EY-2472E-ID-0001

ETHERNET ROUTER SERVER

INFORMATION DOCUMENT

SOFTWARE SERVICES TRAINING

EY-2472E-ID-0001

ETHERNET ROUTER SERVER

INFORMATION DOCUMENT

====== S O F T W A R E S E R V I C E S T R A I N I N G ======

TITLE	Ethernet Router Server
LENGTH	N/A
COURSE CODE	EY-2472E-PS-0001
TYPE	Information Document

TARGET AUDIENCE

This document is geared to specialists who provide telephone backup support, consulting services, and packaged services for the Digital Ethernet Communications Server (DECSA) and its software products. It is intended for specialists who are familiar with the DNA architecture and DECnet software on VMS and/or RSX.

PREREQUISITES

The ability to:

* Install, generate, and test DECnet on VMS and RSX
* Modify and update the configuration database on VMS and RSX
* Perform network monitoring functions using loopback and connectivity tests
* Name the major components of the DECnet software on VMS and RSX
* Demonstrate how these components are connected

Courses fulfilling these prerequisites:

- Network Concepts (SPI)
- Digital Network Architecture Phase IV (SPI)
- Communications Interfaces and Modems
- DECnet RSX and/or DECnet/VAX

COURSE DESCRIPTION

This document presents a thorough study of the DECSA Server software, including Communications Server Base and the DECnet Router Server software product. DECnet Router Server product fucntions, installation, internal data structures, data flow, and troubleshooting techniques will be discussed.

COURSE OBJECTIVES

After studying this document, the specialist will be able to:

* Explain the functions of the Communications Server Base and its layered products
* Use network management utilities as they relate to the Communications Server to:
  - Retrieve event logging information
  - Monitor error counters
  - Run loadable diagnostic image
  - Run on-line diagnostics to isolate harware problems
* Install and run installation checkout test procedures for the Decnet Router Server software product
* Use internal data structures to trace the flow of data through the Communcications Server base as well as the DECnet Router Server
* Use the DECSA on-line debugging tool to analyze server dumps and troubleshoot running server software

COURSE OUTLINE

I. Overview
   a. Communications Server and layered products
   b. Additional network capabilities

II. Installation and checkout procedures
   a. Distribution kit
   b. Loading software into the Communications Server
   c. Node initialization
   d. Installation certification procedures

III. Internals
   a. Server-base components
   b. Server-base data structures
   c. General data flow
   d. Router server components and data structures

IV. Node troubleshooting
   a. Diagnostics
   b. Network management facilities
   c. The DECSA on-line debugging tool

V. DECnet Router V1.Ø SPD (draft)

CONTENTS

OVERVIEW

FIGURES

TABLES

OVERVIEW

OVERVIEW

INTRODUCTION

This module presents the major features of and some key concepts for the DECnet Router product. The DECnet Router is one of a set of software products that runs on the DIGITAL Communications Server. These products, called Communications Servers, implement DECnet Phase IV and are used in Local Area Network (LAN) environments to provide a comprehensive networking service. These products allow shared costs, resources, and communications within both LANs and the more conventional Wide-area networks. The servers are connected directly to an Ethernet LAN and function as nodes on the network. These products allow nodes on and off the Ethernet to communicate:

- With other nodes on the LAN
- With nodes on other LANs
- With remote terminals
- Between an LAN and a conventional DECnet network
- Between a DIGITAL LAN and non-DIGITAL systems or networks

The server products perform many communications functions normally attributed to full-function DECnet nodes. This releases valuable resources from these nodes for other uses. The communications server products and their functions are:

- Terminal Server - connects a cluster of terminals to the LAN. Each terminal has access to nodes on the network that support terminal servers.
- DECnet Router Server - performs routing functions allowing communication between the LAN and remote DECnet nodes.
- DECnet Router/X.25 Gateway - provides the services of the Router Server with protocols that conform to CCITT X.25 and X.29 recommendations. These protocols allow communications between an Ethernet node and another DIGITAL or non-DIGITAL system through facilities provided by a Public Packet Switching Network (PPSN).

Using the communications server products, every node or terminal connected to the Ethernet has access to all network resources.

REFERENCES

Intoduction to Local Area Networks	EB-22714-18
Ethernet Communications Server Operations and Maintenance Manual	EK-DECSA-OP-PRE
The Ethernet A Local Area Network Data Link Layer and Physical Layer Specifications	AA-K759A-TK
Network Presales Handbook Vol. 2	EY-1180E-ID-0202
DECnet Router Server Installation and Operation Guide	AA-X019A-TK
DECnet Router Server SPD	# 30.34.00

OVERVIEW

LAN CONCEPTS

A LAN is a high-speed data communications network that serves a single building or a limited geographical area, such as an industrial complex, hospital, or college campus.

Design Goals

- High speed and bandwidth - LAN channels are designed to handle fast 'bursty' types of messages. This type of data usually originates within one office and is usually valuable only to people working in the same general area.
- Reliable and maintainable components - Node and network interfaces, transmission media, etc. are designed so that failure of a component or node disables only that unit and does not disrupt the rest of the network.
- Low cost - LANs allow optimal use of equipment by sharing resources among many users.
- Compatibility - A large number and variety of equipment is able to exchange data on the network (computers, word processors, intelligent copiers, high speed printers, etc.).
- Flexible and extendable - There is minimal disruption in the operation of the network caused by moving devices or changing the shape of the network.

In general, LANs trade distance for greater data speed, response time, and lowered costs.

ETHERNET CONCEPTS

DIGITAL uses the Ethernet LAN technology to handle high-speed local area communications. Ethernet provides a common path over which nodes can communicate or share resources. These nodes can be connected to the Ethernet either directly or remotely using communications servers.

OVERVIEW

Ethernet Advantages

- Simplified network design - A simple set of design rules exists for an Ethernet.
  - A single cable segment cannot exceed 500 meters (1640 ft) in length.
  - No more than 100 transceivers can be connected per cable segment.
  - Transceivers must be installed in 2.5 meter (8 ft) increments (indicated by markings on the coaxial cable).
  - No more than two repeaters can be placed between any two nodes.
  - The maximum distance between any two nodes is 2800 meters (9,184 ft).
  - The network cannot exceed 1023 nodes.
  - Transceiver cables cannot exceed 50 meters (164 ft).

- Simple installation - Ethernet can be brought up one node at a time. A new node can communicate immediately with all active nodes on the network.

- Reduced wiring - Ethernet has a single network cable that replaces the many interconnecting cables found in traditional networks.

- Flexibility - Additional devices or cable segments can be added or subtracted without interfering with the rest of the network.

- Reliability
  - There is no master-slave relationship or any need for routing nodes. (Routing nodes are highly recommended.)
  - Critical components are designed with triple redundant circuitry to prevent any single failure from disabling the entire Ethernet network.

- High speed
  - Ethernet provides a 10-Mbps signaling rate.
  - The maximum attainable rate between systems is 1-Mbps.
  - The limiting factor is the DEUNA.

- Interconnection with other vendors' equipment - It is possible to connect to other vendors' equipment at the data link level or through X.25 Gateways.

Ethernet Nodes

- Host Node - a full function DECnet Phase IV node.
- Loading Host - a host used to down-line load the server software across the Ethernet.
- Maintenance Host - a node with event logging enabled to accept events from a server node. The maintenance host is also the node that down-line loads or accepts up-line dumps for the server node.

NOTE

The maintenance host is usually the host that initially loads the server. If the server should be reloaded by another host, the new loading host normally becomes the maintenance host. One node may be set up to be the maintenance host if all the Ethernet hosts recognize it in their database.

- Backup Host - alternative hosts set up to serve as maintenance hosts whenever the initial maintenance host is not available.
- Routing Node - a node that keeps an updated database on the state of the other nodes in the network. Depending on the network configuration, these nodes are not restricted to the local Ethernet.
- Designated Router - the router with the highest priority. End nodes depend on the Designated Router to help route messages to a destination.
- End Node - a node that keeps addresses of some Ethernet nodes in "cache", does not know anything about the state of the other nodes in the network. End nodes use the Designated Router, if the Ethernet has one, to get to other Ethernet nodes or to nodes outside of the Ethernet.

OVERVIEW

Ethernet Addressing

A node on the Ethernet can address either a specific node or a group of nodes within the same Ethernet environment. Addressing a group of nodes involves the use of a multicast address. The message is sent, simultaneously in one transmission, to a specific group of nodes identified by the multicast address. Addressing a single Ethernet node is done by using either the Ethernet Hardware Address or the Extended-DECnet Node Address. The Ethernet address types are:

- Ethernet hardware address - a unique address that is permanently associated (in ROM) with each DEUNA controller on the Ethernet. It is made up of 12 hexadecimal digits, represented in six pairs separated by hyphens. This address is displayed by the server hardware during the system loading procedure.

  AA-00-03-00-01-23

- Extended-DECnet node address - this address is built by the DEUNA controller software when DECnet is loaded into the Ethernet node. It is made up of an 8 hexadecimal digit constant that has been assigned to DIGITAL, with the node's DECnet address appended to it.

  AA-00-04-00 (constant)

  04-3B (DECnet node address)

  AA-00-04-00-3B-04 (extended-DECnet node address)

- Ethernet physical address - the address that the DEUNA is currently responding to. Initially this is the Hardware Address. Once DECnet is loaded, the Physical Address becomes the Extended-DECnet Node Address. The Hardware Address is not used again unless the node is powered down.

- Multicast address - a multidestination address of one or more nodes on a given Ethernet. Used to simultaneously address:
  - All Ethernet routers
  - Backup hosts

Ethernet Layers

The Ethernet specification defines the two lowest layers of the DNA, the Data Link and Physical layers. The rest of the DNA or the layers above Data Link are defined as the Client Layer. The Client Layer is responsible for:

- Error recovery
- Flow control
- Internet communications

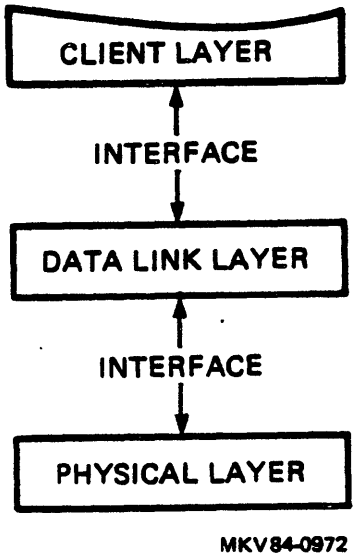


Figure 1 Ethernet Architectural Layering

Data Link Layer

The Data Link Layer defines a medium-independent link level communications facility.

- o CSMA/CD - Implements Carrier Sense Multiple Access with Collision Detection channel access.
  - There are no transmissions if another host is transmitting (check for carrier signal -- carrier sense).
  - All nodes have an equal opportunity to use the channel (multiple access).
  - When transmitting, check whether other stations are transmitting at the same time (collision detect).
  - If a collision occurs, retransmit after a random period of time.

- o Data encapsulation - like all Data Link Layers, it is responsible for:
  - Framing (frame boundaries)
  - Addressing (source and destination)
  - Error detection (physical channel errors only)

- o Link management
  - Channel allocation
  - Contention resolution (collision handling)

Physical Layer

- o Provides a 10-Mbps physical channel through a coaxial cable
- o Insulates the Data Link Layer from the physical aspects of the cable
- o Specifies physical characteristics of the Ethernet
  - Data encoding
  - Timing
  - Voltage levels

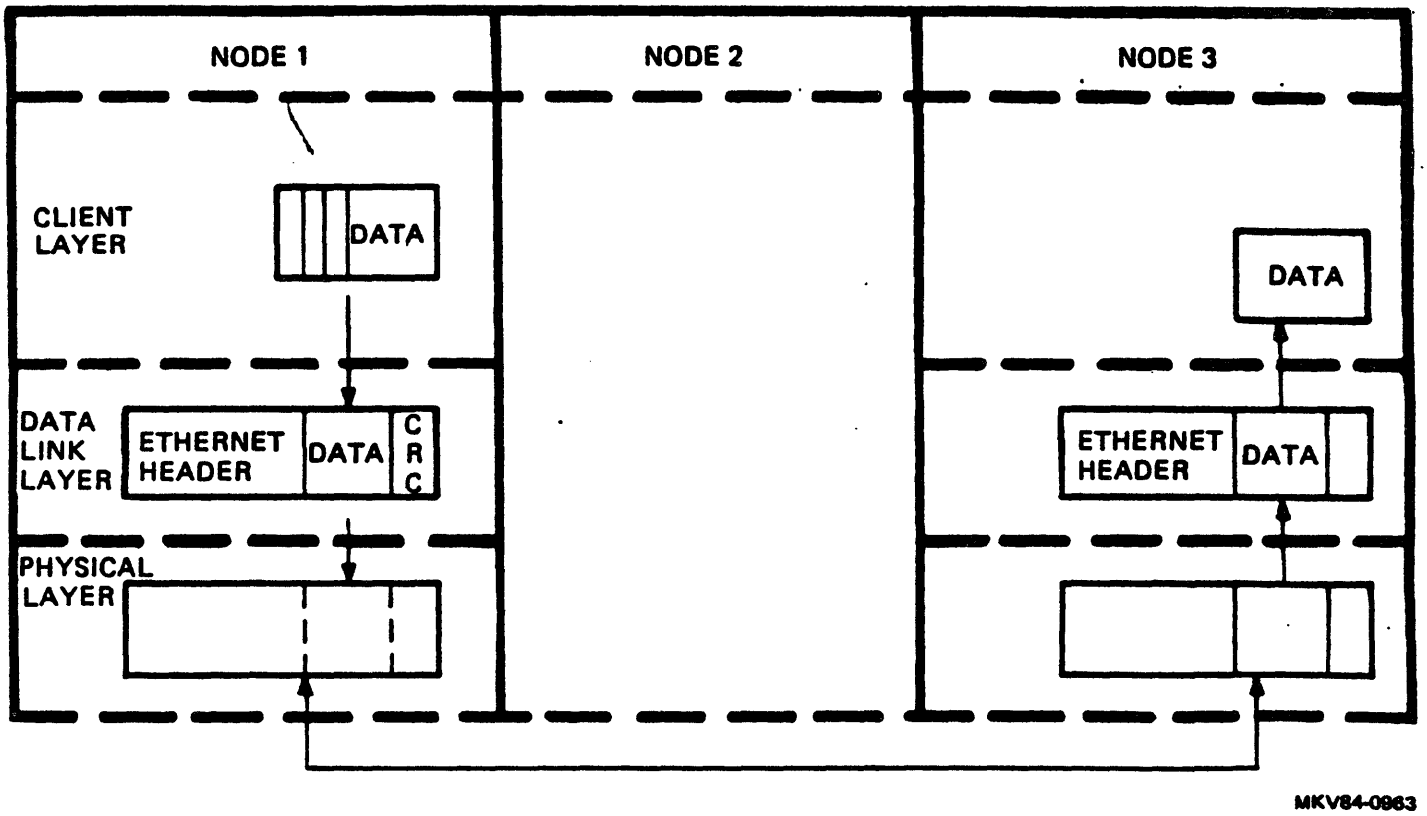


Figure 2 Data Flow from Node 1 to Node 3
(Ethernet Portion of the Network)

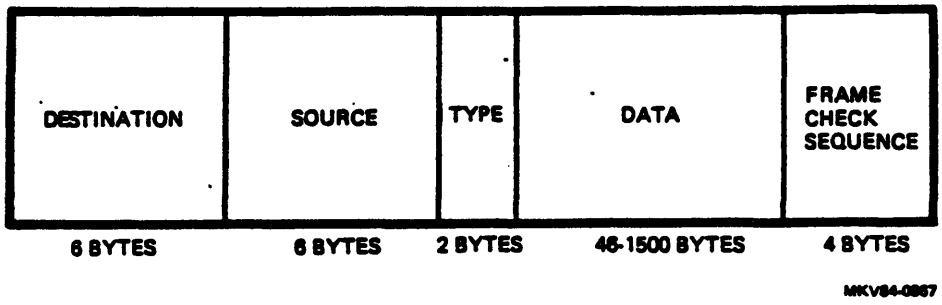


Figure 3 Ethernet Data Link Layer Frame Format

Destination address field - specifies the station(s) for which the frame is intended. It may be a physical or multicast address.

Source address field - specifies the station sending the frame. The source address field is not interpreted by the Data Link Layer. It is specified here for uniformity and because this field is necessary to higher level protocols.

Type field - reserved for use by higher layers. It identifies the client layer protocol associated with this frame. The type field is not interpreted by the Data Link Layer.

Data field - contains data from the Client Layer. This data includes higher level protocols as well as user-defined data. This field provides full transparency to the Client Layer.

Frame check sequence field - 32-bit Cyclic Redundancy Check (CRC).

Ethernet Communications

Transmission Without Contention

- Client Layer requests the transmission of a frame.
- Data Link Layer builds the frame from the client-supplied data and appends a frame check for error detection.
  - Attempts to avoid contention with other traffic by monitoring the carrier sense signal.
  - When the channel is clear, the frame is passed to the Physical Layer for transmision.
  - Once transmission is complete, the Data Link layer informs the Client Layer and waits for the next frame to transmit.

Reception Without Contention

- The arrival of a frame is detected by the Physical Layer.
- The frame is decoded and passed up to the Data Link Layer.
- The Data Link Layer checks the CRC for error.
- The Data Link Layer checks the destination address to determine if the frame belongs to this node. If so, the data is passed on to the Client Layer.

Contention Handling

If more than one Ethernet node attempts to transmit at the same, time the result is a collision. If a collision occurs:

- The Physical Layer detects the collision and turns on the Collision Detect signal.
- The signal is detected by the Data Link Layer and collision handling begins:
  - Data Link Layer transmits a "jam" signal so that the nodes involved realize that a collision has occurred.
  - A retransmission attempt is scheduled after a randomly determined period of time.
  - Retransmissions are repeated in the event of repeated collisions.

NOTE

In the event of repeated collisions Link Management services will adjust to the channel load by voluntarily delaying retransmissions (backing off) to reduce the load on the channel. The Data Link Layer will attempt 15 retransmissions before dropping the message.

Ethernet Routing

The routing layer for Ethernet nodes is handled differently from routing on DECnet nodes. There are:

- Ethernet Router Hello messages
- Ethernet End Node Hello messages
- Ethernet Routing message
- No initialization or verification messages

Ethernet Routing Layer Handshaking

Phase IV Nonrouting Node (Node A) Coming up on the Ethernet

- Node A multicasts an "End Node Hello Message" to all routers on the Ethernet. This message is repeated periodically to prevent corruption of the routing database due to messages that are lost.
- The Routers update their databases to include the new node (Node A). The routers multicast "Router Hello Messages" periodically to all other routers. The Designated Router (the router with the highest router priority) multicasts the "Router Hello Message" to all nonrouting nodes.
- The nonrouting nodes store the information about the Designated Router.
- When Node A wants to communicate with Node B, it does the following:
  - Check if the information about Node B is already in its "cache". If so, Node A will address Node B directly.
  - If no information is in "cache", Node A addresses the designated router to route the message to Node B.
  - If there is no designated router, Node A addresses Node B directly.

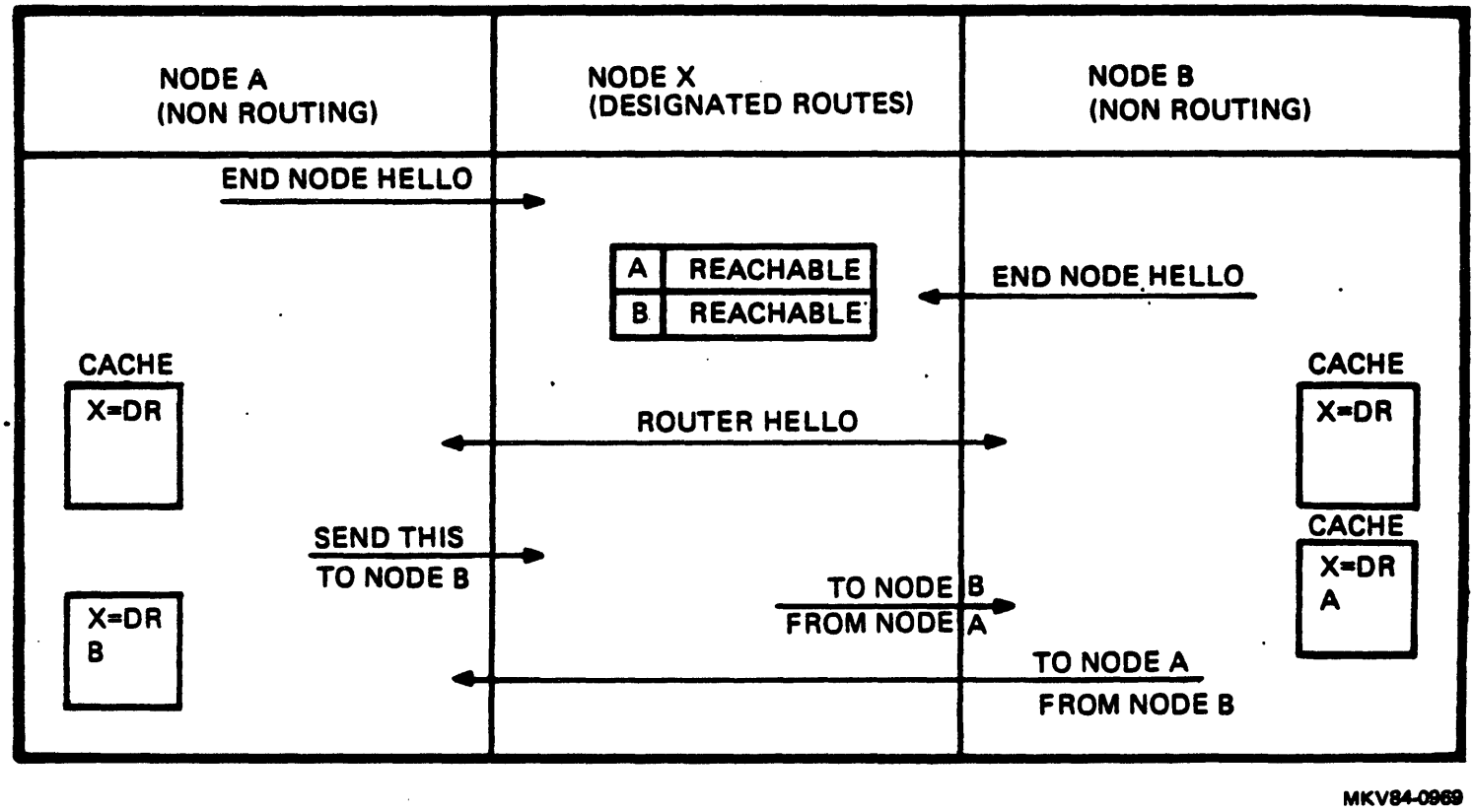


Figure 4 Ethernet End Node Communication

Phase IV Routing Node (Node A) Coming up on the Ethernet

- Node A multicasts the Router Hello message to all Routers.
- The Routers update their database to include Node A as a routing node.
- The Routers determine who will be the Designated Router by a "Router Priority" field in the Router Hello message. The Router with the highest priority is the designated router. If two routers have the same priority, the one with the highest node address becomes the Designated Router.
- End nodes notify Node A that they exist by sending "End Node Hello" messages.
- Due to resource constraints, there is a practical limit of ten routers on any one Ethernet network.

