Go back to recover
        MOVEM T1,PTYJFN         ; Save the PTY JFN
        MOVEM T2,TTYJFN         ; Save the TTY designator
        MOVEI T2,"C"-100        ; Get a Control-C (^C)
        BOUT                    ; Output it to PTY to start up job
        MOVE T1,SRVJFN          ; Get the server JFN
        MOVEI T2,.MOCC          ; Confirm the network connection
        SETZB T3,T4             ; No optional data
        MTOPR                   ; Do the device operation
         ERJMP START                    ; If error, link broken, start again
        MOVEI T1,PTYIN          ; Address of PTY input routine
        MOVEM T1,PC             ; Change the interrupt PC value to start
        GJINF                   ; Get current job information
        JUMPL T4,DONCI          ; If terminal attached to job, done
        HRROI T1,[ASCIZ /
NRTSRV- CONNECTION FROM /]       ; Point to message to say who's connecting
        PSOUT                   ; Output to TTY
        SETZM HSTNAM            ; Clear host name buffer
        MOVE T1,SRVJFN          ; Get the server JFN
        MOVEI T2,.MORHN         ; Read network host name
        HRROI T3,HSTNAM         ; Pointer to buffer
        MTOPR                   ; Do the device operation
         ERJMP .+1                      ; Ignore errors
        HRROI T1,HSTNAM         ; Pointer to host name buffer
        PSOUT                   ; Output the host name to TTY
        HRROI T1,[ASCIZ / ON /] ; Pointer to continuation of message text
        PSOUT                   ; Output to TTY
        MOVEI T1,.PRIOU         ; TTY designator
        MOVE T2,PTYJFN          ; Get the PTY JFN
        SETZM T3                ; Full file specification format
        JFNS                    ; Output file spec associated with PTY JFN
        HRROI T1,[ASCIZ /
/]
        PSOUT                   ; End the message nicely
DONCI:  DEBRK                   ; Dismiss interrupt and resume processing PTY input
```

```
        SUBTTL  GOTDAT - Got data available from network connection


GOTDAT: DMOVEM T1,SAVAC         ; Save AC's
        DMOVEM T3,SAVAC+2
GOTDA1: MOVE T1,SRVJFN          ; Get the server JFN
        MOVEI T2,.MORLS         ; Read link's status
        MTOPR                   ; Do the device operation
        TXNN T3,MO%CON          ; Connected?
        JRST START              ; No, go start over
        SIBE                    ; Skip if input buffer empty
         SKIPA                  ; Not empty, have some data
        JRST [  DMOVE T1,SAVAC          ; Empty, no data, restore AC's
                DMOVE T3,SAVAC+2
                DEBRK]                  ; Dismiss interrupt and resume processing
        CAILE T2,SRVSIZ*5       ; Buffer larger than number of bytes available?
        MOVEI T2,SRVSIZ*5       ; No, get maximum buffer full
        MOVN T3,T2              ; Get byte count
        MOVE T4,T3              ; Save it
        HRROI T2,SRVINP         ; Pointer to server input buffer
        SIN                     ; Read from network link
         ERJMP START                    ; Failed, link broken, start again
        SUB T4,T3               ; Calculate number of bytes read
        MOVE T3,T4              ; Get byte count
        MOVE T1,PTYJFN          ; Get PTY JFN
        HRROI T2,SRVINP         ; Pointer to server input buffer
        SOUT                    ; Output data to PTY
        JRST GOTDA1             ; And go back to do more
```

```
        SUBTTL  PTYIN - Pseudo TTY input routine

PTYIN:  MOVE T1,PTYJFN          ; Get PTY JFN
        BIN                     ; Input a byte from PTY
        IDPB T2,OUTPUT          ; Store byte in output buffer and save pointer
        SOSG OUCNT              ; Output buffer full?
        JRST PTYIN1             ; Yes, write it then
        MOVE T1,TTYJFN          ; Get TTY designator
        SOBE                    ; Skip if TTY output buffer empty
         JRST PTYIN                     ; Not empty, go get more characters from PTY
PTYIN1: MOVNI T3,MAXOUT         ; Get negative maximum output count
        ADD T3,OUCNT            ; Compute number of bytes in buffer
        MOVE T1,SRVJFN          ; Get the server JFN
        MOVE T2,[POINT ^D8,SRVOUP] ; Get pointer to output buffer
        SOUTR                   ; Output data to network link
         ERJMP START                    ; Failed, link died, start over
        MOVE T1,[POINT ^D8,SRVOUP] ; Get pointer to output buffer
        MOVEM T1,OUTPUT         ; Reinitialize output buffer pointer
        MOVEI T2,MAXOUT         ; Get maximum output count
        MOVEM T2,OUCNT          ; Reinitialize output count
        JRST PTYIN              ; Go back and get more characters from PTY
```