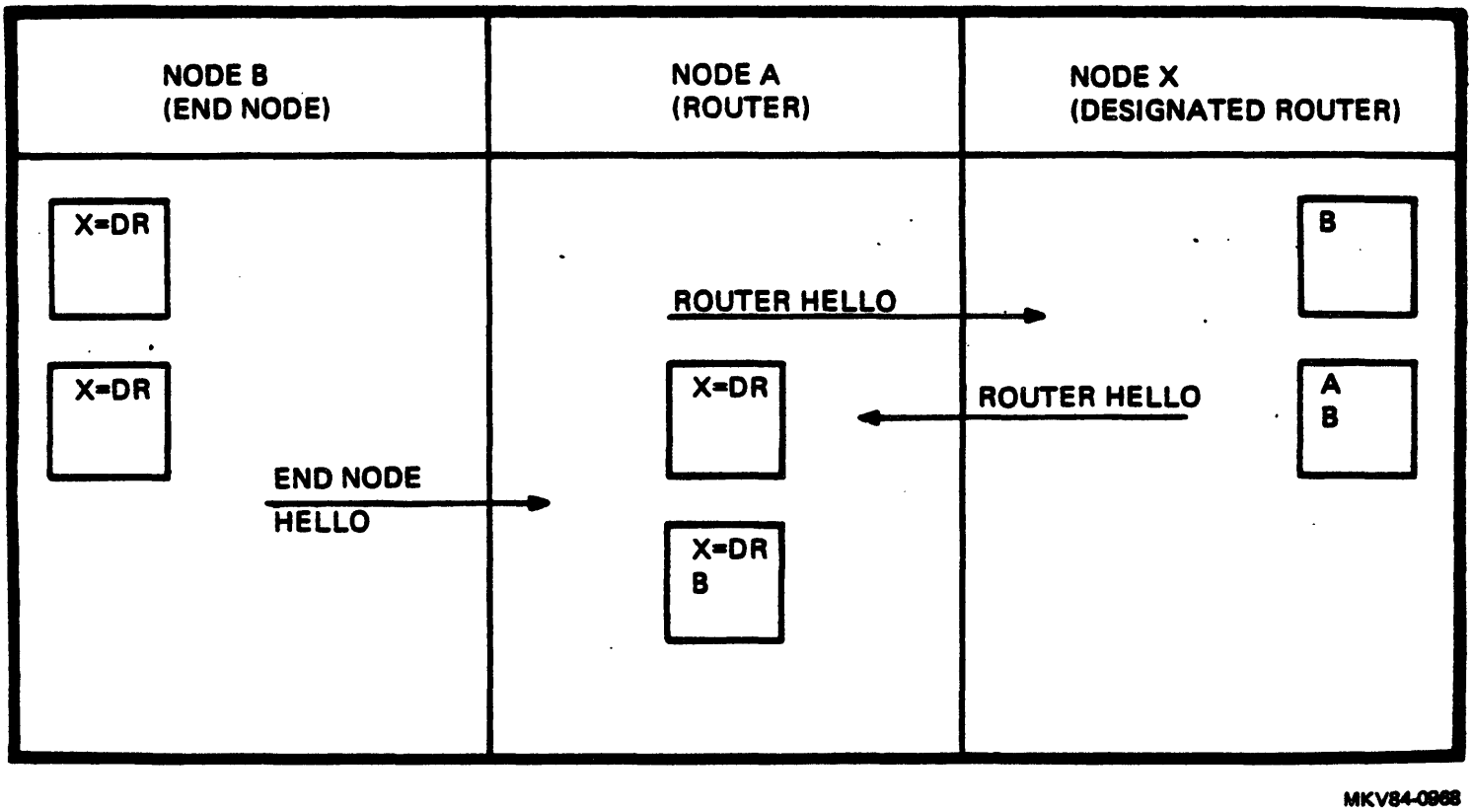


Figure 5 Ethernet Routing Node Communication

DECSA HARDWARE

The DIGITAL Ethernet Communications Server (DECSA) is an Ethernet based system for local area networks. There are three basic versions of the server each supporting one of the following:

- o 8 lines for connections to another Ethernet, DECnet, or a PPSN
- o 16 lines for terminals
- o 32 lines for terminals

The server hardware consists of:

- o PDP-11 Processor - The processor along with the operational software, which resides in memory, controls all of the server operations. This software is down-line loaded over the Ethernet.
- o Memory Module (512KB)
- o Ethernet to Unibus Adapter (DEUNA) - uses DMA to transfer messages between memory and the Ethernet.
- o Console/Boot/ Terminator (CBT) - provides all the console functions and indicators.
- o Protocol Assist Modules (PAM) - uses DMA to transfer messages between memory and line cards. Implements the DDCMP protocol, frames HDLC messages, and provides a forms control asynchronous protocol for terminals.
- o Line Cards - contain the logic to interface between the server and the physical line. The line cards support different line speeds and protocol types (DDCMP, HDLC, Asynchronous for terminals). Table 1 shows the various line cards available for the server products. Figure 6 shows the components of a communications server hardware unit.

Table 1 Communications Server Line Cards

LINE CARD	FEATURES	SERVER TYPE
M3100	- One synchronous line - Up to 19.2 K bps - Full Duplex - RS-233-C/V.24	- DECnet Router - DECnet Router/X.25 Gateway - Terminal Server
M3101	- One synchronous line - Up to 500 K bps - Full Duplex - V.35	- DECnet Router - DECnet Router/X.25 Gateway
M3102	- Two-line asynchronous - Up to 19.2K bps - Full Duplex - RS-232-C/V.24	- Terminal Server

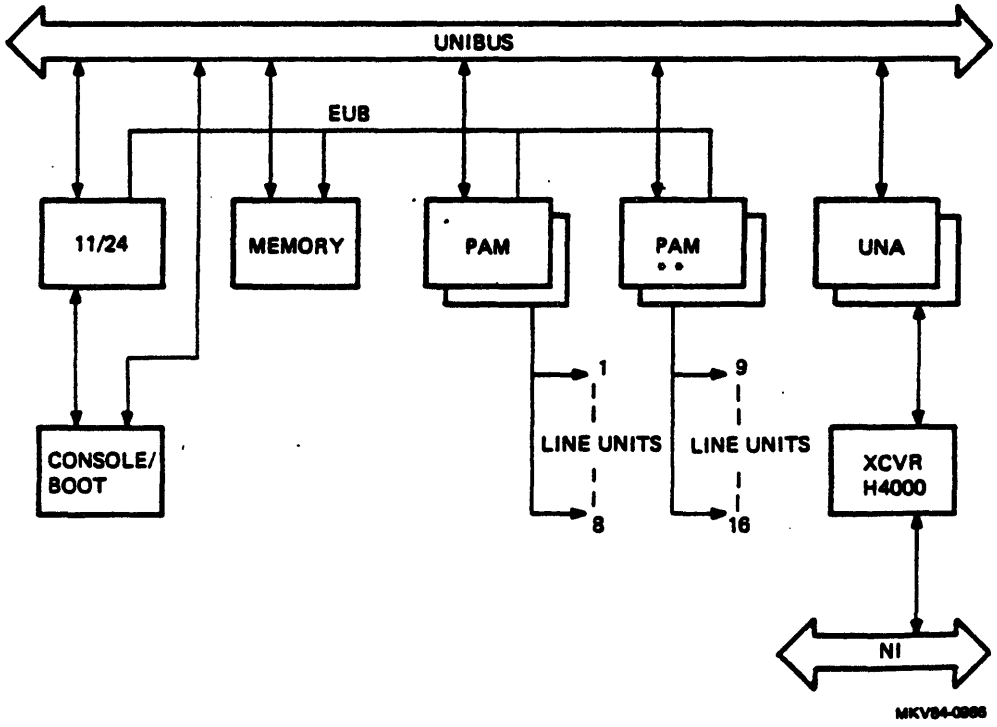


Figure 6 Communications Server Hardware Components

SERVER SOFTWARE

The server software is logically divided into two sections:

- Server base - an RSX 11-S operating system that provides generic DECnet communications capabilities and is common to all server products.
- Server specific software - implements a particular server function using the facilities provided by the Server base.

The Server base itself is not a product; specific software must be layered on top of it. The type of software layered determines the server's function (Terminal server, DECnet Router, etc.). The server-specific software and Server base form a product that is packaged as a single image on the distribution media. This image is copied to an Ethernet host and down-line loaded into the server hardware.

DECnet Router Server

The DECnet Router Server offloads routing tasks from the Ethernet hosts that it serves. This allows the host systems to function as end nodes in the network, thus freeing processor time for applications. The host systems can also share the use of all the Router Server's communications lines. The DECnet Router server supports:

- Up to eight synchronous point-to-point lines
- Routing to 1022 nodes on either local or remote networks
- Communications to all nodes on the Ethernet network
- Connection to another Ethernet network

Table 2 shows the maximum line configurations supported by the DECnet Router Server.

Table 2 DECnet Router Maximum Line Configurations

Max Line Speed	M3100	M3101
19.2 Kb	8	-
56 Kb	-	8
256 Kb	-	2
500 Kb	-	1

NOTE

The maximum line speed for the Router Server is 500 Kbps.

The remote nodes connected to the Router server may be any one of the following:

- o DECnet router servers or Phase IV DECnet routing nodes on other LANs.
- o Phase III or Phase IV DECnet routing nodes in a wide-area network
- o Remote Phase III or Phase IV nonrouting nodes (end nodes)

Figure 7 shows a DECnet Router Server, RTRDEV, with three synchronous lines connected to other DECnet Phase IV and Phase III routing and end nodes.

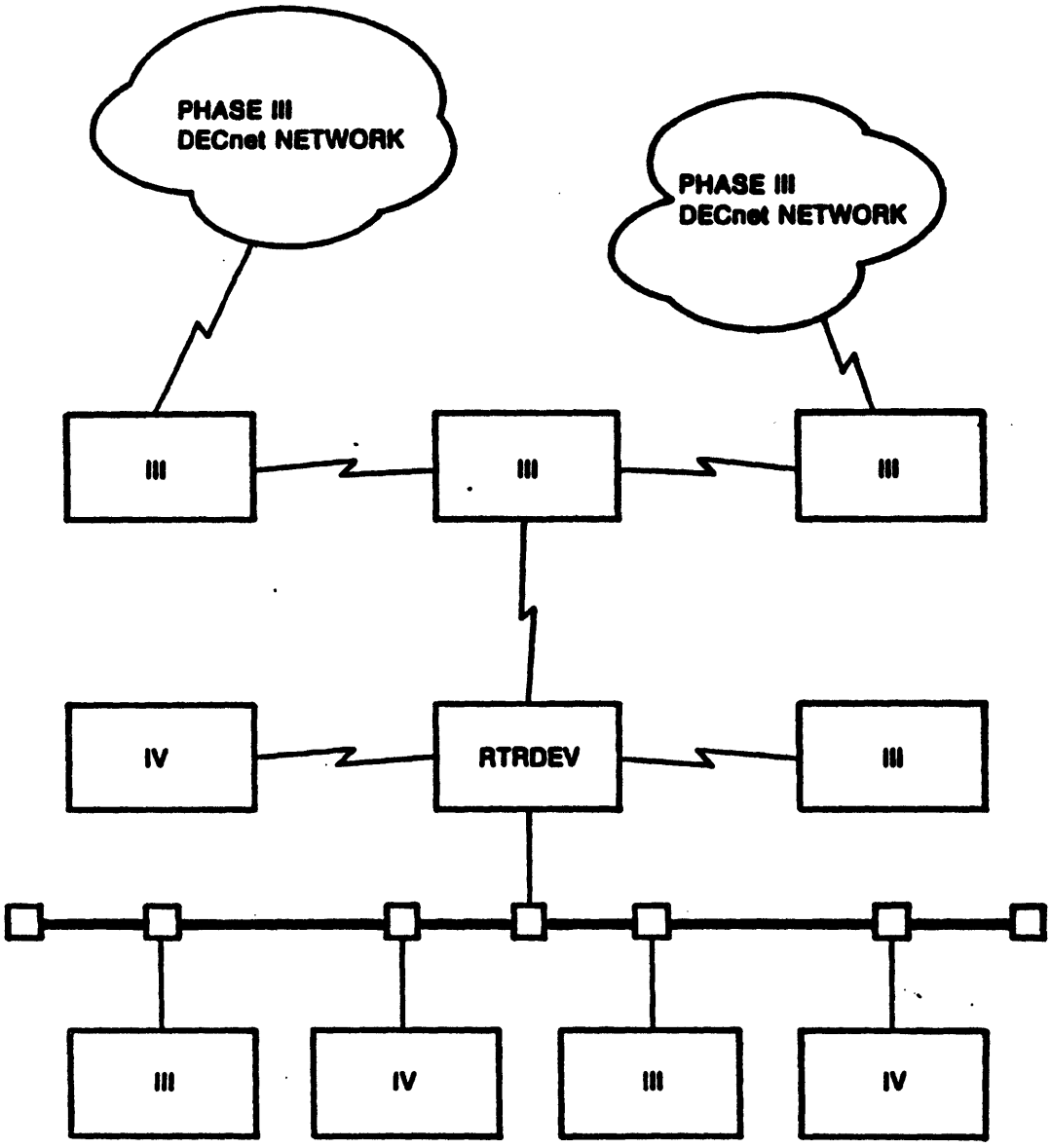


Figure 7 DECnet Router Server with Three Synchronous Lines

DECnet Router Server Features

FEATURE	BENEFIT
Dedicated Phase IV adaptive routing	Shared expense for all users of leased lines
	Routing overhead is offloaded from host systems
	Greater network reliability
	Network configuration flexibility
	LAN to LAN interconnection
Phase III compatibility	Investment protection for existing Phase III networks
Support for up to 8 synchronous lines	Direct access to many local or remote nodes
	Up to 500K bps for 1 line
	Cost effective connections

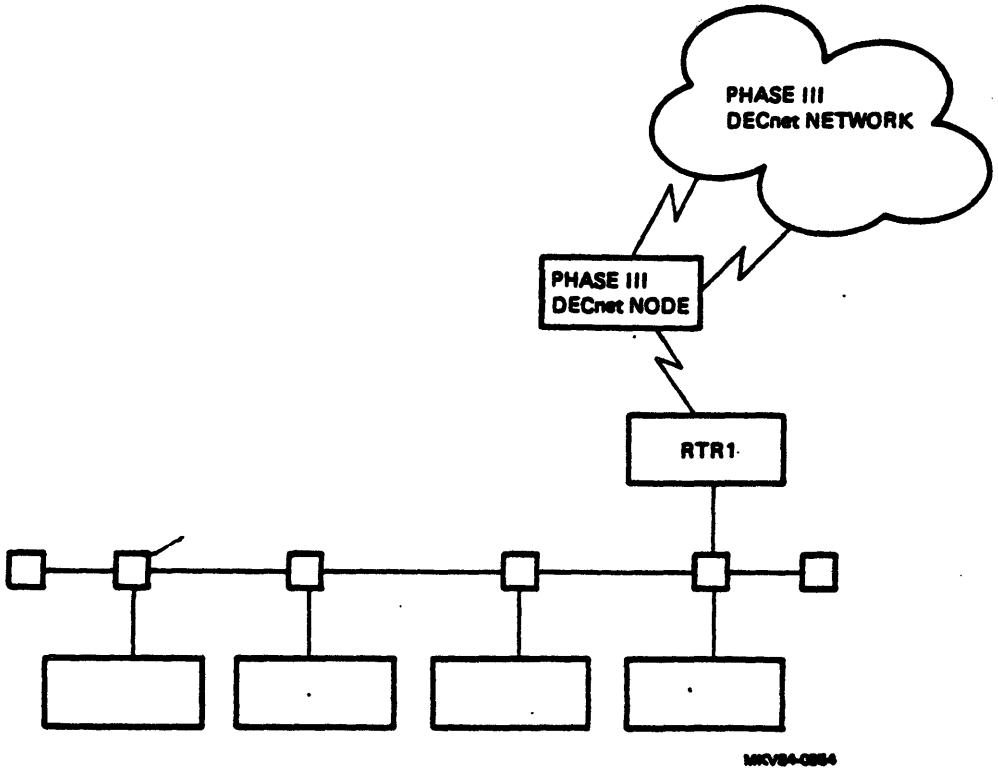


Figure 8 Ethernet Configuration Utilizing a Router Server

INSTALLATION AND CHECKOUT PROCEDURES

Installation consists of copying files from distribution media to each of the hosts that will be used to load the server. It requires the user to:

- Create an account and related directory with an ID (PLUTO) and a password (PLUTO)
- Edit configuration files to customize the server to his needs
- Set up the Down-Line load database
- Use an automated installation procedure (VMSINSTAL or RSXINSTAL)

Configuration Files

Sample configuration files are supplied as part of the distribution kit. There are two such files for each server product:

- Network configuration file - contains network-related information to be used by the server.
- Software Configuration File - contains server product-specific information

These configuration files contain default parameters initially and can be edited by the user to reflect his network needs.

Down-Line Loading on the Ethernet

Once the distribution image is copied, the down-line load database set up, and the configuration files edited, the system image is down-line loaded into the server hardware unit. The down-line load includes:

- The 11-S system image
- DECnet capabilities
- Configuration determined by the configuration files

Down-Line Loading

- NCP>LOAD NODE or TRIGGER command initiates the load.
- Uses MOP protocol.
- The server node transmits a REQUEST PROGRAM message to the loading host or to the "LOAD ASSISTANCE" multicast address for backup hosts.
- The host that responds sends the secondary bootstrap code.
- The loading sequence continues with the tertiary loader.
- Finally, the operating system image is sent.

Up-Line Dumping

If the server node should crash, then an up-line dump will occur to the host or hosts specified in the down-line load/up-line dump database.

- Uses MOP protocol.
- The server attempts to dump its' memory to the host which initially loaded it.
- If that host in unavailable, the server will search its' backup host list for another host to receive the dump.
- If no hosts are available, the server will use the multicast address. The first host that responds will be used. There is no way to determine which host received the dump other than to check each host's event log for event 0.3.

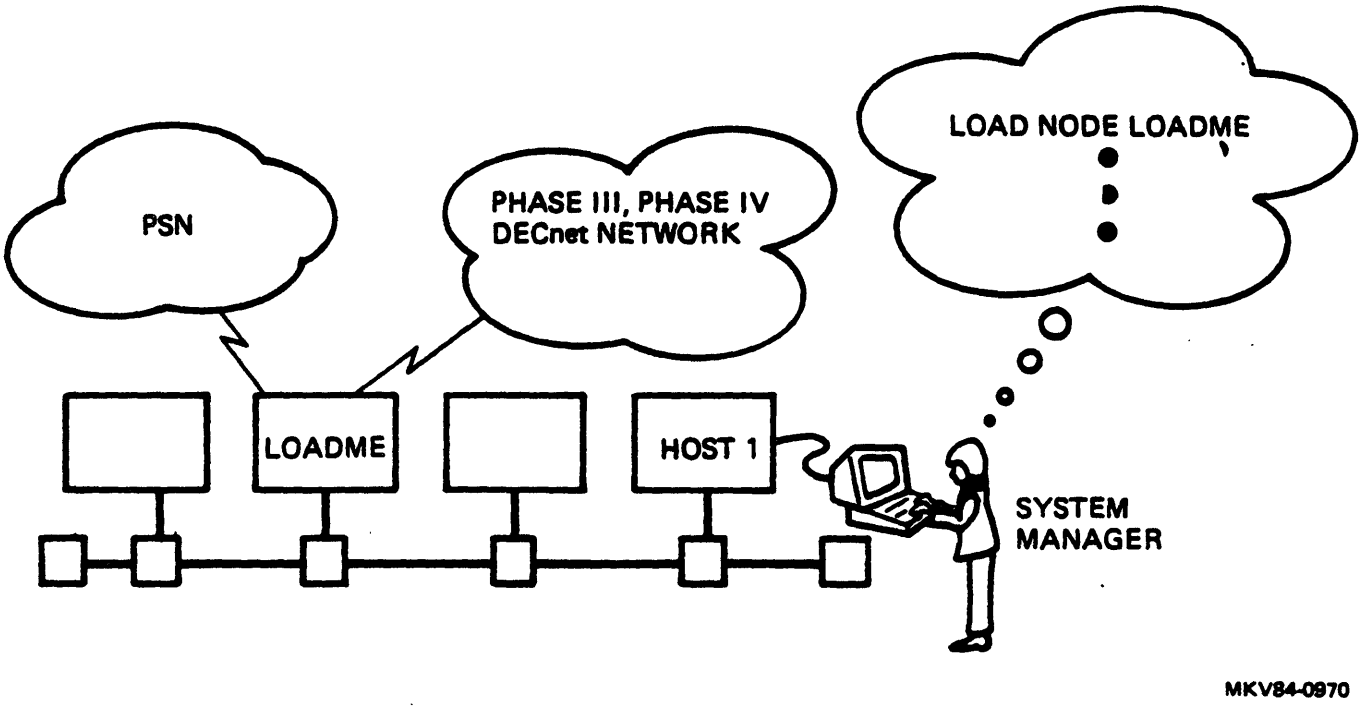


Figure 9 Down-Line Loading the System Image

NODE TROUBLESHOOTING

There are a number of basic tools available for managing the server node. These include the standard NCP functions which allow a user to:

- Configure a node
- Configure the network
- Test nodes and the network
- Control node and network operations
- Inspect and monitor a node or network

Along with the NCP capabilities provided for troubleshooting there several tools available for diagnosing more complicated node and network problems.

- Diagnostic tools
  - Server Self Test
  - Loadable Diagnostic Image (LDI)
- Network management facilities
  - Loopback Tests
  - Error Counters
  - Network Event Logging
  - Remote Console Facility
- On-line Debugging Tool

CONTENTS

FIGURES

TABLES

INSTALLATION AND CHECKOUT PROCEDURES

INSTALLATION AND CHECKOUT PROCEDURES

INTRODUCTION

Installation of a communication server software product involves choosing a host node directly connected to the Ethernet which will serve as the loading host. This loading host can be any Phase IV DECnet RSX-11M/M-PLUS or VAX/VMS node. Additional host nodes may be chosen to serve as backup hosts in the event that the initial or "primary" host is not available. Backup hosts can be used to load the server software as long as the particular software configuration files have been copied to these nodes and a down-line load database on each has been set up.

NOTE

RSX hosts must support multiuser protection.

The procedure for installing the server software products is automated. Briefly, it performs the following functions:

- Creates an account and directory on the loading host. The installation command file prompts for needed information.
- Copies all files from the distribution medium into the newly created directory on the host.
- Sets up the down-line load and up-line dump database on the initial loading host system.
- Copies loader files and system images to SYS$SYSTEM on VMS hosts and to NETUIC on RSX hosts.
- Sets up the server software configuration file.

Once the software has been installed the remaining steps are performed manually:

1. Load the server software into the communications server.
2. Perform the installation checkout procedures to verify that the server product is working properly.
3. Copy files to any backup host(s) that have been selected.
4. Protect directories, accounts, and configuration files

INSTALLATION AND CHECKOUT PROCEDURES

This module covers the preparation, installation, and loading of the DECnet Router server software.

REFERENCES

DECnet Router Server Installation and Operation Manual

INSTALLATION AND CHECKOUT PROCEDURES

PREPARING TO INSTALL THE SERVER SOFTWARE

Prior to installing the server software several preliminary steps must be taken:

- The communications server hardware must be installed; the hardware self tests and diagnostics must be completed without error.
- The user needs to:
  - Know the DEUNA 48-bit Ethernet hardware address. This address uniquely identifies the communications server hardware unit and is displayed during the hardware long self test.
  - Know the line card configuration (recorded on the Line Configuration Worksheet before installation).
  - Know modem and network characteristics such as modem speed, network buffer size, maximum address, etc.
  - Enable event logging at the server's loading host console or monitor.
  - Copy sample configuration files and make necessary changes to it.

NOTE

The Ethernet hardware address should be recorded on the back of the front cover and on the Line Configuration Worksheet provided with the hardware installation guide.

INSTALLATION AND CHECKOUT PROCEDURES

INSTALLING THE SERVER SOFTWARE

Installing the server software is an automated procedure: for a VMS host the procedure is accomplished through the use of the VMSINSTAL command file in SYS$UPDATE. Installing the software on an RSX host is done by executing the RSXINSTAL command file after copying the distribution to any UIC on the loading host. These automated procedures require very little from the user other than answering questions as they appear while the command file runs. Most questions require either (Y)es or (N)o answers.

Installing the Server Software on a VMS Host

To begin the installation session on VMS, the user must first log into the SYSTEM account, and enter the following command:

```
$@SYS$UPDATE:VMSINSTAL CSVRTR device name
```

device name - device on which the distribution kit is mounted

NOTE

The VAX-11 Software Installation Guide provides a full description of the VMSINSTAL procedure.

Example 1 shows a Router Server installation session on a VMS host.

Example 1 VMS Installation Procedures

```
$ SET DEF SYS$UPDATE
$ @VMSINSTAL

        VAX/VMS Software Product Installation Procedure

It is 9-JAN-1984 at 09:34
Enter a question mark (?) at any time for help

%VMSINSTAL-W-DECNET, Your DECnet network is up and running.
%VMSINSTAL-W-ACTIVE, The following processes are still active:
        HARROLD
        JONES
        MARSHAL
        ERICKSON
        WALDEN
* Do you want to continue anyway [NO]? Y
* Are you satisfied with the backup of your system disk [YES]? Y
* Where will the distribution volumes be mounted: MTA0:

Enter the products to be processed from the first distribution volume set.
* Products: CSVRTR

The following products will be processed:

  CSVRTR  V1.0

        Beginning installation of CSVRTR V1.0 at 09:35

%VMSINSTAL-I-RESTORE, Restoring product saveset A...

Router Server V1.0 installation procedures.

The following set of questions asks you for information used to
set up the account which will be used to install the Router
Server software.

* Device for the account [SYS$SYSROOT:]:
* UIC for account [014,001]:

If there is already an account named PLUTO, you will receive a
warning message which you may ignore.
```

```
%VMSINSTAL-I-ACCOUNT, This installation creates an account named PLUTO.
user record successfully added
%%%%%%%%%%  OPCOM   9-JAN-1984 09:37:36.90
Message from user SYSTEM
PID=00003056 SYSTEM added SYSUAF record PLUTO, on 9-JAN-1984 09:37:36:86

%VMSINSTAL-I-SYSDIR, This product creates system directory [PLUTO].

Installing Router Server V1.0 files.
Router Server V1.0 files installed.
The images and command files of Router Server
V1.0 have been restored to disk. The next step in the
installation is to run RTRNCP.COM.

Executing command file RTRNCP.COM.

This command procedure defines the database on your VMS host
which allows you to downline load the server software and
to receive an upline dump of the server software after a
crash.

This command procedure can be run for any number of Router
Servers. Each execution produces an NCP command file unique for
each given Router Server.

This procedure can optionally execute the NCP command file.

* Server node-id       (1-6 chars)       [ ]              : DEAGOL
* Server node number   (1-1023)          []               : 125
* Hardware address     (12 hex-digits)   [ ]              : AA-00-03-01-09-63
* Maintenance host     (1-6 chars)       [EXODUS]         :
* Service circuit      ( )               [UNA-0]          :
* Service password     (1-8 hex digits)  [0]              :
* Store file           (YES or NO)       [YES]            : Y
* File name            (1-12 chars)      [DEAGOL.COM]     :
* Storage device       ( )               [SYS$SYSROOT:]   :
* Storage directory    ( )               [PLUTO]          :

Creating file SYS$SYSROOT:[PLUTO]DEAGOL.COM

* Execute file         (YES or NO)       [NO]             : Y
%NCP-I-NMLRSP, listener response - Success
Remote node =  125 (DEAGOL)
Database entry deleted
%NCP-I-NMLRSP, listener response - Unrecognized component, Node


Executing procedure RTRSB.COM

This procedure creates the DECnet Router Server
network configuration file.
```

```
The network configuration file will be named
'server_node_id'SB.CFG
where 'server_node_id' is the node name you have assigned
to the DECnet Router Server.

* Router Server Node  (1-6 chars)       [DEAGOL]       :
* Service password    (1-8 hex digits) [0]            :
* System password     (1-6 chars)      [null]         :
* Clock frequency     (50 or 60)       [60]           :

The next set of questions asks for backup hosts.
You may enter a maximum of five (5) backup hosts.
A null entry or five (5) entries will terminate solicitation

* Backup host         (1-6 chars)      [null]         :

Created the configuration file SYS$SYSROOT:[PLUTO]DEAGOLSB.CFG

During the first step of the DECnet Router Server installation,
a predefined Router configuration file was copied
to the newly created directory (PLUTO PLUTO) on the loading
host. (Appendix B of your DECnet Router Server Installation and
Operation Guide shows the predefined files used for a
DECnet Router Server.) If you are
using that configuration file as it is, then you have
finished installing your Router Server. If you wish to
customize that file or create your own, then read Chapter 3
of the installation manual.

* more Servers to define [NO]? N
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories.

        Successful installation of CSVRTR V1.0 at 10:16

Enter the products to be processed from the next distribution volume set.
* Products: EXIT

        VMSINSTAL procedure done at 10:17

$
```

INSTALLATION AND CHECKOUT PROCEDURES

Installing the Server Software on an RSX-11M/M-PLUS Host

The server distribution kit for an RSX host includes a copy of the server software image, configuration files, and other software. First the user must copy the distribution kit onto any UIC on the loading host. Then the RSXINSTAL command file may be run. This command file prompts for the main parameters needed to install the DECnet Router server. RSXINSTAL provides the following:

- Sets up a special account on the host for storage of the server configuration and command files (PLUTO/PLUTO).
- Moves other files to their respective locations
- Sets up the Network Configuration file.
- Prompts for parameters to be used to down-line load or receive an up-line dump.

NOTE

> The PLUTO/PLUTO account should be set up so users cannot log into it. Appendix F in the Router Server Installation and Operation Guide provides information on protecting the PLUTO accounts and files.

Copying the Distribution Kit

The procedure for copying the distibution kit depends on whether the kit is a tape or RL02 disk.

Copying a tape distribution requires the user to:

1. Set UIC to the account set up for the installation [x,y]
2. Allocate and Mount the tape drive
3. Use the BRU utility to copy the distribution files

Once the BRU utility is finished copying the files a message will be displayed on the user's terminal. The distribution files are now in the [x,y] account on the host. The .SYS files should now be moved to the network UIC and the configuration files moved to the PLUTO account.

Copying a disk distribution kit:

1. Copy the .SYS files to the network UIC

```
ASN LB:=SY:
SET /UIC=[netuic]
PIP /NV=DLn:[x,y]*.SYS
```

2. Copy the configuration and command files to the PLUTO account

```
SET /UIC=[uic] (PLUTO account)
PIP server-node-idSB.CFG/NV=DLn:SB.CFG
PIP server-node-idRTR.CFG/NV=DLn:RTR.CFG
PIP /NV=DLn:CSVRTRICP.CMD
```

Example 2 shows a Router Server installation session on an RSX host.

Example 2 RSX Installation Procedures

```
>;
>; ================================================================
>; DECnet Router Server Installation Procedure
>;        Started at 13:08:34 on 03-FEB-84
>; ================================================================
>;
>; Copyright (C) 1984 by
>; Digital Equipment Corporation, Maynard, Mass.
>;
>; This installation procedure installs the DECnet Router Server (server)
>; on a RSX-11M/M+ host. This procedure must be run from a privileged
>; account.
>;
>; This procedure will create and copy server files. It will create a
>; network configuration file based on the answers supplied in secton 2.
>; A file named CFE<node>.CMD will contain the CFE commands to define the
>; permanent database on this host. You have the option of not moving the
>; DECnet Router Server system files. This allows you to simply use the
>; command procedure to set up a second DECnet Router Server.
>;
```

In other words, when you are setting up more than one DECnet Router Server, you do not need duplicates of the software images and loader files. The same image and loader files can be used for each server.

```
>; You may type an escape <ESC> at any question to obtain additional
>; information about the question.
>;
>; <EOS>  Do you want to:
>*        <RET>-Continue, E-Exit [S]:
```

```
>;
>;
>; ================================================================
>; Section 1 - General information
>; ================================================================
>;
>; Now you must decide where you would like to put the various DECnet
>; Router Server files. It is suggested that you pick one uic for the
>; configuration files, one uic for the loaders and system images and one
>; uic for the dump images.
>;
>; NOTE: The uic where the configuration files reside must have a user id
>; and password of PLUTO/PLUTO.
>;
>; When answering questions in this section please type the full
>; directory specification. (DEV:[nnn,nnn])
>;
```

The last question of this section gives you the option of running the ACNT program to add new directories.

```
>* 1.00 Where will the configuration files reside [S D:"SY:[40,40]"]:
>;
>; This is the uic that is set up as the PLUTO/PLUTO account on the
>; host. Your <server-node-id>SB.CFG and <server-node-id>RTR.CFG
>; configuration files must reside in this uic. These files are
>; read by the DECnet Router Server's initialization tasks from
>; this account. Enter the uic specification in the form:
>;
>;                       DEV:[nnn,nnn]
>;
>* 1.00 Where will the configuration files reside [S D:"SY:[40,40]"]:
>* 1.01 Where will the system files reside [S D:"dev:[nnn,nnn]"]:
>;
>; The system files can reside on any uic on the host system. A
>; separate uic is a neat and convenient method of partitioning host
>; software from server software. Enter the uic specification in the
>; form:
>;
>;                       DEV:[nnn,nnn]
>;
>* 1.01 Where will the system files reside [S D:"dev:[43,43]"]:
>* 1.02 Where will the dump files reside [S D:"dev:[43,43]"]:
>;
>; The dump files can reside on any uic on the host system. A
>; separate uic is a convenient method of insuring disk storage for
>; a dump image. Enter the uic specification in the form:
>;
>;                       DEV:[nnn,nnn]
>;
>* 1.02 Where will the dump files reside [S D:"dev:[43,43]"]:
>* 1.03 Do you need to run the account program to add these directories? [Y/
>;
```

The uic specification you supplied may not exist on your disk. You can elect to run the accounting program now and add the new uic's to the system. Remember you must also create a PLUTO/PLUTO account for the configuration files.

If you answer YES, RSXINSTAL runs the account file maintenance program (ACNT) so that you can add directories. Section 2.3.4 explains how to respond to the ACNT program.

```
* 1.03 Do you need to run the account program to add these directories? [Y/N
;
; <EOS>  Do you want to:
*        <RET>-Continue, R-Repeat section, E-Exit [S]:
;
;
;; ================================================================
>; Section 2 - Building the network configuration file
>; ================================================================
>;
```

This file contains LAN software network parameters used by the DECnet Router Server. Chapter 3 describes the parameters in detail and explains how to edit the configuration file to change the parameters. Appendix B contains a sample network configuration file.

```
>* 2.00 What is the node-id of the server [S]:
>;
>; Every DECnet node can be identified by a unique name. You should
>; choose a unique name for each node in the network. A node name is
>; one to six alphanumeric characters. At least one character must be
>; a letter.
>;
>* 2.00 What is the node-id of the server [S]:
>* 2.01 What is the node-address of the server [D R:1.-1023.]:
>;
>; Every DECnet node can be identified by a unique number. You
>; should pick a unique number for the DECnet Router Server.
>;
>* 2.01 What is the node-address of the server [D R:1.-1023.]:
>* 2.02 What is the UNA-0 password for the server [S R:1.-16. D:"0"]:
>;
>; The UNA-0 password (service password) is used to prevent
>; unauthorized users from triggering or loading the DECnet Router
>; Server. The password is one to sixteen hexadecimal characters.
>;
>* 2.02 What is the UNA-0 password for the server [S R:1.-16. D:"0"]:
>* 2.03 What is the network privileged password for the server [S R:0.-8.]:
>;
>; The network privileged password is supplied on network management
>; commands. This is to prevent unauthorized users from issuing
>; privileged commands to the DECnet Router Server.
```

```
>;
>* 2.03 What is the network privileged password for the server [S R:0.-8.]:
>* 2.04 What is the system line clock frequency for the server [D R:50.-60. D
>;
>; The clock frequency used in the United States is 60 Hz and in
>; Europe is 50 Hz.
>;
>* 2.04 What is the system line clock frequency for the server [D R:50.-60. D
>;
>; If you want more than one backup host enter them separated by commas.
>; i.e. HOST1,HOST2,HOST3
>;
>* 2.05 What are the node-id's of the backup hosts [S]:
>;
>; Backup hosts are needed when the server's maintenance host is out
>; of service. A DECnet Router Server may need to log an event or
>; upline dump a system image. If the maintenance host is down, a
>; backup host is needed to assist the DECnet Router Server. Be sure
>; to set up the server databases on the backup hosts. Enter the
>; backup hosts as a list. The maximum is five backup hosts. For
>; example:
>;
>;                    HOST1,HOST2,HOST3,HOST4,HOST5
>;
>* 2.05 What are the node-id's of the backup hosts [S]:
>;
>;
>; Building the network configuration file. (dev:[nnn,nnn]server-node-idSB.CFG
>;
>;
>; NOTE: You must also tailor the router configuration file in
>; dev:[nnn,nnn]server-node-idRTR.CFG after this procedure completes.
>;
```

The router configuration file contains parameters for the routing database used by the DECnet Router Server. The file supplied with the distribution kit sets up default parameters for 8 point-to-point lines and the UNA. Appendix B shows this file. You can use the file as is, provided your configuration matches the defaults defined in the file. You can edit the file to tailor it to the needs and characteristics of your network, as explained in Chapter 3. Sample configuration files are contained in Chapter 3 and Appendix B.

```
>;
>*        <RET>-Continue, R-Repeat section, E-Exit [S]:
>;
>;
>; ============================================================
>; Section 3 - Moving the DECnet Router Server files.
>; ============================================================
>;
>* Do you want to move the system files? [Y/N]:
```

```
>; Section 4 - Building the downline load database
>; ================================================================
>;
```

The downline load and upline dump database is kept on the loading host and contains information that the loading host uses to load the server software into the communications server hardware unit and to receive an upline dump of the server memory if the server crashes. The following questions allow you to specify:

1. The host's service circuit used for downline loading the server and receiving dumps of the server.

   and

2. The server's 12-digit Ethernet hardware address.

RSXINSTAL automatically sets up other parameters in the database by using values specified in Section 2 of RSXINSTAL, such as the server's node name and service password (UNA password). RSXINSTAL defines the loading host as the server's maintenance host. The system images and loader files have default names. You determine their location in Section 1 of RSXINSTAL. The names and descriptions of each are:

CSVRTR.SYS - The server's system image.
CSVLDI.SYS - The loadable diagnostics image.
PLUTO2.SYS - The secondary loader file used for loading the server software.
PLUTO3.SYS - The tertiary loader file used for loading the server software.

RSXINSTAL builds a command file containing Configuration File Editor (CFE) commands that define the permanent database parameters. After RSXINSTAL completes, you can run this command file to update the permanent database on the host, as explained below. Or, you can use the CFE DEFINE NODE command directly to change one or more parameters in the permanent database of the host, as explained below.

NOTE

For security reasons, you may choose not to define the service password in the downline load database. In this way, anyone wishing to load the server must specify the password with the LOAD or

TRIGGER command, as explained in Section 2.4.1.

Figure 2-2 shows how an installer uses CFE to set up the database on HOST1 for server RTRDEV. The installer has assigned the system images and files to UIC [100,100] on LB:. By default, the host is HOST1; that is, HOST1 will be RTRDEV's maintenance host. Appendix C includes the DEFINE NODE command and all the possible downline load and upline dump parameters. (See the DECnet-RSX System Manager's Guide for further information on CFE.) When using CFE directly to set up the database, make sure the UICs you specify for parameters correspond to the UICs where you have stored the files on the host.

NOTE

Make sure you set up a similar database for the server node on all other hosts that are to be used for loading the server or for receiving crash dumps.

Figure 2-2: Setting Up the Downline Load and Upline Dump Database on an RSX Host

```
>* 4.00 What is the loading host's service circuit-id  [S D:"UNA-0"]:
>;
>; This  host  node  may  have more than one active circuit. You must
>; specify the circuit over which service will be performed.
>;
>* 4.00 What is the loading host's service circuit-id  [S D:"UNA-0"]:
>* 4.01 What is the server's hardware address [S D:"FF-FF-FF-FF-FF-FF"]:
>;
>; Every  Ethernet controller has a unique  12-digit  address.  This
>; hardware address is displayed by running the long diagnostics self
>; test. This address will be used by the host to service the  DECnet
>; Router Server. Enter the address as six pairs of digits  separated
>; by hyphens (-).
>;
```

This address is the unique address assigned to the server's hardware unit. The address is used to communicate with the server node before the server software has been loaded. If you do not know the address, you can have it displayed on the front panel of the

communications server hardware unit by turning on the hardware unit with the TEST button pushed in, as explained in Section 5.2 of this manual. The address is displayed after the long self-test. After the address is displayed, turn the unit off or start it again.

```
>* 4.01 What is the server's hardware address [S D:"FF-FF-FF-FF-FF-FF"]:
>;
>;
>;
>; You can define these node characteristics in the host's permanent
>; database by running CFE with the command file built in this
>; procedure (CFERTRDEV.CMD), as follows:
>;
>; >RUN CFE
>; Enter filename: DEV:[NETUIC]CETAB.MAC
>; CFE>@CFERTRDEV
>; CFE>sho node RTRDEV
>;     .... Displays new node characteristics
>; CFE>Z
>;
>;
>;
>; <EOS>  Do you want to:
>*       R-Repeat section, E or <RET>-Exit [S]:
>;
>;
>; Remember to:
>;
>;   1. Edit your router configuration file.
>;
>;   2. Update your permanent database with the newly created CFE command
>;      file. The next time your network is loaded the volatile database
>;      on the host will be setup.
>;
>;   3. If your volatile database is set up, then you may trigger or
>;      load the server. If the volatile database is not set up, then
>;      downline load your server with the full NCP command:
>;
>;      NCP LOAD NODE RTRDEV FROM dev:[nnn,nnn]CSVRTR.SYS SER PASS password
>;
```

For the server image file (CSVRTR.SYS), use the device and UIC that you specified in Section 1 of RSXINSTAL. For the service password, use the password specified in response to question 2.02 in Section 2 of RSXINTAL.

```
>;
>; ============================================================
>; DECnet Router Server Installation Procedure
>;       Stopped at 13:11:03 on 03-FEB-84
>; ============================================================
>;
>@ <EOF>
>
```

The installation procedures copy the following system images to the SYS$SYSTEM directory in VMS or the NETUIC account in RSX:

Table 1 Server System Image Files

PLUTOCC.SYS	The console carrier image. This image is used by the Remote Console Facility (RCF). When RCF is invoked from a host, the console carrier image is loaded into the DEUNA device on the server node.
PLUTOWL.SYS	The console carrier loader. This loads the console carrier image when RCF is invoked.
CSVLDI.SYS	The communications server hardware unit loadable diagnostic image.
PLUTO2.SYS	The secondary loader, used for down-line loading the server software.
PLUTO3.SYS	The tertiary loader, used for down-line loading the server software.
CSVRTR.SYS	The DECnet router server system image.

NOTE

The configuration files are copied to the directory in the PLUTO PLUTO account.

INSTALLATION AND CHECKOUT PROCEDURES

SERVER SOFTWARE SETUP

When the communications server is loaded, the software's initialization task uses the network file access routines to read network and server software configuration files from the loading host. The information that is contained in these files is used to set up the server's configuration. Before loading the server these files may be edited to reflect the needs of a particular configuration. They can also be modified after loading to reflect changing needs of the configuration. Any text editor can be used to modify the configuration files. The changes take effect when the software is reloaded into the server. There are two files that need to be modified to configure the server system:

- Network Configuration File - this file is set up during the installation procedure. During this procedure the user is asked to supply the parameters for his network configuration. This file is the same for all server products.

- Server Software Configuration File - this is the file supplied with the distribution kit. It may be edited to reflect a particular network's needs. This file is specific to each server product.

During installation the files are created with the following names and are stored on all loading hosts. These names must not be changed.

- server-node-idSB.CFG - Network Configuration File

- server-node-idRTR.CFG - Router Server Configuration File

NOTE

server-node-id refers to the node name given to the server

To edit the configuration files for a router server node BARNEY, using EDT as an example the following command is used:

```
EDT>BARNEYSB.CFG

EDT>BARNEYRTR.CFG
```

Example 3 Editing the Router Configuration Files

Modifying the Network Configuration File

The network configuration file contains commands that set up the following parameters:

- The UNA password - prevents unauthorized nodes from reloading the server. When the server is loaded with the NCP LOAD or TRIGGER commands and a service password is specified with the command, the server software compares it to the UNA password. If the passwords do not match then the loading is denied. If a password is not specified with the NCP command the software will compare a default service password from the down-line load database in the servers loading host. To set the UNA password edit the network configuration file command:

  SET LINE UNA-0 PASSWORD hex-password

- The network privileged password - identifies the password used for access control verification at the server node. Users must specify this command when using privileged network management functions to access the server node. To set this parameter edit the following command:

  SET SYSTEM PASSWORD password

- The clock frequency for the server node - used by the server's internal clock to keep track of time for such things as the time recorded on network event logging messages. The wrong frequency specification will record the wrong time in event messages. To set the clock frequency edit the following command:

  SET SYSTEM LINE FREQUENCY 60

- A list of backup hosts - These are nodes the server can access to record event messages or send up-line crash dumps. Up to five backup hosts may be specified. A server looks to the backup host list for a node to receive the event message or crash dump if it's primary host is unavailable. The server requests access to the first node on the list. If it is not available the server continues to the next on the list, and so on. If none of the hosts is available for a network event then the event will not be recorded. If no host is available for a crash dump then the server will send a request to the multicast address for the Ethernet. After a crash the host that receives the dump triggers a reload of the server. To specify the list of backup hosts change the following command:

SET EXECUTOR BACKUP-HOSTS name1,name2,name3,name4,name5

NOTE

If all backup hosts are "busy" and unable to receive a crash dump from the server, it sends the dump to the multicast address. The node which accepts the dump does not have to be one of those specified in the backup host list. However only those nodes which have the up-line dump database built can respond. Also the node that receives the dump should be able to reload the server.

Modifying the Server Configuration File

The server configuration file is used by the communications server to initialize, allocate databases, set up routing parameters, etc. Each of the server products has a configuration file which defines parameters unique to the product.

Configuring the Router Server

The router configuration file is used by the DECent Router server to determine the network routing configuration. The file contains commands that set up the routing parameters to be used by the router server. The router configuration file uses the following commands:

- SET CIRCUIT
- SET EXECUTOR
- SET LINE

Table 2 lists the the parameters that may be defined or modified for each of the three components.

Table 2 Router Server Component Parameters

COMPONENT	PARAMETERS
EXECUTOR	Buffer Size Segment Buffer Size Maximum Address Maximum Cost Maximum Hops Maximum Broadcast Nonrouters Maximum Broadcast Routers
CIRCUIT	Cost Hello Timer State Router Priority
LINE	Speed Duplex Modem

INSTALLATION AND CHECKOUT PROCEDURES

LOADING THE SOFTWARE INTO THE SERVER

A database needs to be set up that contains information needed by the loading host to load the software into the server or to receive an up-line dump from a crashed server. The database contains the following information:

- The identification of the circuit to be used for the load or dump.
- The service password needed to trigger the bootstrap loader. This is the matching password that was set up as the UNA password.
- The name of the file containing the server image to be loaded (CSVRTR.SYS).
- The name of the file containing the loadable diagnostic image (CSVLDI.SYS).
- The names of the secondary and tertiary loader files containing software for down-line loading (PLUTO2.SYS, PLUTO3.SYS).
- The name of the node that will serve as the server's maintenance host after it is loaded. This host will receive all logged events generated by the server and it will receive the up-line dump of server memory should the server crash.
- The 48-bit Ethernet Hardware Address of the server node.