```
        SUBTTL  GETPTY - Get a pseudo TTY designator and return opened PTY JFN

GETPTY: STKVAR <SAVCHR>          ; Get stack variable storage
        MOVE T1,[SIXBIT /PTYPAR/] ; Pseudo TTY parameter table name
        SYSGT                    ; Get system table information for PTY
        HRRZM T1,SAVCHR          ; Save TTY number of first pseudo TTY
        HLRZ T4,T1               ; Get number of PTY's
        MOVNS T4                 ; Get negative of count
        HRLZS T4                 ; Make an index loop pointer
GETPT1: MOVSI T1,.DVDES+.DVPTY   ; Get device designator for PTY
        HRR T1,T4                ; Get PTY number
        DVCHR                    ; Get device characteristics of PTY
        TXNN T2,1B5              ; Available?
        JRST GETPT2              ; No, try next PTY then
        MOVE T2,T1               ; Get device designator
        HRROI T1,DEVNAM          ; Pointer to device name string buffer
        DEVST                    ; Translate device designator to string
         JRST GETPT2                    ; Failed, try next PTY
        MOVEI T2,":"             ; Get device string terminator character
        IDPB T2,T1               ; Store it after device name string
        SETZM T2                 ; Get a null byte
        IDPB T2,T1               ; Make device name string ASCIZ
        MOVX T1,GJ%SHT           ; Short form GTJFN
        HRROI T2,DEVNAM          ; Get pointer to PTY name string
        GTJFN                    ; Get a JFN for PTY
         JRST GETPT2                    ; Failed, try next PTY
        MOVE T3,T1               ; Save the PTY JFN
        MOVX T2,<FLD(^D8,OF%BSZ)!OF%RD!OF%WR> ; 8 bit bytes, read/write access
        OPENF                    ; Open the PTY
         JRST [ MOVE T1,T3              ; Failed, get PTY JFN
                RLJFN                   ; Release the PTY JFN
                 JFCL                           ; Ignore errors
                JRST GETPT2]            ; And try next PTY
        MOVEI T2,.TTDES(T4)      ; Get PTY index
        ADD T2,SAVCHR            ; Get PTY designator
        RETSKP                   ; Done, return skip for success

; Step through all PTY's

GETPT2: AOBJN T4,GETPT1          ; Increment PTY number and go back if more to do
        RET                      ; No, give up, return no skip for failure

ENDPRG:                          ; End of code
        END BEGIN
```

```
        TITLE   STNRT - Program to create NRTSRV as a not logged in job


;COPYRIGHT (C) 1979 BY
;DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
;
;
;THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
;ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH LICENSE AND WITH THE
;INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY  OTHER
;COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
;OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE  SOFTWARE  IS  HEREBY
;TRANSFERRED.
;
;
;THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT  NOTICE
;AND  SHOULD  NOT  BE  CONSTRUED  AS A COMMITMENT BY DIGITAL EQUIPMENT
;CORPORATION.
;
;DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY  OF  ITS
;SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.


        SALL                    ; Keep listing uncluttered, no macro expansion
        SEARCH  MONSYM,MACSYM   ; Universals to search
        .REQUIRE SYS:MACREL     ; Make LINK load these automatically


        SUBTTL  DECLARED SYMBOLS

        T1==1                   ; AC definitions
        T2==2
        P==17

        NPDL==10                ; Size of pushdown list
```

```
        SUBTTL  IMPURE STORAGE

PDL:    BLOCK NPDL              ; Pushdown list (stack)