INSTALLATION AND CHECKOUT PROCEDURES

Setting up the Down-line Load/Up-line Dump Database on VMS

To set up the database on a VMS host use the server-node-id.COM file. This command procedure allows the definition of some parameters such as those for the NCP DEFINE NODE command. Other parameters are predefined such as the names of the loadable diagnostic image, dump files , and loader files. Example 4 shows the NCP DEFINE NODE command, set up at loading host VMSHOST, defining the database for the DECnet Router Server node BARNEY.

```
NCP>DEFINE NODE 134 NAME BARNEY -
    DUMP FILE BARNEY.DMP -
    HARDWARE ADDRESS AA-00-03-00-00-06 -
    HOST VMSHOST
    DIAGNOSTIC FILE LB:[100,100]CSVLDI.SYS -
    LOAD FILE LB:[100,100]CSVRTR.SYS -
    SERVICE CIRCUIT UNA-0 -
    SERVICE PASSWORD FF7A -
    SECONDARY LOADER LB:[100,100]PLUTO2.SYS-
    TERTIARY LOADER LB:[100,100]PLUTO3.SYS
```

Example 4 Setting Up the Down-line Load/Up-line Dump Database on a VMS Host

The NCP command file asks the following questions to fill in the above command:

- Server node-id - the name of the server node
- Server node number - the node number chosen for the server node
- Hardware address - 48-bit Ethernet Hardware Address assigned to the DEUNA
- Maintenance host - the name of the node that will be the maintenance host for the server after its loaded. The default host is the loading host.
- Service circuit - the UNA circuit to be used for down-line loading or up-line dumping.
- Service password - the password required to enable the bootstrap mechanism on the server node. This password must match the UNA password in the network configuration file.

- Store file - the NCP file can be stored for future use, executed now, or both. If the file is to be stored answer YES and the next three questions will be displayed.

- File name - the name to be given to the NCP command file.

- Storage device - the name of the device where the file will be stored.

- Storage directory - the name of the directory on which the file is stored.

- Execute file - Do you want to execute this file now?

NOTE

These commands change only the permanent database so for them to be executed the the server must be rebooted, or the NCP>SET SYSTEM command used (after NCP>CLEAR SYSTEM ALL and NCP>SET NODE ALL)

- More servers to define - answer YES, to define parameters for other servers, and the above procedures will be repeated. If there are no other servers then the installation procedure is complete and the answer to this question is NO.

```
$! NCP Configuration file for node BARNEY

$ ncp      purge node BARNEY all
$ ncp      purge node 4 all
$ ncp      define node 4 name BARNEY
$ ncp      define node 4 diagnostic file CSVLDI.SYS
$ ncp      define node 4 dump file SYS$SYSROOT:[PLUTO]BARNEY.DMP
$ ncp   define node 4 hardware address AA-00-03-00-00-06
$ ncp      define node 4 host VMSHOST
$ ncp      define node 4 service circuit UNA-0
$ ncp      define node 4 service password FF7A
$ ncp      define node 4 load file CSVRTR.SYS
$ ncp      define node 4 secondary loader PLUTO2.SYS
$ ncp      define node 4 tertiary loader PLUTO3.SYS
```

Example 5 NCP Command File for Router Node BARNEY

Setting up the Down-line Load/Up-line Dump Database on RSX

To set up the down-line load database on an RSX host the Configuration File Editor (CFE) is used. A similar database has to be set up on all the nodes that will be used for loading the server or for receiving crash dumps. The CFE command, set up at loading host RSXHOST, defining the database for the DECnet Router Server node BARNEY.

```
CFE>DEFINE NODE 136 NAME BARNEY -
    DUMP FILE BARNEY.DMP -
    HARDWARE ADDRESS AA-00-03-00-00-06 -
    HOST RSXHOST
    DIAGNOSTIC FILE LB:[100,100]CSVLDI.SYS -
    LOAD FILE LB:[100,100]CSVRTR.SYS -
    SERVICE CIRCUIT UNA-0 -
    SERVICE PASSWORD FF7A -
    SECONDARY LOADER [100,100]PLUTO2.SYS
    TERTIARY LOADER [100,100]PLUTO3.SYS
```

Example 6 Setting Up the Down-Line Load/Upline Dump Database on an RSX Host

Down-Line Loading Procedures

There are two ways to initiate the down-line load of the server software:

- Use the NCP LOAD NODE or TRIGGER NODE command.
- Press the START button on the front of the communications server.

NOTE

This assumes that the hardware is already turned on. If not then the load will start automatically when the switch is turned to the ON position.

Both methods produce the same results when loading the server for the first time. There are several steps the user needs to take before trying to load the server:

- The communications server hardware is powered up.
- The line associated with the UNA circuit over which the down-line load will take place is already loaded at the host, and the UNA circuit is set to the ON state.
- The UNA circuit at the host is service enabled (NCP SET CIRCUIT UNA-0 SERVICE ENABLED).
- All operating system and configuration files needed to load the server have been copied to the approprate places on the backup hosts.
- Optionally event logging at the most accessible host, usually the loading host, should be enabled. When a server is loaded, the software generates several event messages concerning the stage of the loading process and of the success or failure of the process.

Boot Button Initiated Load

The loading process is started at the server hardware unit by pressing the start button on the front panel. Figure 1 shows the front panel of the communications server. When the load is initiated the server hardware responds by sending a multicast request-to-load message to host nodes on the Ethernet. The host that loads the server is the first host to respond that contains the down-line load information for that server.

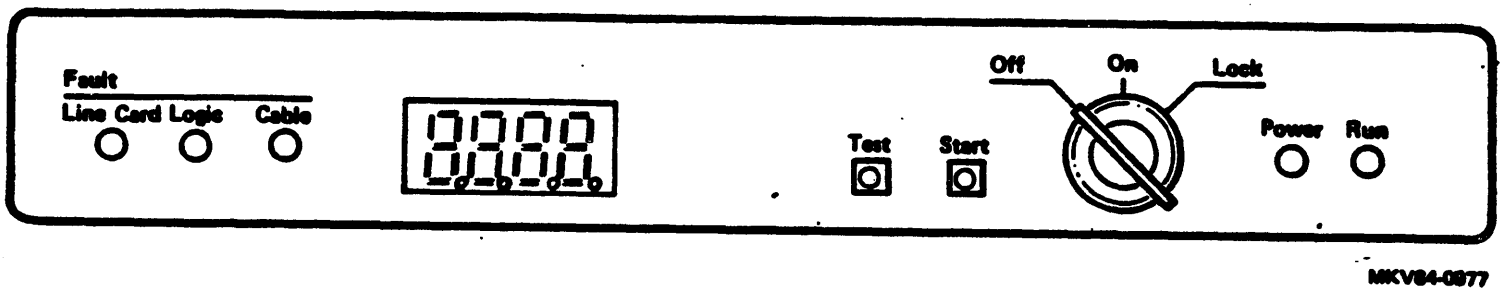


Figure 1 Communications Server Front Panel

The load process takes several minutes. During a successful load, the server's digital displays will be blinking, then L 30 will be diplayed. After about 30 seconds the display will show L 5n. The n rotates from 0 - 7 repeatedly, indicating that the software image is being loaded. The server node is up when a cyclic pattern shows on the server's digital displays. While the software is running the DECnet node address will alternate with the cyclic pattern.

NOTE

If an error occurs during or after the loading process, one of the fault indicators will light up and the digital display will show error information.

INSTALLATION AND CHECKOUT PROCEDURES

Host Initiated Load

Loading the server software from a host is done by using the NCP LOAD or TRIGGER command.These commands can be issued from a terminal either on the loading host or from a remote node connected to the loading host that has the server node defined in it's database. The LOAD command causes the load to begin quicker but the TRIGGER command releases the terminal quicker.

NOTE

The TRIGGER command completes when the server starts it's self tests while the LOAD command doesn't complete until the image is loaded.

Using The LOAD Command

The LOAD command forces the load from the host at which you execute the command. In addition the LOAD command allows the user to provide more information or override information in the volatile down-line load database.

The LOAD command is issued in the following formats:

```
NCP>LOAD NODE server-node-id [SERVICE PASSWORD hex-password]

                    or

        NCP>LOAD VIA circuit-id [NAME node-name]
                                [PHYSICAL ADDRESS E-address]
                                [SERVICE PASSWORD hex-password]
```

NOTE

When using the LOAD VIA command to specify the circuit to use when loading the server the server node name or the physical address must be specified.

Example 7 shows the NCP LOAD command being used to load a DECnet Router.

```
NCP>SET LINE UNA-0 ALL

NCP>SET CIRCUIT UNA-0 SERVICE ENABLED STATE ON

NCP>LOAD NODE RTRDEV
```

Example 7 Loading the Communications Server Using the NCP LOAD Command

The LOAD command allows the user to override any of the following default parameters defined in the running down-line load database of the loading host:

- HOST node-id - a node other than the current host to receive logged events and up-line dumps generated by the server node. This node must be reachable from the loading host or be the loading host itself.
- FROM file-id - identifies the server software image file that the user wishes to load.
- SECONDARY LOADER, TERTIARY LOADER file-id - identifies the loader programs.
- PHYSICAL ADDRESS E-address - either the Ethernet hardware address or the extended DECnet node address of the server node.
- SERVICE PASSWORD hex-password - specifies the password required by the server node for starting the load operation.

NOTE

If the NCP LOAD command is issued while the TEST button on the server is in, then NCP ignores the state of the TEST button and sends the server software, and not the diagnostic image.

Using The TRIGGER Command

The TRIGGER command has the same effect as loading the server by pushing the start button. The TRIGGER command starts the bootstrap mechanism on the server which in turn sends a multicast request-to-load message to all nodes on the Ethernet. The TRIGGER command is issued in the following ways:

```
NCP>TRIGGER NODE server-node-id [SERVICE PASSWORD hex-password]
                                [PHYSICAL ADDRESS E-address]

                                or

      NCP>TRIGGER VIA circuit-id PHYSICAL ADDRESS E-address
                                   [SERVICE PASSWORD hex-password]
```

The TRIGGER command uses the information from the server node's down-line load database in the loading host. The TRIGGER VIA command is used from a host node on which the database has not been defined. The following procedures take place when the TRIGGER command is issued:

- The host where the command is issued sends the trigger to the server over the specified circuit.
- The server responds by sending a request-to-load message to ALL hosts on the Ethernet.
- The first host to respond, loads the server.

The Example 8 shows the TRIGGER command being used to load a Router Server (RTRDEV) over circuit UNA-0.

```
NCP>SET LINE UNA-0 ALL

NCP>SET CIRCUIT UNA-0 SERVICE ENABLED STATE ON

NCP>TRIGGER NODE RTRDEV
```

Example 8 Loading the Communications Server Using the NCP TRIGGER Command

NOTE

If the SET CIRCUIT SERVICE DISABLED command is used the host from which the command is issued can never be the loading host.
If the TRIGGER command is issued while TEST button on the server is in, then the host will load the Loadable Diagnostic Image instead of the server software.

Expected Event Messages

The loading process goes through the following three phases and may take up to several minutes to complete.

1. A primary loader, contained in the server's DEUNA ROM, loads the secondary loader (PLUTO2.SYS) into the DEUNA and causes it to begin executing.

2. The secondary loader loads the tertiary loader (PLUTO3.SYS) into server memory and starts the tertiary loader.

3. The tertiary loader loads the system image (CSVRTR.SYS) into server memory. The tertiary loader also passes host supplied parameters to the system image (server's node name, node number, and hosts node number). Finally the tertiary loader starts the system image.

During each phase of the load, the host issues two event messages. The first event message of the pair indicates the load phase that has been requested (eg. the secondary loader has been requested to load the tertiary loader). The second message of the pair indicates whether the load was successful or not. Table 3 contains a list of expected events during the loading of server software.

Table 3 Installation Event Message Classes

EVENT	MEANING
225	Errors that occur before and during the initialization of the network configuration file.
226	Errors that occur during initialization of the Router Server configuration file.
229	Errors that occur anytime after the server software is running. These include protocol errors as well as other uncommon events involving the server software.

Example 9 shows some typical events that can be expected during the loading and initialization of the server software.

```
DECnet event 0.3, automatic line service
From node 224 (BARNEY), 1-JUL-1983 10:59:03.55
Circuit UNA-0, Load, Requested, Node = 225 (WILMA)
File = PLUT02.SYS, Secondary loader


%%%%%%%%%%  OPCOM  1-JUL-1983  10:59:10.84
Message from user DECNET
DECnet event 0.3, automatic line service
From node 224 (BARNEY), 1-JUL-1983 10:59:03.55
Circuit UNA-0, Load, Requested, Node = 225 (WILMA)
File = PLUT02.SYS, Secondary Loader


%%%%%%%%%%  OPCOM  1-JUL-1983  10:59:19.68
Message from user DECNET
DECnet event 0.3, automatic line service
From node 224 (BARNEY), 1-JUL-1983 10:59:03.78
Circuit UNA-0, Load, Successful, Node = 225 (WILMA)
File = PLUT02.SYS, Secondary Loader
```

```
%%%%%%%%%%%  OPCOM  1-JUL-1983  10:59:28.37
Message from user DECNET
DECnet event 0.3, automatic line service
From node 224 (BARNEY), 1-JUL-1983 10:59:05.03
Circuit UNA-0, Load, Requested, Node = 225 (WILMA)
File = PLUTO3.SYS, Tertiary loader


%%%%%%%%%%%  OPCOM  1-JUL-1983  10:59:37.00
Message from user DECNET
DECnet event 0.3, automatic line service
From node 224 (BARNEY), 1-JUL-1983 10:59:06.41
Circuit UNA-0, Load, Successful, Node = 225 (WILMA)
File = PLUTO3.SYS, Tertiary loader

%%%%%%%%%%%  OPCOM  1-JUL-1983  10:59:45.65
Message from user DECNET
DECnet event 0.3, automatic line service
From node 224 (BARNEY), 1-JUL-1983 10:59:06.82
Circuit UNA-0, Load, Requested, Node = 225 (WILMA)
File = CSVRTR.SYS, Operating system

DECnet event 0.3, automatic line service
From node 224 (BARNEY), 1-JUL-1983 11:00:14.27
Circuit UNA-0, Load, Successful, Node = 225 (WILMA)
File = CSVRTR.SYS, Operating system

%%%%%%%%%%%  OPCOM  1-JUL-1983  11:00:20.72
Message from user DECNET
DECnet event 0.3, automatic line service
From node 224 (BARNEY), 1-JUL-1983 11:00:14.27
Circuit UNA-0, Load, Successful, Node = 225 (WILMA)
File = CSVRTR.SYS, Operating system

DECnet event 225.0
From node 225 (WILMA), 1-JUL-1983 11:00:45.76
Parameter #0 = Server initializing, Identification = RTR 1

DECnet event 4.15, adjacency up
From node 225 (WILMA), 1-JUL-1983 11:01:55.76
Circuit UNA-0, Adjacent node = 224 (BARNEY)

DECnet event 4.15, adjacency up
From node 225 (WILMA), 11:02:15.27
Circuit UNA-0, Adjacent node = 320 (FRED)
```

Example 9 Typical Event Messages During Server Initialization

If the software detects errors in the configuration files, the event message shows which command is in error as well as the nature of the error. The reported errors should be corrected and the software loaded again.

COMMUNICATIONS SERVER INITIALIZATION

After the system has been loaded, it starts the initialization process:

- RSX Executive starts
- Communications Executive starts
- NTINIT starts up the DECnetnode (turns the Executor on)
- PAM and UNA microcode is loaded
- UNA driver changes the UNA physical address from the hardware address to the extended DNA address
- LED task starts - displays the cyclic light pattern
- SBINIT starts
  - Waits for the network to come up
  - Issues event to the host - Server Initializing
  - Uses NICE to read node names from the host
  - Uses DAP to read <node>SB.CFG file from the host
  - Starts the server product initialization task

INSTALLATION AND CHECKOUT PROCEDURES

INSTALLATION CERTIFICATION PROCEDURES

Once the software is loaded without error and the server comes up, the next step is to perform the Installation Checkout Procedures (ICP). This is an indirect command file supplied with the server software that checks the connectivity between the server node and all it's adjacent nodes. The ICP is invoked using the following commands:

```
$@SYS$SYSROOT:[PLUTO]CSVRTRICP
```

or

```
>@ dev:[uic]CSVRTRICP
```

NOTE

The dev and [uic] refer to the device and UIC of the Pluto account.

There is a specific ICP designed for each of the server products. Figure 2 shows a DECnet Router configuration tested by CSVRTRICP.COM.

```
$@CSVRTRICP

$ SET NOVERIFY
!
!This procedure runs a loop test to each adjacent node to the
!DECnet Router Server and reports the status of the circuit.
!reference the System Managers Guide if errors are not familiar.
!
!You may edit this procedure to test any lines you wish.
!
$ON CONTROL_Y THEN EXIT
$INQUIRE NODE "Please enter the DECnet Router Server name"
$INQUIRE PASS "Please enter 'NODE' password
$
$SET VERIFY
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' SET NODE LC1 CIR LC-1
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' LOOP NODE LC1
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' CLE NODE LC1 CIR

$NCP TELL 'NODE' USER [1,1] PASS 'PASS' SET NODE LC2 CIR LC-2
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' LOOP NODE LC2
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' CLE NODE LC2 CIR

$NCP TELL 'NODE' USER [1,1] PASS 'PASS' SET NODE LC3 CIR LC-3
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' LOOP NODE LC3
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' CLE NODE LC3 CIR

$NCP TELL 'NODE' USER [1,1] PASS 'PASS' SET NODE LC4 CIR LC-4
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' LOOP NODE LC4
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' CLE NODE LC4 CIR

$NCP TELL 'NODE' USER [1,1] PASS 'PASS' SET NODE LC5 CIR LC-5
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' LOOP NODE LC5
$NCP TELL 'NODE' USER [1,1] PASS 'PASS' CLE NODE LC5 CIR
!
!$NCP TELL 'NODE' USER [1,1] PASS 'PASS' SET NODE LC6 CIR LC-6
!$NCP TELL 'NODE' USER [1,1] PASS 'PASS' LOOP NODE LC6
!$NCP TELL 'NODE' USER [1,1] PASS 'PASS' CLE NODE LC6 CIR
!
!$NCP TELL 'NODE' USER [1,1] PASS 'PASS' SET NODE LC7 CIR LC-7
!$NCP TELL 'NODE' USER [1,1] PASS 'PASS' LOOP NODE LC7
!$NCP TELL 'NODE' USER [1,1] PASS 'PASS' CLE NODE LC7 CIR
!
!$NCP TELL 'NODE' USER [1,1] PASS 'PASS' SET NODE LC8 CIR LC-8
!$NCP TELL 'NODE' USER [1,1] PASS 'PASS' LOOP NODE LC8
!$NCP TELL 'NODE' USER [1,1] PASS 'PASS' CLE NODE LC8 CIR
$EXIT
```

Example 10 ICP Test for DECnet Router Configuration

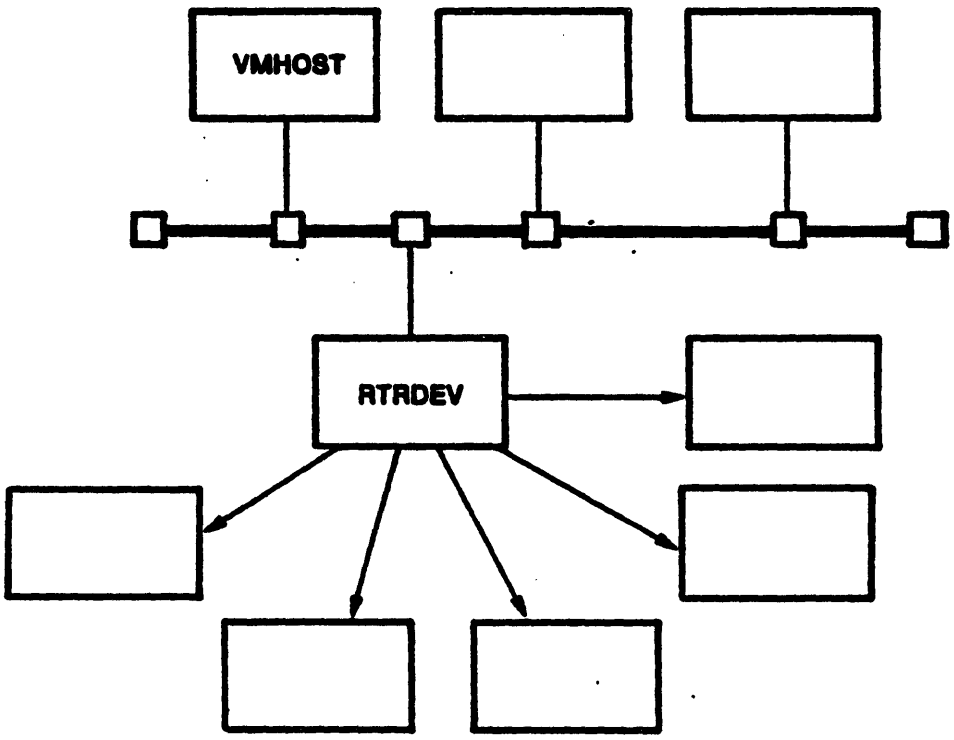


Figure 2 DECnet Router Configuration Tested by ICP Example

COPYING FILES TO BACKUP HOSTS

After checking that the server is working properly, the software images and files are copied to all the backup hosts that were specified in the network configuration file. These files must be copied to the proper place on the host system:

- o VMS backup hosts
  - Server system image, diagnostics system image, and the loader files reside in SYS$SYSTEM.
  - Configuration files and the ICP command file reside in the PLUTO PLUTO account.
- o RSX backup hosts
  - Server system image, diagnostics system image, and loader files reside in NETUIC or whatever server UIC has been chosen.
  - Configuration files and the ICP command file reside in the PLUTO/PLUTO account.

Copy the files to their respective hosts by using the VMS COPY or RSX NFT utilities.

Once the files have all been copied to their respective places the final steps in the installation are:

- o Set up the Down-line load/Up-line dump database on the backup hosts and on any host which will be used to accept a server dump.
- o Make sure that all copied files have protections set for WORLD READ
- o Set up the PLUTO account on all backup hosts

APPENDIX A

NETWORK CONFIGURATION FILE

```
;
;           Configuration File - 'server-id'SB.CFG
;

;
;1. Defining the UNA password (viewed by hosts as the service password).
;   Default if invalid or missing: NO PASSWORD SET

        SET LINE UNA-0 PASSWORD FFFF


;
;2. Defining the Network Verification password.
;   Default if invalid or missing: NO PASSWORD SET

        SET SYSTEM PASSWORD PRIV

;
;3. Setting the clock frequency to 50 Hz.
;   Default if invalid or missing: 60 Hz.
        SET SYSTEM LINE FREQUENCY 60


;
;4. Backup host node list.
;   Default: server's maintenance host

        SET EXECUTOR BACKUP HOSTS A,B,C,D
```

APPENDIX B

DECNET ROUTER SERVER CONFIGURATION FILE

```
;
; DECnet Router Server Initialization File
;
; Copyright (c) 1983 by
; Digital Equipment Corporation, Maynard, Mass.
;
; File name = "Server's node name"RTR.CFG
;

;
; This configuration file resides on the PLUTO/PLUTO account or
; General DECnet account of the downline loading host system. This
; file uses all the default values for configuring a DECnet Router
; Server node. Note all values are in decimal.
;

;
;       1.  Set the MAXIMUM ADDRESS in the DECnet network.  The legal
;           range for this parameter is 2 to 1023 inclusive. The
;           default value for the MAXIMUM ADDRESS parameter is 255.
;

          SET EXECUTOR MAXIMUM ADDRESS 255


;
;       2.  Set the maximum number of routing nodes (excluding this
;           node) on the Ethernet simultaneously. The legal range for
;           the BROADCAST ROUTERS parameter is 1 to 33 inclusive.  The
;           default value for this parameter is 10.
;

          SET EXECUTOR MAXIMUM BROADCAST ROUTERS 10


;
;       3.  Set the maximum number of nonrouting nodes on the  Ethernet
```

; simultaneously. The legal range for the BROADCAST
; NONROUTERS parameter is 0 to 1022 inclusive. The default
; value for this parameter is 32. The number of nonrouters
; will be set to the maximum address minus one if the number
; of nonrouters is greater than the maximum address minus
; one.

```
SET EXECUTOR MAXIMUM BROADCAST NONROUTERS 32
```

;
; 4. Set the total number of hops from the DECnet Router Server
; to any other reachable node in the network. A remote node
; is unreachable over a path if the number of hops required
; to get to it exceeds the value set for this parameter. The
; legal range for the HOPS parameter is 2 to 30 inclusive.
; The default value for this parameter is 20.
;

```
SET EXECUTOR MAXIMUM HOPS 20
```

;
; 5. Set the total cost from the DECnet Router Server to any
; other reachable node in the network. A remote node is
; unreachable over a path if the cost required to get to it
; exceeds the value set for this parameter. The legal range
; for the COST parameter is 2 to 1022 inclusive. The default
; value for this parameter is 1022.
;

```
SET EXECUTOR MAXIMUM COST 1022
```

;
; 6. Set the BUFFER SIZE for the DECnet Router Server. These
; buffers are used for the intermediate storage of all user
; data being transmitted or received. Legal range for the
; BUFFER SIZE parameter is 246 to 1484 inclusive. The
; default value for this parameter is 576.
;

```
SET EXECUTOR BUFFER SIZE 576
```

;
; 7. Set the SEGMENT BUFFER SIZE. This parameter allows nodes
; with different buffer sizes to communicate efficiently.
; The legal range for the SEGMENT BUFFER SIZE parameter is
; 246 to 1466 inclusive. The default value for this
; parameter is 576.

```
;
;
        SET EXECUTOR SEGMENT BUFFER SIZE 576


;
;       8.  Set the line characteristics (SPEED, DUPLEX, MODEM control)
;           for each point-to-point connection from the DECnet Router
;           Server.  The options for each parameter are the following:
;
;           The speed range varies depending on your configuration.
;           Check the DECnet Router Server Installation guide for the
;           proper values.  The total bandwidth for the point-to-point
;           lines must not exceed 500 kilobaud.
;
;           The duplex parameter for the line can be DUPLEX FULL or
;           DUPLEX HALF.
;
;           The modem control parameter can be MODEM YES or MODEM NO.
;
;           The default values for these parameters are SPEED 9600,
;           DUPLEX FULL and MODEM NO.  The legal devices are LC-1
;           through LC-8 and LC-*.
;

        SET LINE LC-1 SPEED 9600 DUPLEX FULL MODEM YES
        SET LINE LC-2 SPEED 9600 DUPLEX FULL MODEM YES
        SET LINE LC-3 SPEED 9600 DUPLEX FULL MODEM YES
        SET LINE LC-4 SPEED 9600 DUPLEX FULL MODEM YES
        SET LINE LC-5 SPEED 9600 DUPLEX FULL MODEM YES
        SET LINE LC-6 SPEED 9600 DUPLEX FULL MODEM YES
        SET LINE LC-7 SPEED 9600 DUPLEX FULL MODEM YES
        SET LINE LC-8 SPEED 9600 DUPLEX FULL MODEM YES

;
;       9.  Set the circuit characteristics (COST, HELLO TIMER, STATE)
;           for the point-to-point circuits from the DECnet Router
;           Server.  The legal range for the cost parameter is 1 to 25
;           inclusive.  This sets the path cost between the DECnet
;           Router and the adjacent node on the circuit.  The legal
;           range for the HELLO TIMER parameter is 3 to 8191.  The
;           HELLO TIMER is the rate in seconds that idle transport
;           messages are sent to maintain reachability. The options for
;           the state parameter are STATE ON or STATE OFF.  The default
;           values for COST, HELLO TIMER and STATE are 1, 15 seconds
;           and off respectively.
;

        SET CIRCUIT LC-1 COST 1 HELLO TIMER 15 STATE OFF
        SET CIRCUIT LC-2 COST 1 HELLO TIMER 15 STATE OFF
        SET CIRCUIT LC-3 COST 1 HELLO TIMER 15 STATE OFF
        SET CIRCUIT LC-4 COST 1 HELLO TIMER 15 STATE OFF
```

```
SET CIRCUIT LC-5 COST 1 HELLO TIMER 15 STATE OFF
SET CIRCUIT LC-6 COST 1 HELLO TIMER 15 STATE OFF
SET CIRCUIT LC-7 COST 1 HELLO TIMER 15 STATE OFF
SET CIRCUIT LC-8 COST 1 HELLO TIMER 15 STATE OFF
```

;
; 10. Set the circuit UNA-0 parameters (COST, ROUTING PRIORITY,
; HELLO TIMER). The COST and HELLO TIMER parameters have the
; same attributes and defaults as described above for LC
; circuits. The ROUTING PRIORITY parameter designates which
; router will serve the Ethernet. If two routing nodes have
; the same priority, then the node with the higher node
; address will be the designated router for the Ethernet.
; The legal range for the ROUTING PRIORITY parameter is 0 to
; 127 inclusive. The default for this parameter is 100.
;

```
SET CIRCUIT UNA-0 COST 1 ROUTING PRIORITY 100 HELLO TIMER 15
```

;
; 11. Set the TRANSMIT PASSWORD, which the executor node must
; transmit to an adjacent node during a node initialization
; sequence. The password is one to eight ASCII characters.
; The default for this parameter is no password.
;
; SET EXECUTOR TRANSMIT PASSWORD DECNET

;
; 12. Set the RECEIVE PASSWORD, which the executor node expects
; to receive from an adjacent node during a node
; initialization sequence. The password is one to eight
; ASCII characters. The default for this parameter is no
; password.
;
; SET EXECUTOR RECEIVE PASSWORD DECNET
;

CONTENTS

FIGURES

INTERNALS

INTRODUCTION

The Communications server hardware consists of:

- A dedicated PDP-11/24
- 512 KB of memory
- The Protocol Assist Module (PAM)
- Appropriate line cards
- DIGITAL Ethernet to UNIBUS Adapter (DEUNA)
- Console Boot Terminator (CBT)

The PAM provides the interface between a terminal, remote DECnet node or PPSN and the Communications server. The PAM moves characters between line cards and server memory. It can transmit or receive characters using a number of data link protocols including DDCMP and HDLC.

The Communications Server Software is made up of two basic parts: the Server Base and given server product software. The Server base is a set of components common to all server products; it provides the foundation on which the products are built. The Server Base implements generic DNA Phase IV, including a Communications Executive (CEX), capabilities for down-line loading, up-line dumping, and maintenance facilities. The server software is layered on top of this base forming the server product. This product is distributed as a system image, installed on an Ethernet host, modified by means of a configuration file, and down-line loaded into the communications server.

The DEUNA provides the interface between the Communications server and the Ethernet. It is responsible for receiving and transmitting data to or from an Ethernet host.

This module covers the major components, data structures, and data flow for the Communications server software, and the DECnet Router server. It is divided into 3 sections describing the Server Hardware (PAM), Server Base, and the DECnet Router server product.

INTERNALS

COMMUNICATIONS SERVER HARDWARE

- PDP-11/24 with 512 KB memory
- DEUNA - to connect to the Ethernet
- PAM (Protocol Assist Module)
  - Intelligent interface
  - Supports DDCMP, HDLC, and asynchronous protocol for terminals
  - Supports up to eight line cards

    M3100 - 1 synchronous line, up to 19.2 KB

    M3101 - 1 synchronous line, up to 500 KB

    M3102 - 2 asynchronous lines, up to 19.2 KB

- Extended Unibus to allow 22-bit addressing by the PAM
- No Unibus Mapping Registers
- No disk or console terminal
- LED on the front panel used for diagnostics, fault indications, and node identification

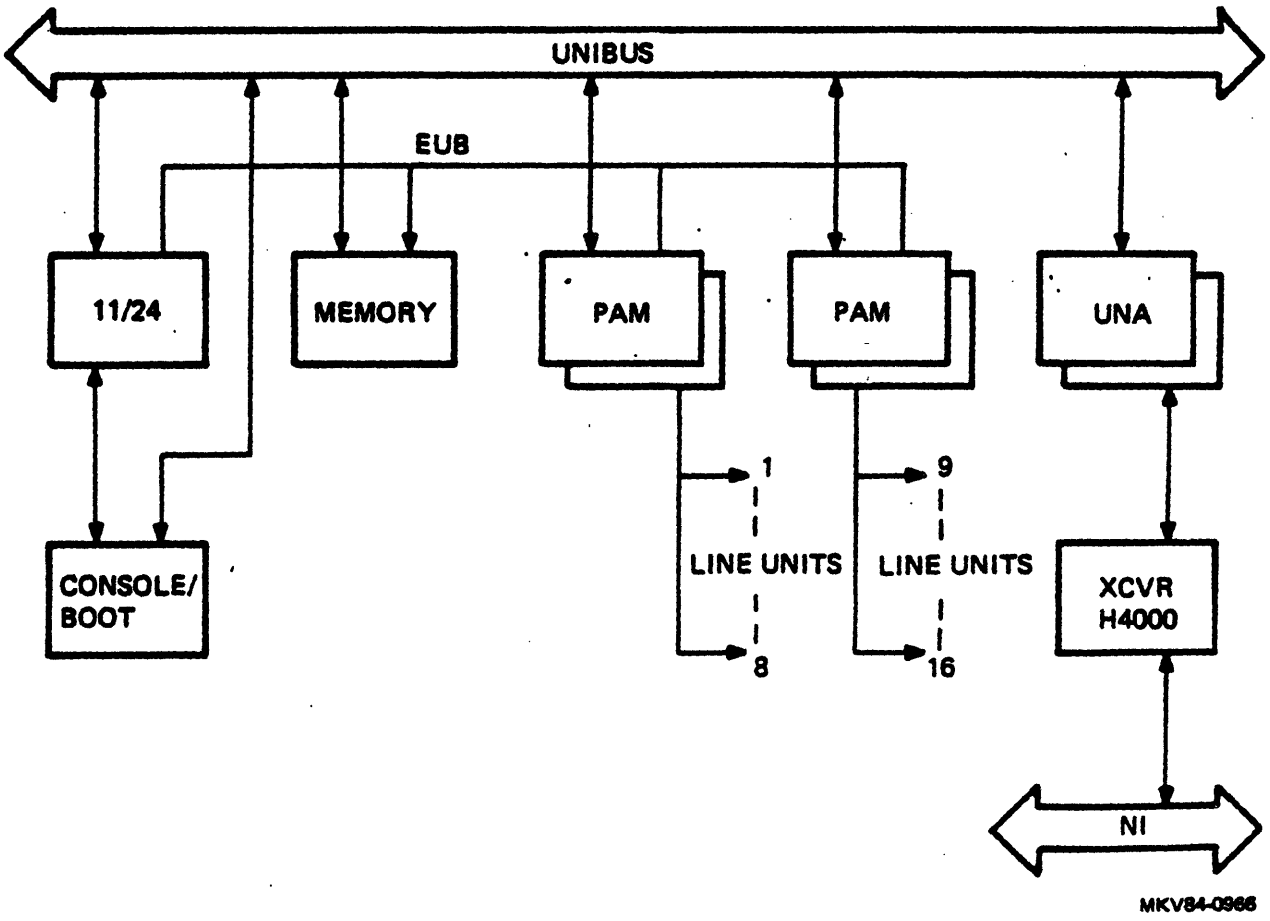


Figure 1 Communications Server Hardware Components

PROTOCOL ASSIST MODULE (PAM)

- Moves characters between the line cards and server memory.
- Assembles received messages into receive buffers in server memory.
- Transmits/receives messages in any one of the following protocol types:
  - Asynchronous (for terminals)
  - DDCMP
  - HDLC
- The Pam is loaded with either the terminal or the protocol microcode.
- A specific PAM line is limited to only one protocol at any particular time.

PAM Operation

- Receiving data
  - Characters enter the line cards one character at a time.
  - The line cards are scanned by the PAM hardware, the characters are moved into a silo buffer, and the PAM microcode is notified.
  - Protocol header is stripped off on protocol lines and the CRC is checked.
  - The microcode moves the data from the silo into receive buffers (RDBs) in server memory (done by DMA).
  - Each RDB is pointed to by a Receive Segment Descriptor.

    There are 16 Receive Segment Descriptors that are arranged into a ring buffer.

    Each segment descriptor points to one of the RDBs allocated for the PAM.

    Receive Segment Descriptors contains pointers to a particular RDB, status information, and the number of bytes placed into the buffer.

- Once the RDB is full, or the message is completed, the status is set in the segment descriptor for that RDB (RCVCMP).

- The PAM microcode generates a "receive complete" (RCVCMP) to the next level process.

o Transmitting data

- The PAM microcode transfers data, 1 word at a time, from transmit buffers (LDBs) in server memory into a PAM buffer (local store) via DMA.

- From the local store, the characters are moved to the line cards, the protocol headers are added, and the packet is transmitted.

NOTE

There are two network buffers allocated for each line using the M3100 or M3102 (19.2 KB) line cards and four network buffers allocated for each line using the M3101 (500 KB) line card.

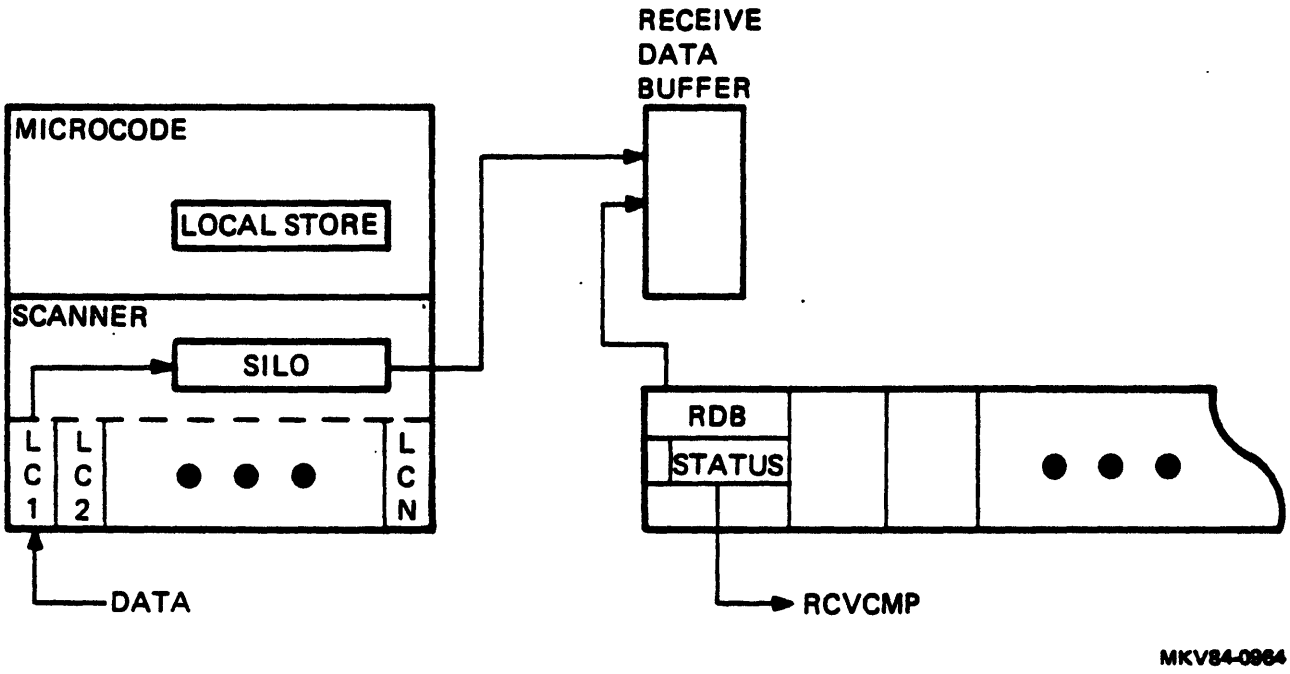


Figure 2 Integration of PAM Components

SERVER BASE

The server base is made up of an RSX-11S system with DECnet capabilities.

CEXPAR	111734	00112000	00006000	MAIN COM
NTPOOL	111670	00120000	00470000	MAIN SYS
SBPOOL	075300	00120000	00341400	SIB DYNAMIC
EXCOM1	111624	00610000	00010100	MAIN COM
EXCOM2	111560	00620100	00005400	MAIN COM
PAMMC0	111514	00625500	00021000	MAIN COM
PAMMC1	111450	00646500	00035000	MAIN COM
UNAMC	111404	00703500	00001000	MAIN COM
GEN	111340	00704500	01073300	MAIN SYS
	111004	00704500	00004500	SUB (NTINIT)
	110414	00711200	00006100	SUB (MLD...)
	110024	00717300	00037400	SUB (EVC...)
	107450	00756700	00235000	SUB (NMVACP)
	107244	01213700	00000700	SUB DRIVER - NM:
	106720	01214600	00022100	SUB (NETACP)
	106330	01236700	00002300	SUB (LED...)
	105720	01241200	00007200	SUB (RED...)
	105340	01250400	00011300	SUB (NIC...)
	104750	01261700	00030300	SUB (LOO...)
	104360	01312200	00007600	SUB (MIR...)
	103770	01322000	00040300	SUB (SBI...)
POOL..	103670	01362300	00031400	SUB DYNAMIC
NT.AUX	075234	01413700	00004000	SUB DYNAMIC
NT.EVL	074564	01417700	00004500	SUB DYNAMIC
NT.ECL	074520	01424400	00015000	SUB DYNAMIC
NT.XPT	074244	01441400	00012400	SUB DYNAMIC
NT.DTR	074034	01454000	00004000	SUB DYNAMIC
NT.DLX	073260	01460000	00012500	SUB DYNAMIC
NT.EPM	072734	01472500	00006300	SUB DYNAMIC
NT.UNA	072670	01501000	00010700	SUB DYNAMIC
NT.APC	072154	01511700	00003700	SUB DYNAMIC
NT.PAM	072110	01515600	00020000	SUB DYNAMIC

Figure 3 Communications Server Memory Layout (Server Base)

Communications Executive (CEX)

- o A set of modules used to help run the communication processes.
- o In the communications server, it resides in CEXPAR (DSR space)
- o Provides:
  - Interprocess communication routines that allow processes to communicate with each other.
  - Process scheduling routines that allow DECnet processes to schedule other DECnet processes for execution.
  - Buffer management routines that allocate/deallocate main memory buffers from available buffer pools.
  - Timer support routines that provide processes with logical timers.
  - Modem control routines that support modem handling for lower level processes (actually resident in AUX).
  - Common routines used by the communication processes.

System Level Interface (SLI)

- o Allows processes to schedule one another and convey data buffers.
- o Utilizes the interprocess communications mechanisms of the CEX.
- o Functions incude:
  - Request a logical link
  - Accept a logical link
  - Send data
- o Used in place of the standard QIO interface provided by RSX.

Communication Processes

LLC Processes

- NT.AUX (Auxiliary Process)
  - Non-time-critical extension of the CEX
  - Functions:
    - Buffer allocation failure recovery
    - Powerfail recovery
    - Modem control
    - Timer services
    - Fork processing and process dispatch
    - Allocation/deallocation of buffers from network pool (NTPOOL)

- NT.ECL (End Communications Layer)
  - Accepts QIO requests from user tasks for logical links (NS: Driver)
  - Processes time critical requests such as SND$, REC$, and XMI$ (all others are sent to NETACP for processing)
  - Contains time critical code for logical link processing
  - Implements the NSP protocol

- NT.XPT (Routing Process)
  - Provides DECnet routing functions
    - Checks if the selected channel is broadcast or non broadcast
    - Checks visit counts
    - Builds routing header

- NT.DTR
  - System Level Interface version of DTR
  - Used for running DTS tests
- NT.EVL (Event Logging Process)
  - Creates records of network events
  - Sets up event buffers
  - Dispatches the events to sink nodes
  - Sends events to EVC...
- NT.DLX (Direct Line Access)
  - Accepts and processes QIO requests from user tasks for DLX access (NX: Driver)

DLC Processes

- NT.APC
  - Auxiliary PAM Control process
  - Used when PAM lines are running DDCMP
  - Provides compatibility for HDLC and asynchronous terminals that use a software DLC. The APC provides PAM with someplace to go when the lines are running DDCMP (which PAM handles in microcode).
- NT.EPM (Ethernet Protocol Manager)
  - Protocol multiplexer
  - Controls the users communicating on the Ethernet
  - Uses protocol address pairs to identify who is communicating and what type of protocol is being used.

DDM Processes

- I/O driver for a communication device
- Executes as a result of either a device interrupt or an I/O request from a DLC
- Primary functions:
  - Starting and stopping the device
  - Giving the device buffers to receive and transmit
  - Retrieving/setting device characteristics
  - Terminating the current operation on the device
  - Providing special notifications to higher level processes
- NT.UNA
  - DEUNA interface to the Ethernet
- NT.PAM
  - Provides PAM support for asynchronous terminals, DDCMP, and HDLC

PAM MICROCODE

UNAMC (UNA Microcode)

- UNA ECO microcode
- Loaded with the help of AUX and MLD by drivers
- Loaded after the CEX enables the DEUNA
- Not microcode but rather a common area for the microcode

PAMMC1

- The synchronous PAM microcode
- Loaded with the help of AUX by drivers
- Loaded after the first "enable" is sent

MAJOR TASKS

NTINIT (Network Initializer)

- Used to start up the network after a down-line load
- Sets down-line load parameters into database
- Enables lines and circuits

MLD... (General Microcode Loader)

- Supports PAM and UNA microcode
- Calls MLP; the PAM microcode loader to perform device specific parts of the microcode loading

EVC... (Event Collector)

- Server task for the EVL process
- Dispatches event records to a console or monitor terminal, residing on a host that is on or off the Ethernet.

NMVACP (Network Management Volatile ACP)

- Executes NCP or Nice commands as requested by NM:
- Makes changes to volatile database only

NMDRV (Network Management Device Driver - NM:)

- Provides initial checking for given NCP or NICE requests
- Gives valid requests to NMVACP for processing

NETACP (DECnet Network ACP)

- Works with NS: (ECL)
- Processes user QIO requests from NS: that are not considered time critical (OPN$, CLS$, CON$, DSC$, ACC$, etc)
- Handles incoming calls by requesting tasks for them
- Notifies user tasks when network events occur
- Contains session control routines
- Contains line control routines for routing (line stop, MOP message received dispatch, etc.)

LED...

- Implements the LED display on the Server

RED... (Remote Examine and Deposit)

- Server task for the Server On-line Debugging tool (POD)
- Resides on the Server

NIC... (Network Information and Control Executor)

- Serves network management requests from remote nodes
- Implements the NICE protocol

LOO..., MIR...