                                ; Argument block for CRJOB JSYS
CRJBLK: 0                       ; No user name string
        0                       ; No password string
        0                       ; No account string
        -1,,[ASCIZ /SYS:NRTSRV.EXE/] ; Pointer to file name to be placed in job
        0                       ; Entry vector offset
        377777                  ; Controlling terminal designator is null device
        0                       ; (Reserved)
        0                       ; Address of AC block
        0                       ; Command language processor flags
        0                       ; Primary I/O designators for inferior processes
        0                       ; CPU runtime limit (0 = no limit)
        0                       ; Capability mask for job
        0                       ; IPCF PID for logout message
```

```
        SUBTTL  START - Start of program

START:  RESET                   ; Initialize ourself
        MOVE P,[IOWD NPDL,PDL]  ; Initialize PDL pointer
        MOVX T1,CJ%FIL!CJ%NPW!CJ%CAP ; CRJOB flags
        MOVEI T2,CRJBLK         ; Address of argument block
        CRJOB                   ; Create the job
         JSHLT                          ; Failed
        HALTF                   ; Made it

        END START
```

APPENDIX G

TKB20 AND VNP20 WARNING AND ERROR MESSAGES

This appendix contains a list of messages that may be directed to the user's I/O device by TKB20 and/or VNP20. The messages are separated into two groups: warning messages and fatal error messages. Within each group, the messages are listed in alphabetical order.

Each warning message is preceded by a percent sign (%). TKB20 or VNP20 issues a warning message to indicate a possible problem. Processing continues; however, subsequent processing may or may not be successful. If processing terminates some time after a warning message is issued, the user should be aware of any prior warning messages. The problem highlighted in the prior warning message may be related to the termination of processing.

Fatal error messages are characterized in two ways: a fatal error message may be preceded by a question mark (?) or it may have no leading punctuation mark at all. After a fatal error message has been issued by TKB20 or VNP20, processing is always terminated. The user should then call his local Software Support Specialist, unless the corrective action is apparent from the message text.

Replaceable arguments in both warning and fatal error messages are shown as abbreviations. The following abbreviations are found in several messages:

Abbreviation	Meaning
addr	octal address
cmdline	command line
cntname	controller name
ddmname	device driver module name
dlcname	data link controller name
fn	file name
ft	file type
llcname	logical link controller name
modname	module name
objfile	object file name
psect	p-section name
rtnname	routine name (usually follows the message text and two dashes)
symname	symbol name

Certain messages contain replaceable arguments that are relevant only to the message containing the replaceable argument. In these cases, the replaceable arguments are defined immediately following the message text.

G.1 TKB20 AND VNP20 WARNING MESSAGES

% Badly formatted OBJ file objfile -- rtnname

% Checksum error in OBJ file objfile; should be mmm, was nnn -- rtnname

mmm	is the octal value of the computed checksum.
nnn	is the octal value of the checksum that was read.

% Divide by zero when resolving relocation in P-section psect -- rtnname

% Global symbol symname defined in module mod1 with value xxx and in mod2 with value yyy

mod1	is the name of the first module.
xxx	is the value of the first symbol.
mod2	is the name of the second module.
yyy	is the value of the second symbol.

% Global symbol symname is not defined in fn -- rtnname

% Global symbol symname is undefined

% Module modname multiply defines P-section psect; flags = xxx, were yyy - rtnname

xxx	is the octal value of the new flags.
yyy	is the octal value of the old flags.

% Object file feature is unsupported -- rtnname

% Only one module allowed in symbol table file fn -- rtnname

% Partition length of mmm less than task length of nnn -- rtnname

mmm	is the partition length in octal.
nnn	is the task length in octal.

% Relocation value xxx truncated to 8 bits in P-section psect -- rtnname

xxx	is the octal value that was truncated.

% Unspecified output file: "fn", ignored - rtnname

G.2 TKB20 AND VNP20 FATAL ERROR MESSAGES

Some fatal error messages may be issued by more than one routine or module. These messages are identified in this appendix by the "rtnname" replaceable argument following the message text. Other fatal error messages are issued by only one routine from a specific module. In this latter type of fatal error message, the actual routine name is shown at the end of the message text.

Each fatal error message beginning with ? is followed by the message:

?Terminating - No recovery Available for Previous Error.

ADDRESS OUT OF RANGE - GET1l

ADDRESS OUT OF RANGE - STO1l

Attempting to OPEN an already OPEN channel - OPEN

BACKSLASH MISSING - VRBL_SYM

BAD PSECT POINTER - PRC_RELOC

BAD PSECT POINTER - STO_TEXT

CALL TO GETSTG BEFORE INISTG

? Cannot find nnn bytes in system pool in fn -- rtnname

nnn	is the number of decimal bytes requested.

Channel is not OPEN - CLOSE

Channel is not OPEN - INPUT

Channel number out of range - INPUT

Channel number out of range - OUTPUT

? Complex relocation exceeded stack limit of nnn, file fn -- rtnname

nnn	is the stack limit in decimal.

? DCP dlcname is unknown -- rtnname

? DDM ddmname is unknown -- rtnname

EOF IGNORED IN SCAN

? Error in text file fn -- rtnname

? Failure freeing nnn bytes at addr in system pool in fn, code errcode -- rtnname

nnn	is the number of octal bytes in the block in question.
errcode	is the decimal error code.

? File fn size of xxx bytes is larger than max of yyy - rtnname

xxx	is the file's size in decimal.
yyy	is the maximum permissible file size in decimal.

? Illegal error

This message indicates that the error number is out of the range of the defined error numbers. The "Terminating....." message does not follow this message, as it does for all other error messages preceded by a question mark.

? Indirect files nested too deep: "cmdline" -- rtnname

? Input error or unexpected EOF -- rtnname

? Invalid address addr in file fn -- rtnname

? Invalid Address Match for KDPLN For CNT$DF xxx -- D_KDP

xxx is a controller number.

? Invalid Address Match Starting KDZLN For CNT$DF xxx -- D_KDZ

xxx is a controller number.

? Invalid CNT$DF for DDM ddmname -- LD_DDM

Invalid Channel Number - CLOSE

Invalid Channel Number - OPEN

? Invalid contype - xxx found -- rtnname

contype is the type of construct that was found to be invalid.
xxx is the decimal value of the invalid construct.

? Invalid error

This message indicates that the error message has no text.

? Invalid file name: "fn" -- rtnname

? Invalid Interrupt Setup for driver in file fn -- LOADD

? Invalid UNT$DF for DDM ddmname and CNT$DF cntname -- LD_DDM

? I/O DATA ERROR on fn, status = stat -- rtnname

stat is the octal status of the I/O data error.

? LLC llcname is unknown -- rtnname

LLC NOT FOUND - LD_LLC

? Loading library file libname overflowed network pool by nnn bytes -- rtnname

nnn is the decimal number of bytes that overflowed the pool.

? LOOKUP/ENTER UUO failed for fn, code = errcode -- rtnname

errcode is the octal error code.

May not OPEN channel 0 - OPEN

MISSING CONTROLLER - LD_DLC

MISSING SLT - D_DCP

NO DEVICE CO0: - MOUACP

NO DEVICE LB - INSTALL

NO GEN PARTITION - INSTALL

? No input files -- rtnname

? OPEN UUO failed for filename -- rtnname

? Program logical address space (PLAS) is not supported -- rtnname

NOT ENOUGH STORAGE FOR INITIALIZATION - INISTG

? Second ft file in command string: "fn" -- rtnname

? Storage exhausted -- rtnname

? Switches must be associated with a file -- rtnname

? Symbol symname has conflicting values in symtab1 and symtab2 -- rtnname

symtab1	is the first symbol table file.
symtab2	is the second symbol table file.

? Syntax error in command line: "cmdline" -- rtnname

TCB LINKS BAD - INSTALL

? The Communications Executive has already been loaded into kernfile -- rtnname

kernfile	is the name of the kernel file

? The top of the COMM EXEC is too high: ceaddr .GT. maxaddr in fn -- rtnname

ceaddr	is the address of the top of the communications executive.
maxaddr	is the maximum permitted address of the RSX11S executive.

UNIT MISSING - LD_DLC

You May Not Close Channel 0 - CLOSE

? 22-bit memory management not supported in fn - rtnname

INDEX

TOPS-20 DECnet-20
Programmer's Guide and
Operations Manual
AA-5091B-TM

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____ Telephone_____

Street_____

City_____ State_____ Zip Code_____
or
Country

Please cut along this line.

Do Not Tear – Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MR1-2/E37
MARLBOROUGH, MASSACHUSETTS 01752

Do Not Tear – Fold Here and Tape

Cut Along Dotted Line