- Network Management Loop and Mirror tasks

SBI... (Server Base Initializer)

- Server Base Initializing task
- Defines logical line card names (eg. LC-1 for PAM-0-0)
- Sets up the server's node database from the host using NICE protocol (equivalent to NCP> TELL $HOST SHOW KNOWN NODES)
- Reads the network configuration file <node>SB.CFG file on the host (using Network File Access Routines)
- Starts the Server Product Initializer task

SPI... (Server Product Initializer)

- **Sets up the Server software configuration**
- **Reads the server product configuration file (<node>xxx.CFG)**

BUFFERS

POOL.. (Byte/Block Pool)

- Located in GEN
- Contains data structures for DECnet
  - Logical Link Tables
  - Adjacency Database
  - Object blocks
  - Alias lists
  - Remote name blocks
  - Node counters
  - Minhop/Mincost vectors
- Contains databases that do not need to be accessed by the UNA directly

NTPOOL (Network Pool)

- SBPOOL - subpartition name of NTPOOL
- Contains the DECnet Buffers
  - Large Data Buffers (LDBs)
  - Small Data Buffers (SDBs)
- Can be accessed by the UNA using 18-bit addressing (resides below 124 K)

SDB (Small Data Buffers)

- Used for protocol or task control messages
- Obtained from NTPOOL

LDB (Large/Receive Data Buffers)

- "System Buffers" for data communication over the network
- Each LDB is pointed to by a corresponding CCB

CCB (Communication Control Blocks)

- **Obtained from available space in CEXPAR or DSR**
- **Used for interprocess communication**

INTEGRATION OF COMPONENTS

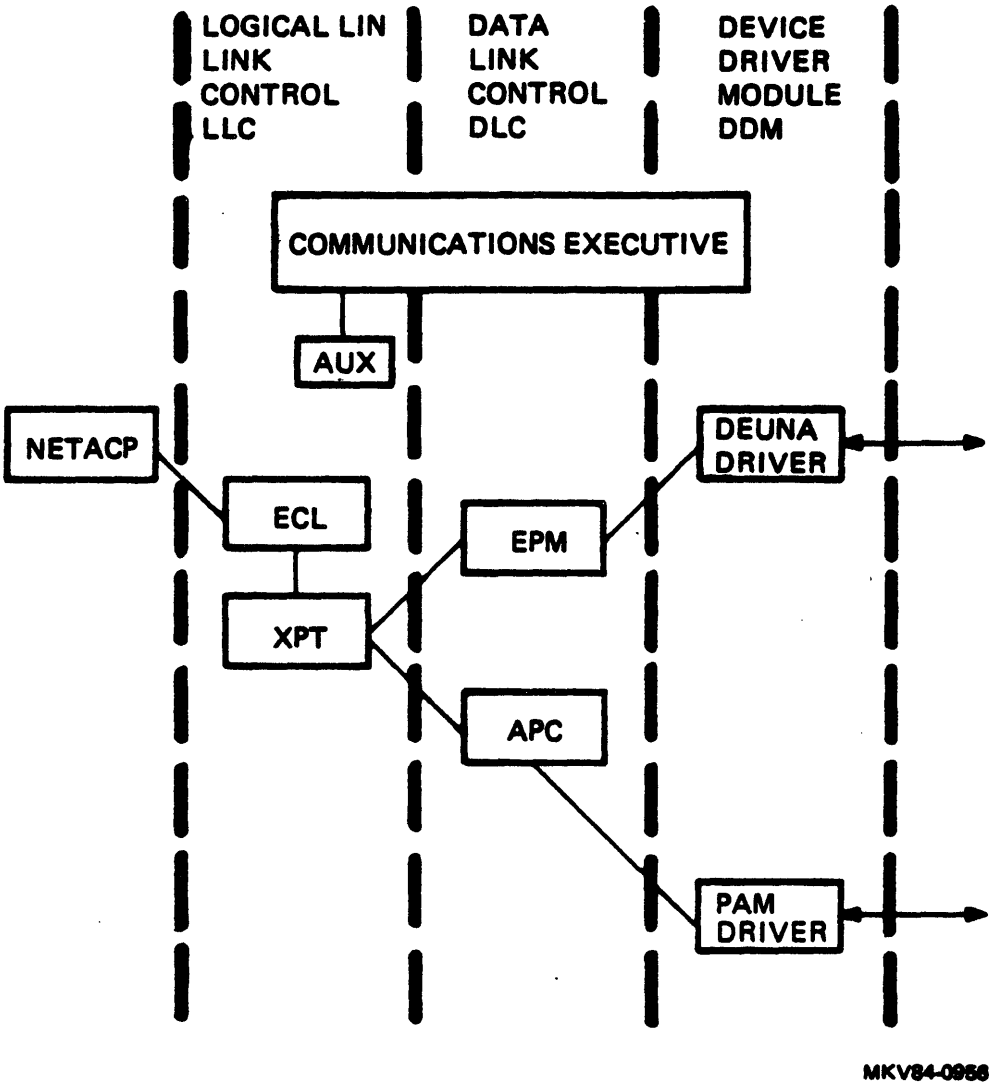


Figure 4 Integration of Server Base Components

COMMUNICATIONS SERVER DATA STRUCTURES

Communications Executive Common Database (CEXCM)

The CEXCM database:

- Contains information and pointers to all of the data structures important to the Communications Executive (CEX) and the various network components.
- Resides in the CEX.
- Actual initialization occurs during the network management "SET SYSTEM ALL" command.

Some important pointers found in CEXCM are:

- $PDVTA
  - Pointer to the Process Descriptor Vector Table.
  - This table contains an index for all of the DECnet processes specified.
  - Each entry (index) points to a Process Descriptor Vector (PDV) for the process.
  - The PDV contains:
    Pointers to the process' dispatch table
    Pointers to the process' database
    The name of the process in radix 50

- $SLTMA
  - Pointer to the System Line Vector Table.
  - This table contains pointers to the System Line Tables (SLT) set up for each physical line.
  - Each entry points to an SLT for the line.
  - The SLT is used between a DLC and a corresponding DDM.
  - The SLT contains pointers to:
    DDM Line Table
    DLC Line Table

- $LLCTA
  - Pointer to the Reverse Mapping Table.
  - The Reverse Mapping Table contains the correspondence between an LLC (eg. XPT) and its channel.

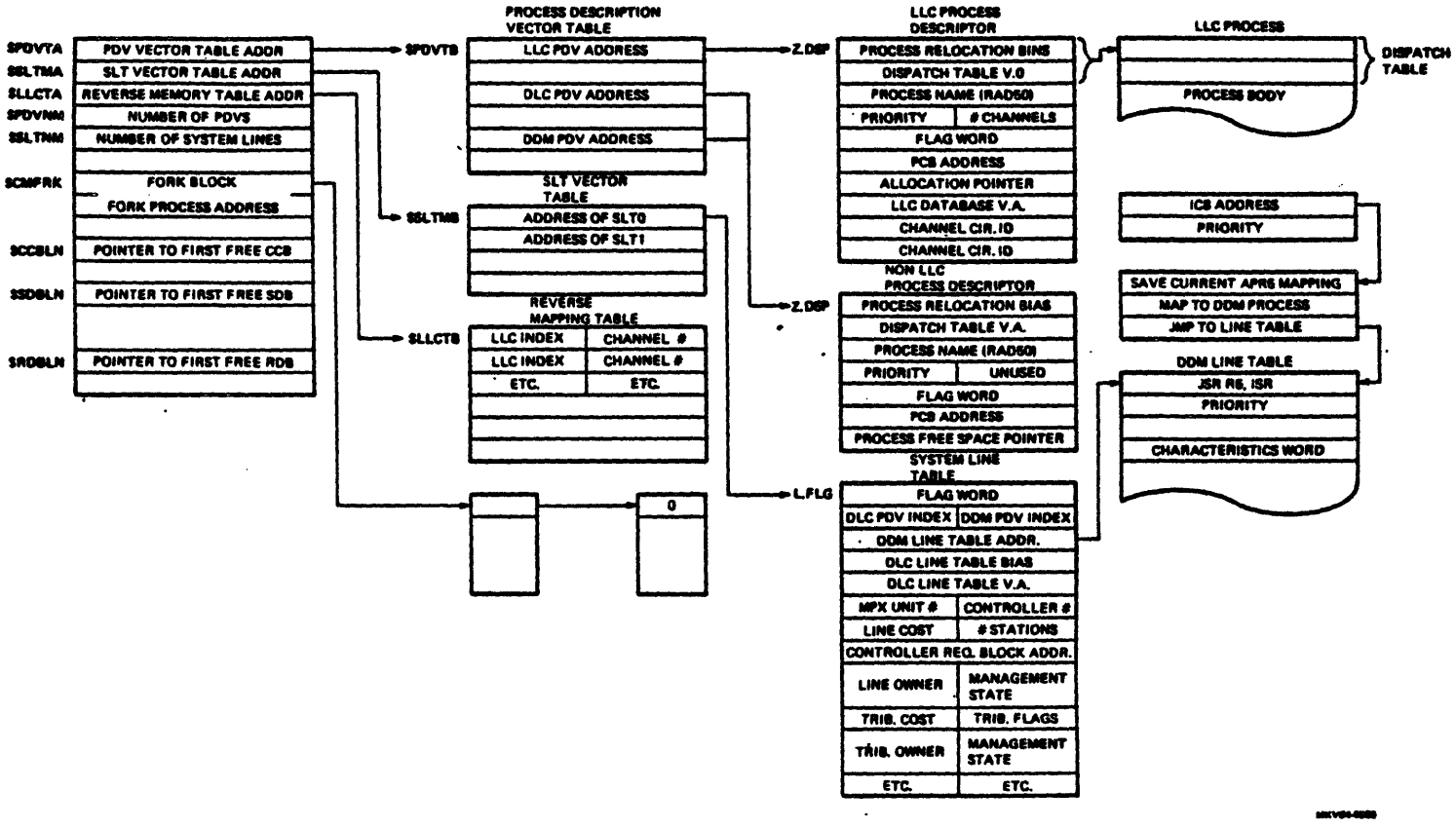


Figure 5 Communications Executive Database (CEXCM)

DECnet ROUTER SERVER

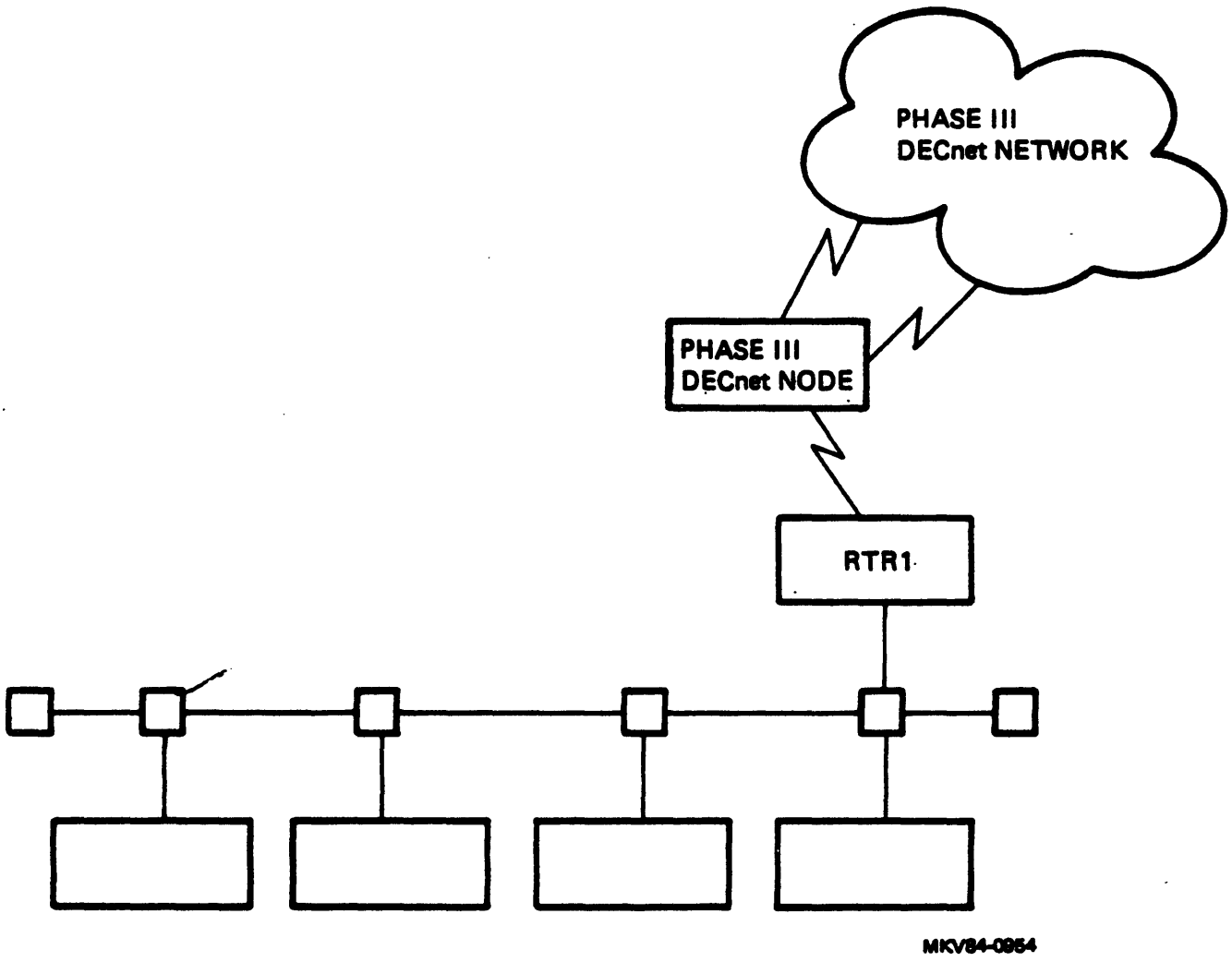


Figure 6 Ethernet Configuration Utilizing the Router Server

DECnet ROUTER SOFTWARE

- Server Base
- Router specific code
- System image is down-line loaded from an Ethernet host
- Network Management functions are performed from the Host node
- No SYSGEN or NETGEN necessary
- Network fine tuning is done by modifying the configuration files
- Up-line dump to the loading host in the event of a system crash

MAJOR COMPONENTS (as distributed)

- Router system image to be down-line loaded
- Configuration files for tuning
  - Server Base Configuration File (<node-id>SB.CFG)
  - Router Configuration File (<node-id>RTR.CFG)

```
CEXPAR   111734   00112000   00006000   MAIN COM
NTPOOL   111670   00120000   00500000   MAIN SYS
SBPOOL   073554   00120000   00364000   SUB DYNAMIC
EXCOM1   111624   00620000   00010100   MAIN COM
EXCOM2   111560   00630100   00005400   MAIN COM
PAMMC1   111514   00635500   00034000   MAIN COM
UNAMC    111450   00671500   00001000   MAIN COM
GEN      111404   00672500   01105300   MAIN SYS
         111050   00672500   00005000   SUB  (NTINIT)
         110460   00677500   00006100   SUB  (MLD...)
         110070   00705600   00036400   SUB  (EVC...)
         107514   00744200   00215000   SUB  (NMVACP)
         107310   01161200   00000700   SUB  DRIVER - NM:
         106764   01162100   00022300   SUB  (NETACP)
         106410   01204400   00176100   SUB  (RCP1..)
         106020   01402500   00002300   SUB  (LED...)
         105410   01405000   00007000   SUB  (RED...)
         105030   01414000   00011700   SUB  (NIC...)
         104440   01425700   00021400   SUB  (LOO...)
         104050   01447300   00001600   SUB  (MIR...)
         103460   01451100   00035700   SUB  (SBI...)
         103050   01507000   00037100   SUB  (SPI...)
POOL..   102770   01546100   00050700   SUB  DYNAMIC
NT.AUX   073510   01617000   00003700   SUB  DYNAMIC
NT.EVL   073340   01622700   00005600   SUB  DYNAMIC
NT.ECL   073264   01630500   00014700   SUB  DYNAMIC
NT.XPT   073010   01645400   00020000   SUB  DYNAMIC
NT.DTR   072600   01665400   00004000   SUB  DYNAMIC
NT.DLX   072024   01671400   00011000   SUB  DYNAMIC
NT.EPM   070714   01702400   00006500   SUB  DYNAMIC
NT.UNA   070650   01711100   00010200   SUB  DYNAMIC
NT.APC   070114   01721300   00003100   SUB  DYNAMIC
NT.PAM   070050   01724400   00015400   SUB  DYNAMIC
```

Figure 7 DECnet Router System Image

DECnet Router Major Tasks and Processes

- RCP1.. (Routing Control Processor)
  - Maintains the routing, forwarding, and adjacency databases
  - Monitors circuits for down-line loading, up-line dumping, and circuit loopback tests
- SPI...
  - Server Product Initializing task
  - Reads the <node>RTR.CFG file on the host

```
CEXPAR 111734 00112000 00006000 MAIN COM
NTPOOL 111670 00120000 00470000 MAIN SYS
SBPOOL 075300 00120000 00341400 SUB  DYNAMIC
EXCOM1 111624 00610000 00010100 MAIN COM
EXCOM2 111560 00620100 00005400 MAIN COM
PAMMC0 111514 00625500 00021000 MAIN COM
PAMMC1 111450 00646500 00035000 MAIN COM
UNAMC  111404 00703500 00001000 MAIN COM
GEN    111340 00704500 01073300 MAIN SYS
       111004 00704500 00004500 SUB  (NTINIT)
       110414 00711200 00006100 SUB  (MLD...)
       110024 00717300 00037400 SUB  (EVC...)
       107450 00756700 00235000 SUB  (NMVACP)
       107244 01213700 00000700 SUB  DRIVER - NM:
       106720 01214600 00022100 SUB  (NETACP)
       106330 01236700 00002300 SUB  (LED...)
       105720 01241200 00007200 SUB  (RED...)
       105340 01250400 00011300 SUB  (NIC...)
       104750 01261700 00030300 SUB  (LOO...)
       104360 01312200 00007600 SUB  (MIR...)
       103770 01322000 00040300 SUB  (SBI...)
POOL.. 103670 01362300 00031400 SUB  DYNAMIC
NT.AUX 075234 01413700 00004000 SUB  DYNAMIC
NT.EVL 074564 01417700 00004500 SUB  DYNAMIC
NT.ECL 074520 01424400 00015000 SUB  DYNAMIC
NT.XPT 074244 01441400 00012400 SUB  DYNAMIC
NT.DTR 074034 01454000 00004000 SUB  DYNAMIC
NT.DLX 073260 01460000 00012500 SUB  DYNAMIC
NT.EPH 072734 01472500 00006300 SUB  DYNAMIC
NT.UNA 072670 01501000 00010700 SUB  DYNAMIC
NT.APC 072154 01511700 00003700 SUB  DYNAMIC
NT.PAM 072110 01515600 00020000 SUB  DYNAMIC
```

```
CEXPAR 111734 00112000 00006000 MAIN COM
NTPOOL 111670 00120000 00500000 MAIN SYS
SBPOOL 073554 00120000 00364000 SUB  DYNAMIC
EXCOM1 111624 00620000 00010100 MAIN COM
EXCOM2 111560 00630100 00005400 MAIN COM
PAMMC1 111514 00635500 00034000 MAIN COM
UNAMC  111450 00671500 00001000 MAIN COM
GEN    111404 00672500 01105300 MAIN SYS
       111050 00672500 00005000 SUB  (NTINIT)
       110460 00677500 00006100 SUB  (MLD...)
       110070 00705600 00036400 SUB  (EVC...)
       107514 00744200 00215000 SUB  (NMVACP)
       107310 01161200 00000700 SUB  DRIVER - NM
       106764 01162100 00022300 SUB  (NETACP)
       106410 01204400 00176100 SUB  (RCP1..)
       106020 01402500 00002300 SUB  (LED...)
       105410 01405000 00007000 SUB  (RED...)
       105030 01414000 00011700 SUB  (NIC...)
       104440 01425700 00021400 SUB  (LOO...)
       104050 01447300 00001600 SUB  (MIR...)
       103460 01451100 00035700 SUB  (SBI...)
       103050 01507000 00037100 SUB  (SPI...)
POOL.. 102770 01546100 00050700 SUB  DYNAMIC
NT.AUX 073510 01617000 00003700 SUB  DYNAMIC
NT.EVL 073340 01622700 00005400 SUB  DYNAMIC
NT.ECL 073264 01630500 00014700 SUB  DYNAMIC
NT.XPT 073010 01645400 00020000 SUB  DYNAMIC
NT.DTR 072600 01665400 00004000 SUB  DYNAMIC
NT.DLX 072024 01671400 00011000 SUB  DYNAMIC
NT.EPH 070714 01702400 00006500 SUB  DYNAMIC
NT.UNA 070650 01711100 00010200 SUB  DYNAMIC
NT.APC 070114 01721300 00003100 SUB  DYNAMIC
NT.PAM 070050 01724400 00015400 SUB  DYNAMIC
```

Figure 8 Server Base and DECnet Router System Image Comparison

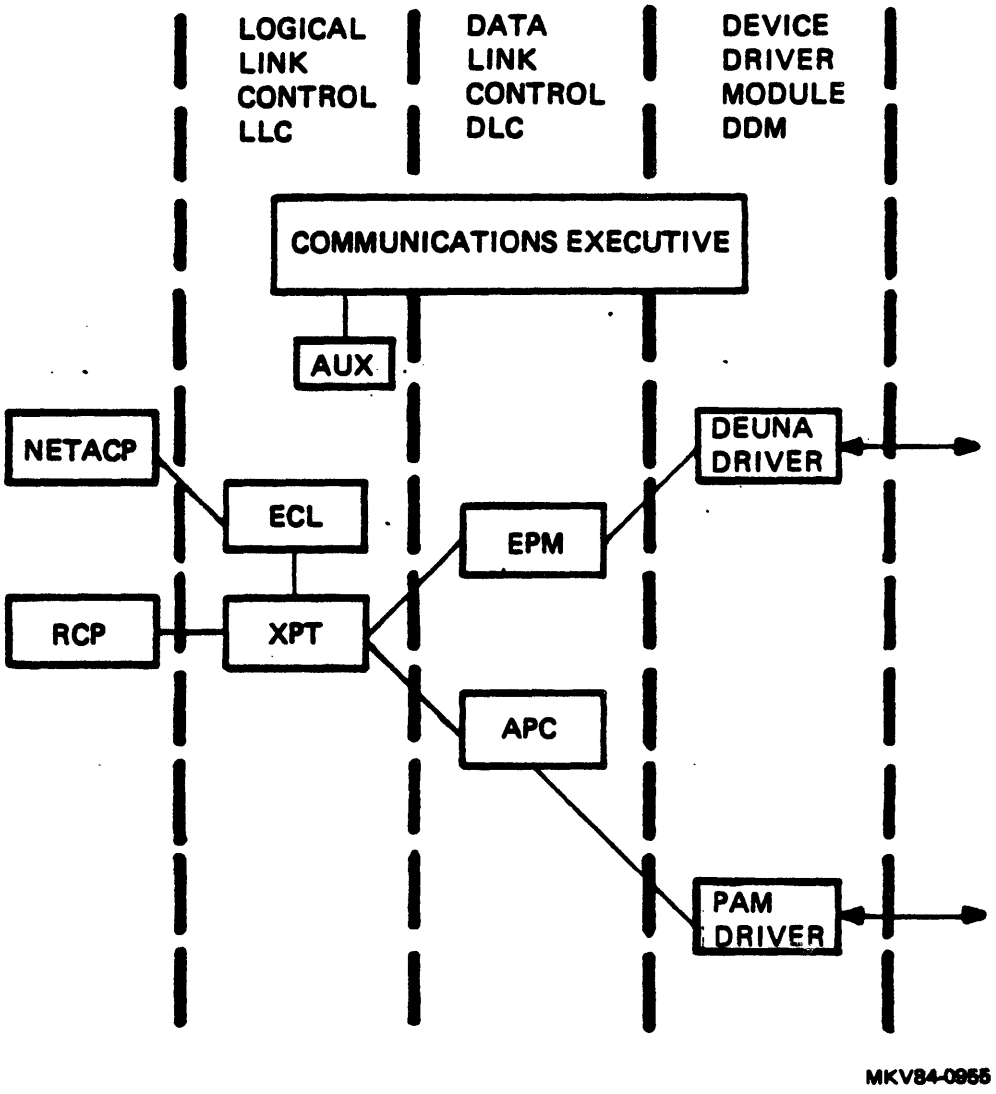


Figure 9 Integration of DECnet Router Server Components

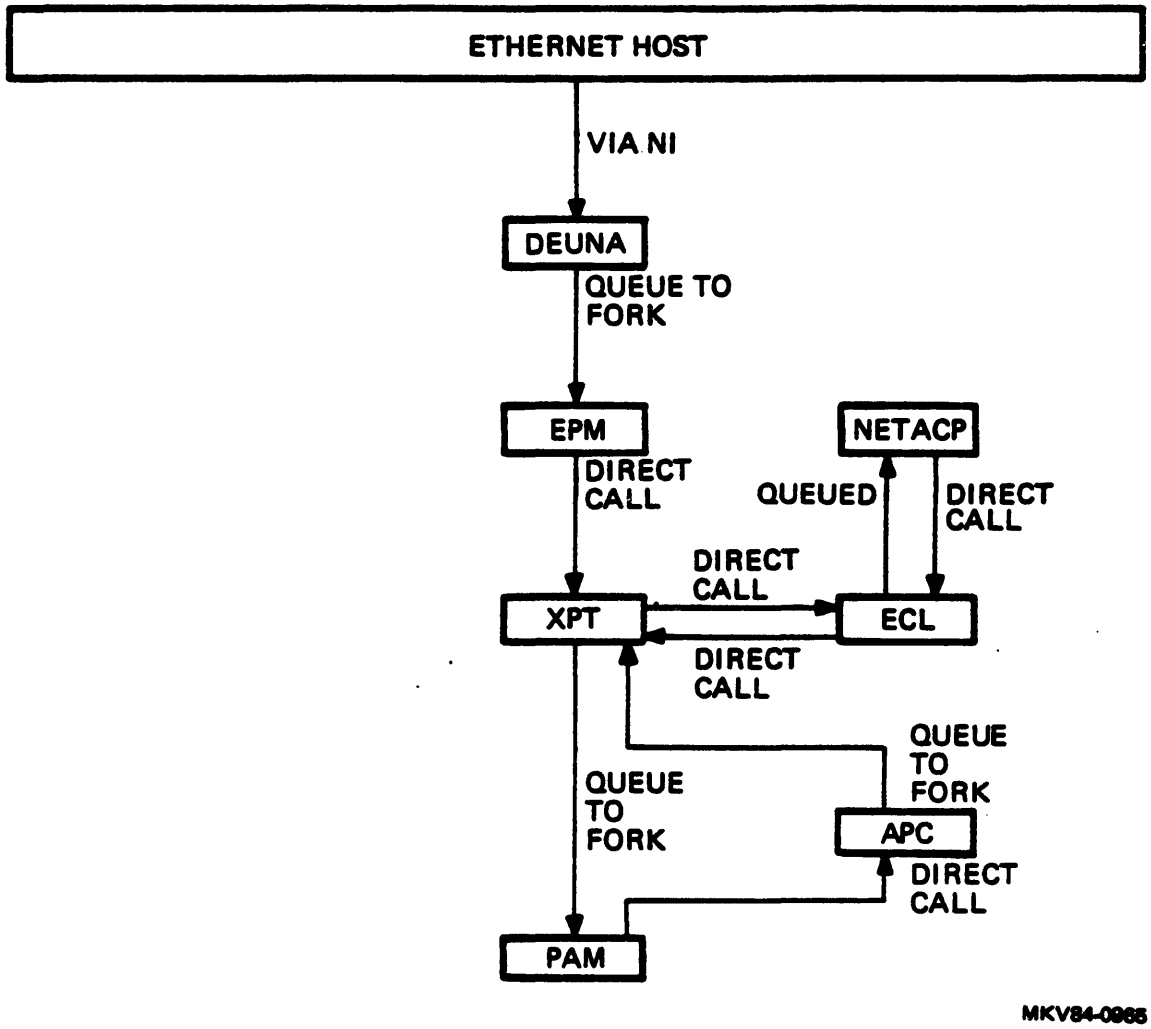


Figure 10 General Data Flow for the DECnet Router Server

MAJOR DATA STRUCTURES

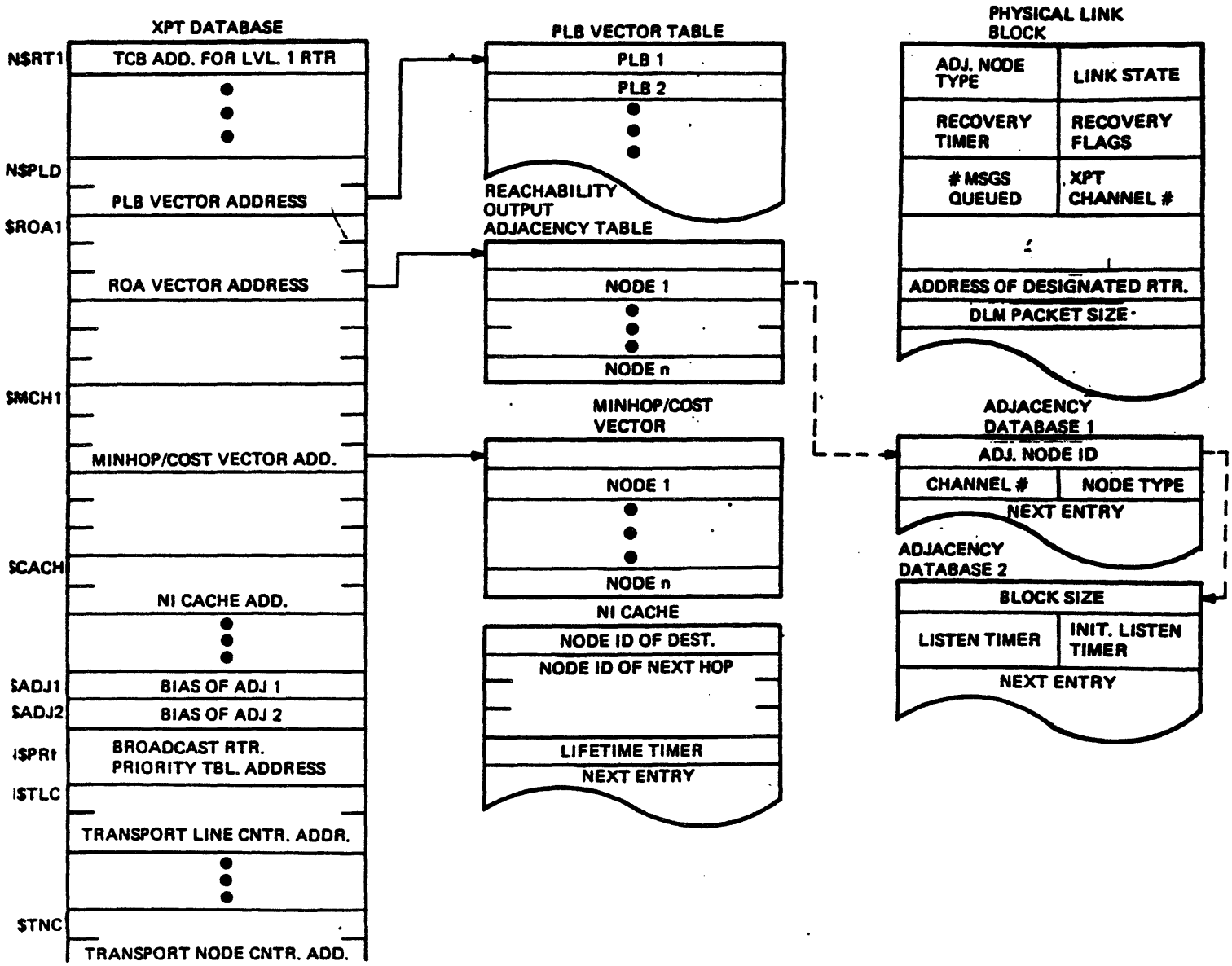


Figure 11 Routing Database

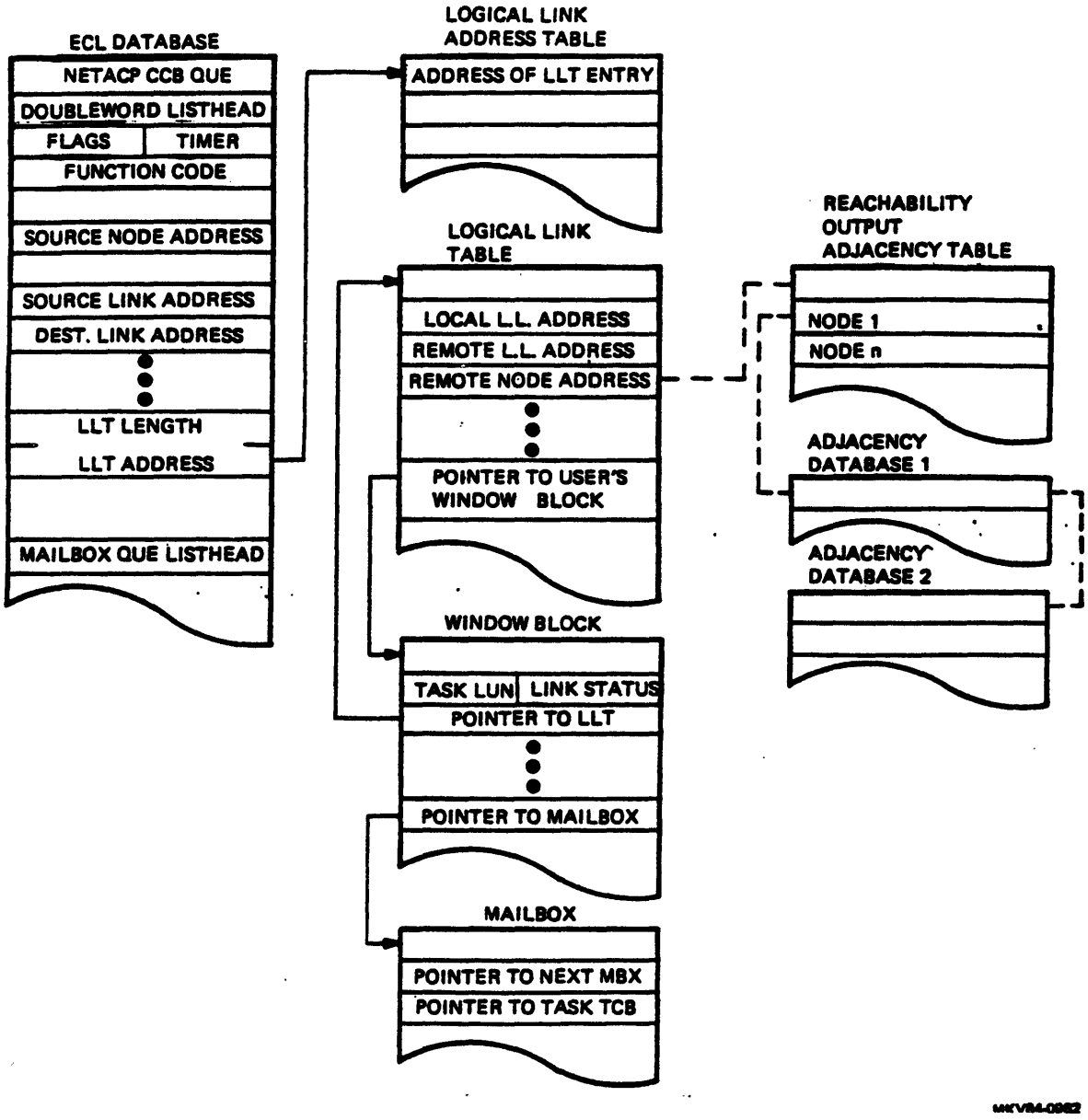


Figure 12 ECL Database (Server Base -- not specific to the Router)

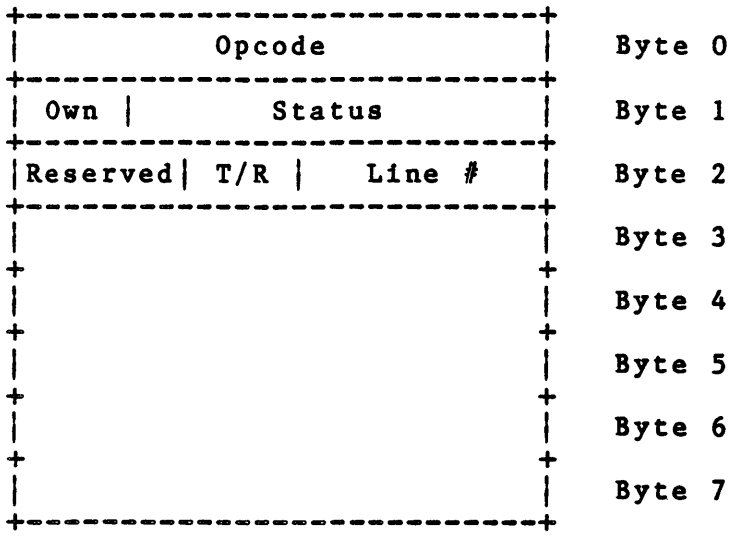
APPENDIX A

PAM SEGMENT DESCRIPTORS

- Command Segment Descriptor
- Status Segment Descriptor
- Receive Segment Descriptor
- Transmit Segment Descriptor
- Free Segment Descriptor

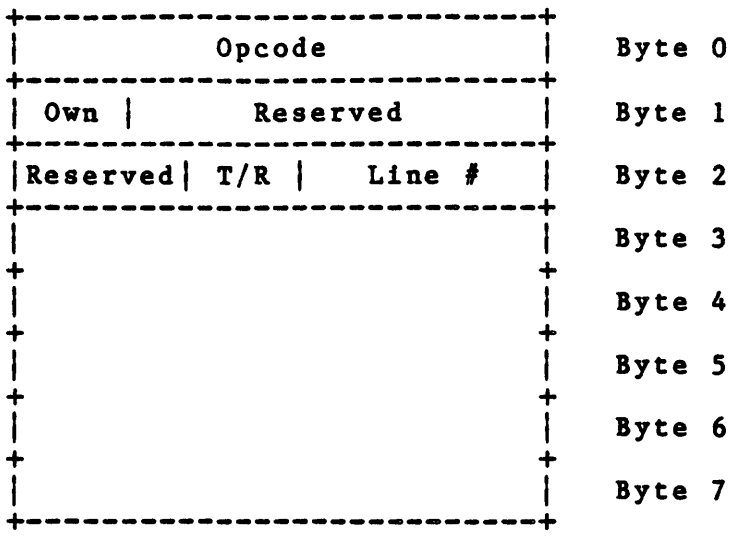
PAM SEGMENT DESCRIPTORS

Command Segment Descriptor



- Byte 0 - Type of command
- Byte 1 - Indicates ownership of the command descriptor; 1 = PAM ownership and 0 = CS ownership (bit 7)
  - Contains the status returned by the PAM upon completion of a command (bit 0:6)
- Byte 2 - Reserved (bit 7:5)
  - Transmit or receive indicator (bit 4). Transmit = 0, Receive = 1
  - Indicates the particular line for the operation (bit 3:0)
- Byte 3:7 - Used for command specific information (refer to Protocl Assist Module Functional Specification for more details on specific command information)

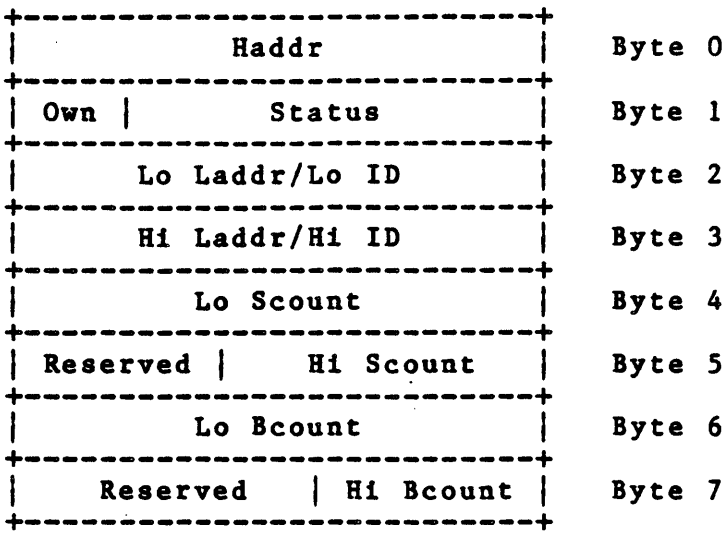
Status Segment Descriptor



- Byte 0 - Indicates the type of operation
- Byte 1 - Indcates ownership of the status descriptor
  - Indicates the reason that PAM was trying to obtain a free segment or the reason for halting transmission
- Byte 2 - Line number for the operation
- Byte 3:7 - Status dependent information

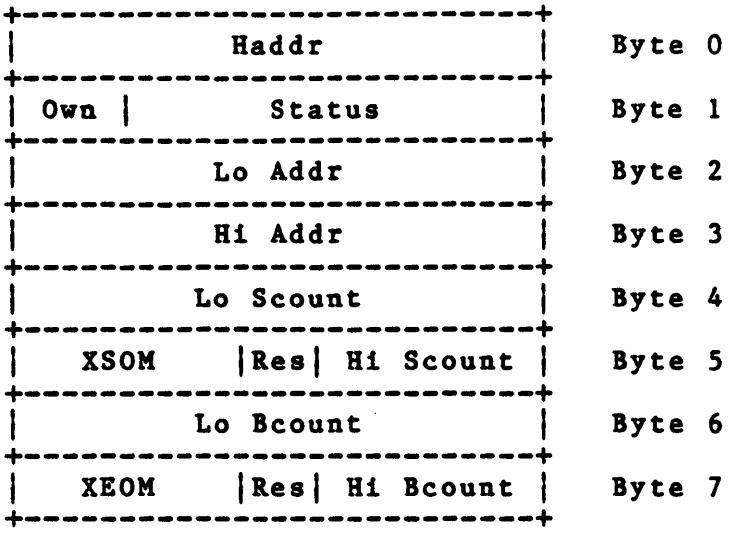
PAM SEGMENT DESCRIPTORS

Receive Segment Descriptor



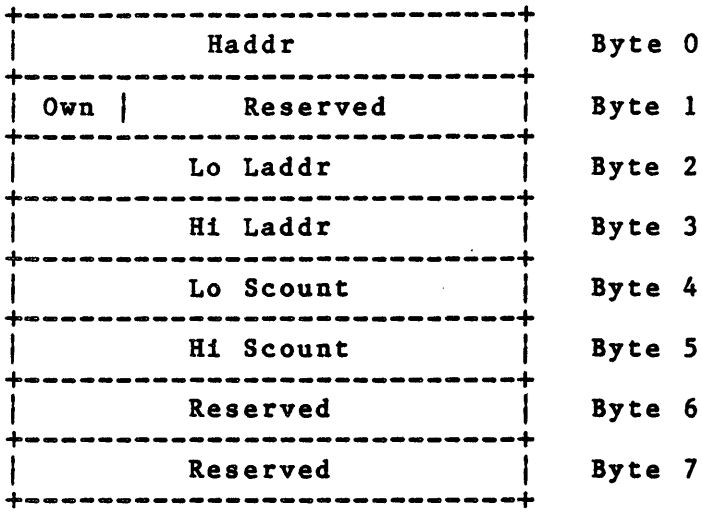
- Byte 0 - High order 6 address bits of the pointer to the receive segment
- Byte 1 - Indicates ownership of the segment (PAM or the CS) (bit 7)
  - Completion status of the segment (bit 6:0)
- Byte 2 - Low order 8 bits of the pointer to the receive segment
- Byte 3 - Middle 8 bits of the pointer to the receive segment
- Byte 4 - Segment byte count. The low order 8 bits of an 11 bit number indicating the length in bytes of the segment pointed to in the descriptor
- Byte 5 - Reserved (bit 7:3)
  - The high 3 bits of the byte count (bit 0:2)
- Byte 6 - Byte count. Low order 8 bits of the length in bytes of the received data in this segment
- Byte 7 - Reserved (bit 7:3)
  - The high order 3 bits of the length in bytes of the received data (bit 2:0)

Transmit Segment Descriptor



- Byte 0 - High order 6 address bits of the pointer to the transmit segment
- Byte 1 - Ownership
  - The completion status of the segment
- Byte 2 - Low order 8 address bits of a pointer to the transmit segment
- Byte 3 - Middle 8 address bits of a pointer to the transmit segment
- Byte 4 - Segment byte count
- Byte 5 - Transmit start of message command. Indicates the action that the PAM takes when beginning to transmit this segment (bit 7:4)
  - Reserved (bit 3)
  - Segment byte count
- Byte 6 - Byte count. Low order 8 bits of the length in bytes of the transmitted data in this segment
- Byte 7 - Transmit end of message command. This field indicates the action that the PAM takes when finished with the transmission of this message
  - Byte count. High order three bits of the length of the transmitted data in this segment

Free Segment Descriptor Block



- Byte 0 - High order 6 address bits of the pointer to the free segment
- Byte 1 - Indicates owner of the descriptor (bit 7)
  - Reserved (bit 6:0)
- Byte 2 - Low order 8 address bits of the pointer to the free segment
- Byte 3 - Middle 8 address bits of the pointer to the free segment
- Byte 4 - Segment byte count. Low order 8 bits of the length in bytes of the segment pointed to in the descriptor
- Byte 5 - Segment byte count. High order 3 bits of the length in bytes of the segment pointed to in the descriptor (bit 2:0)
- Byte 6:7 - Reserved

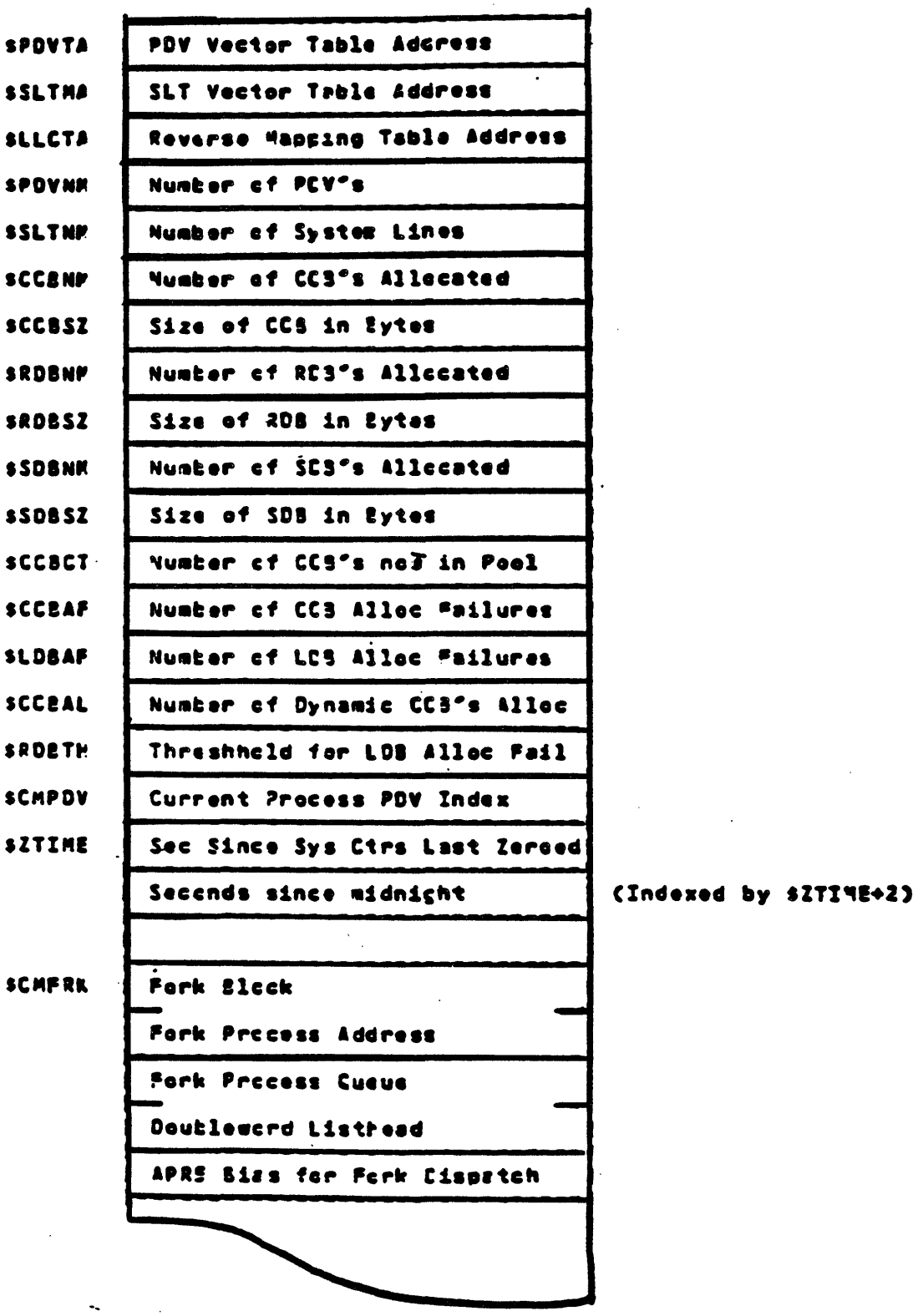
APPENDIX B

DECNET DATA STRUCTURES

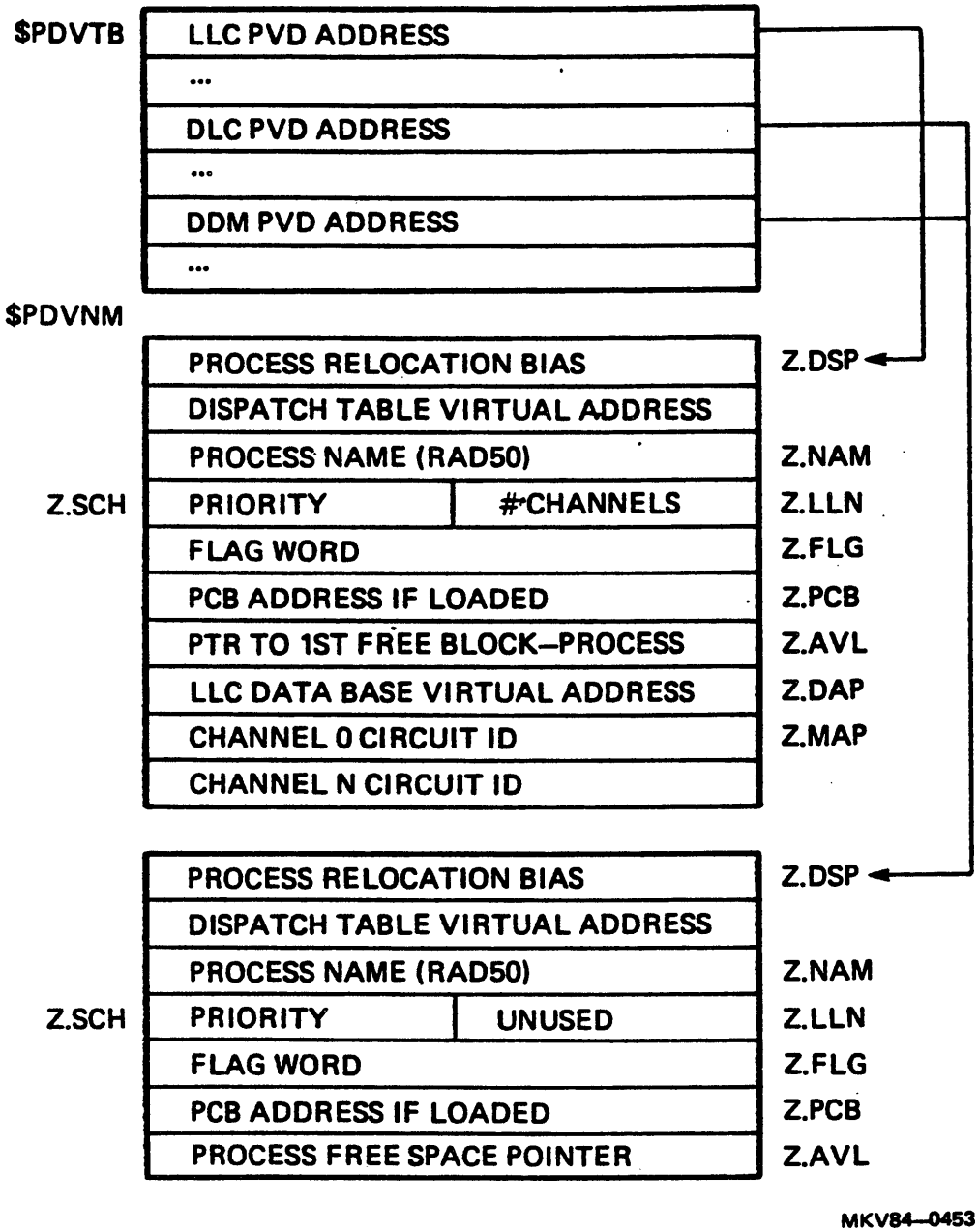
- CEX Common Database (CEXCM)
- Process Descriptor Vector
- System Line Table
- DDM Line Table
- Physical Link Descriptor
- XPT Database
- Adjacency Database 1 + 2
- Minhop/Mincost Vector
- Reachability and Output Adjacency Table
- Reverse Mapping Table
- Communication Control Block (CCB)
- ECL Database
- Mailbox
- Window Block
- Logical Link Table

DECNET DATA STRUCTURES

DECnet Structure: Communications Executive Common Database (CEXCM)
Total Number: 1
Allocated from: CEX
Use: Contains pointers to pertinent data structures
for the CEX and network components.

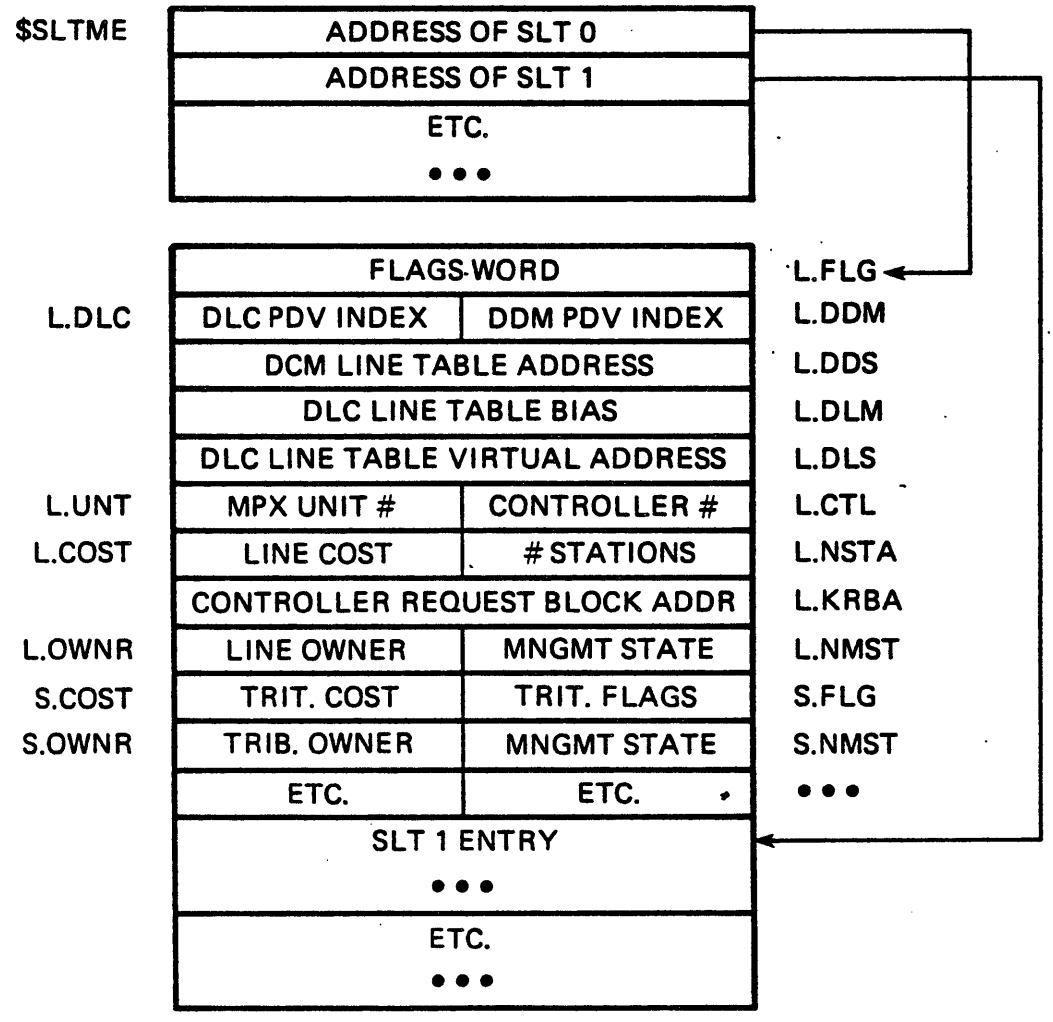


DECnet Structure: Process Descriptor Vector (PDV)
Size (bytes): 16. plus 2. per channel
Total Number: 1 per process
Allocated from: CEX
Pointed to by: CEXCM - $PDVTA
Use: Communication process description



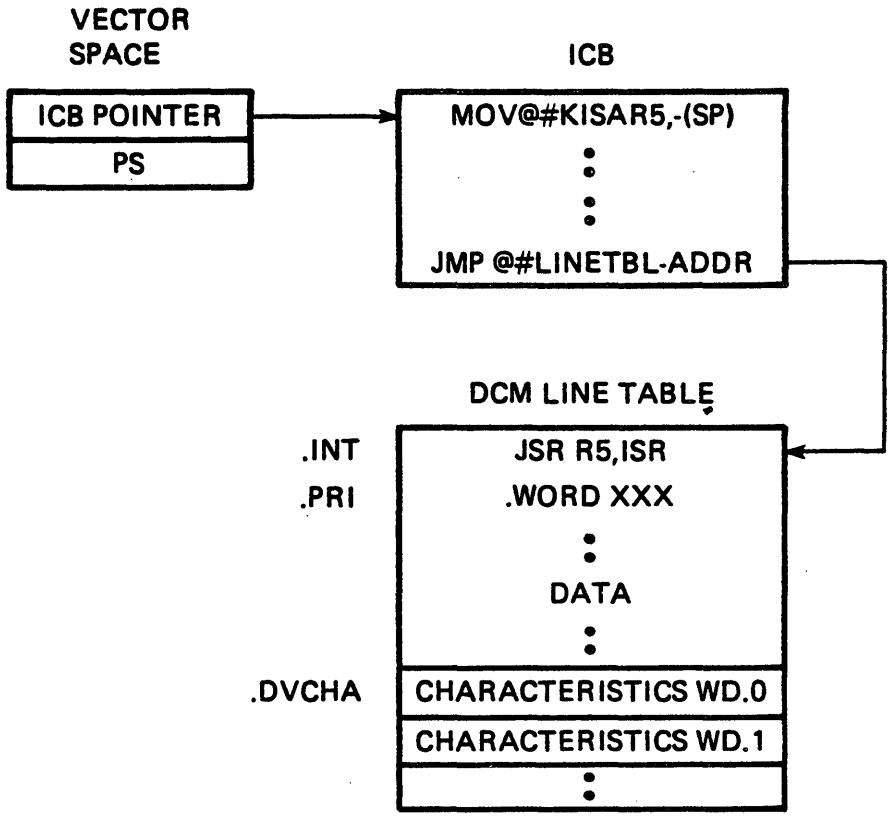
MKV84-0453

DECnet Structure: System Line Table (SLT)
Size (bytes): 18. plus 4 per tributary
Total Number: 1 entry per physical line
Allocated from: CEX
Pointed to by: CEXCM - $SLTMA
Use: Interprocess interface for the communication channel

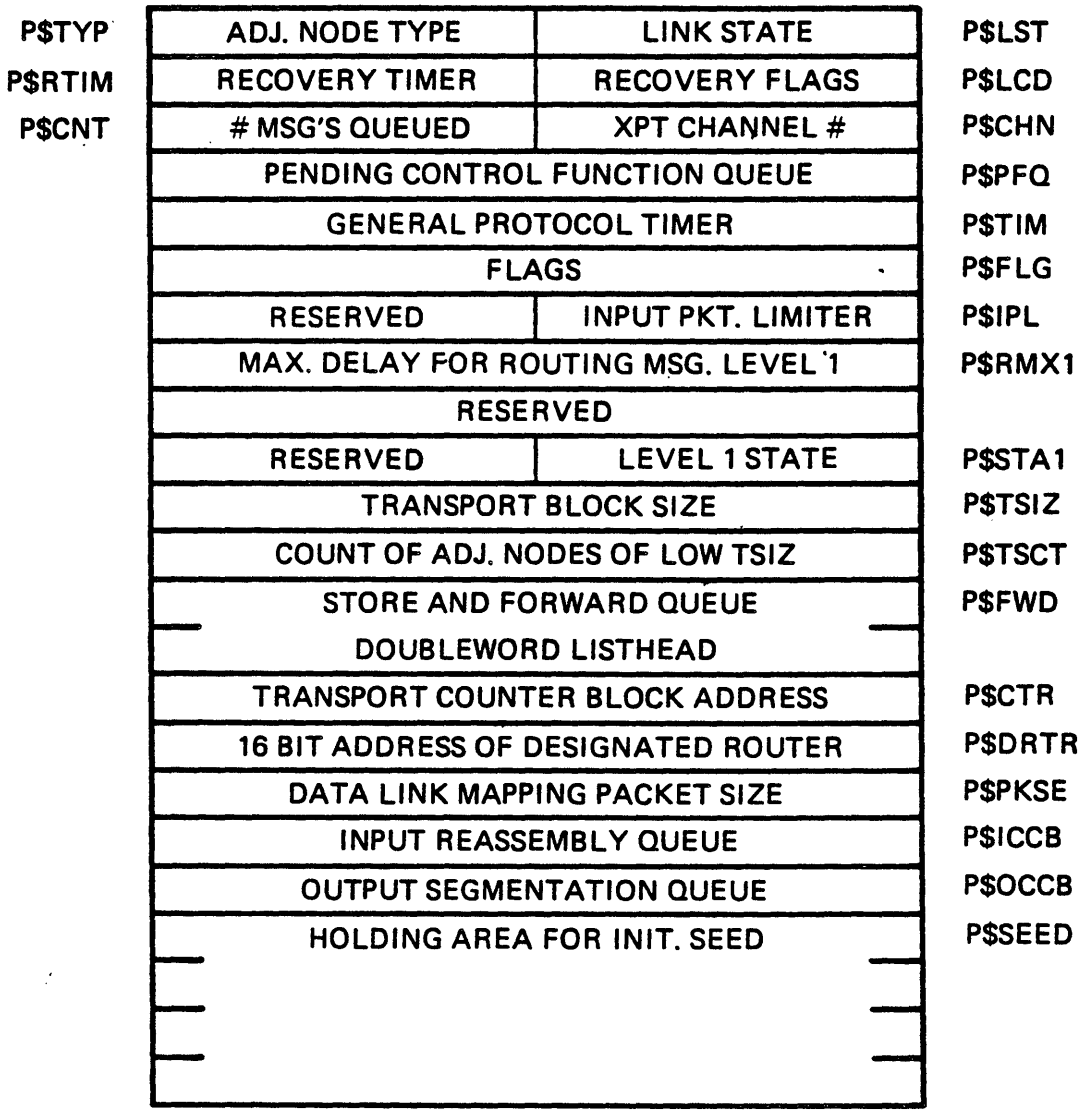


MKV84-0164

DECnet Structure: DDM Line Table
Size (bytes): Varies
Total Number:
Allocated from: Process space
Pointed to by: SLT - L.DDS
Use: Communication Device specific data, interrupt
linkage code

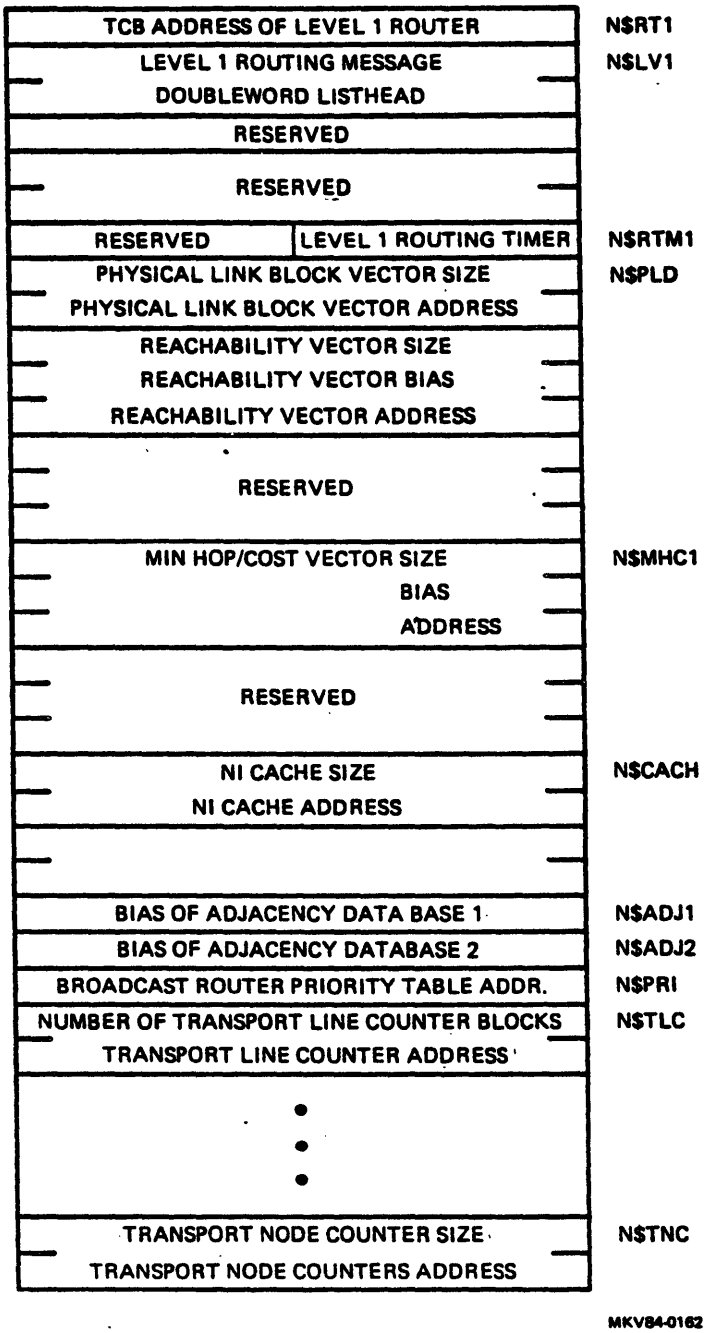


DECnet Structure: Physical Link Descriptor (PLD)
Size (bytes): 28.
Total Number: 1 per physical line
Allocated from: DSR
Pointed to by: XPT Database (N$PLD)
Use: used by XPT to control/monitor each physical connection to an adjacent node

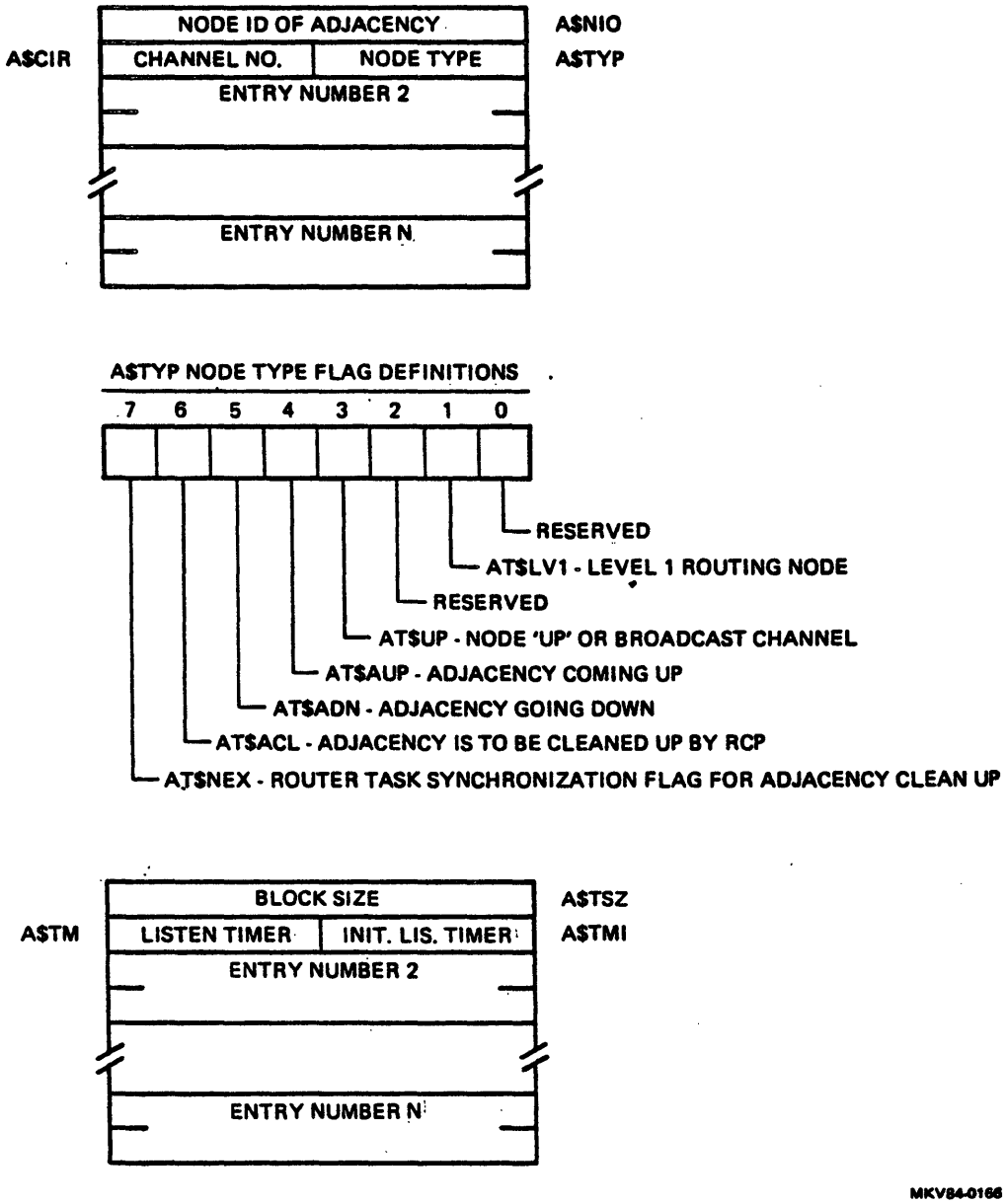


MKV84-0163

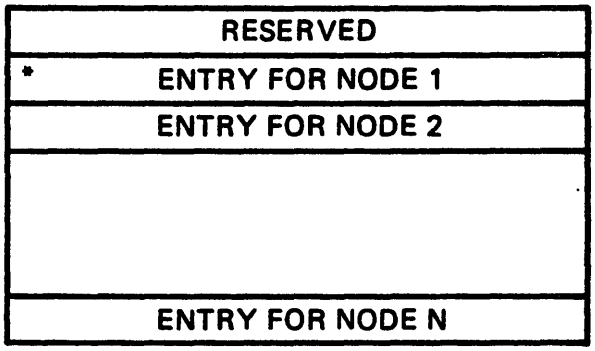
DECnet Structure: XPT Database
Size (bytes): 70.
Total Number: 1
Allocated from: DSR
Pointed to by: XPT PDV - Z.DAT
Use: Contains pointers to data shared by the XPT process and the routing task (RCP).

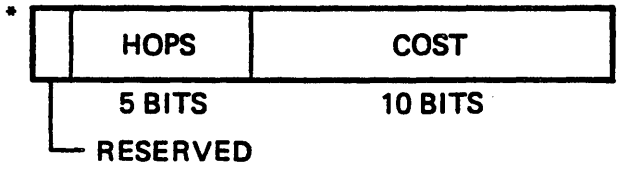


DECnet Structure: Adjacency Database 1 + 2
Size (bytes): 4. per entry in each database
Total Number: 1 entry in each database per adjacent node
Allocated from: POOL..
Pointed to by: XPT Database - N$ADJ1, N$ADJ2
Use: Contains information about each adjacent neighbor



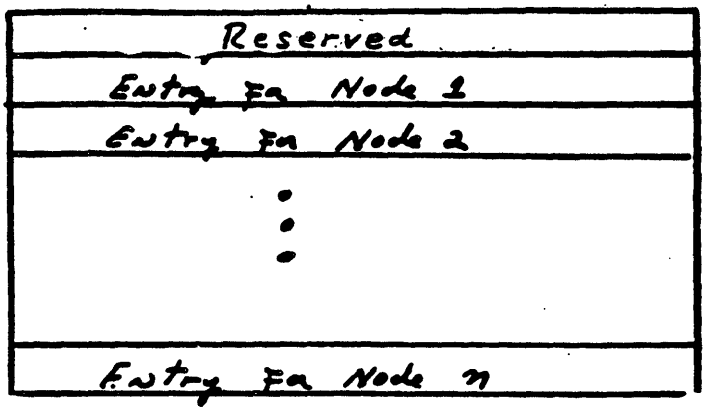
DECnet Structure: Minhop/Mincost Table
Size (bytes): 2. per entry
Total Number: 1.
Allocated from: POOL..
Pointed to by: XPT Database - N$MHC1
Use: Each entry contains the minimum hops and cost along a path to a target node.



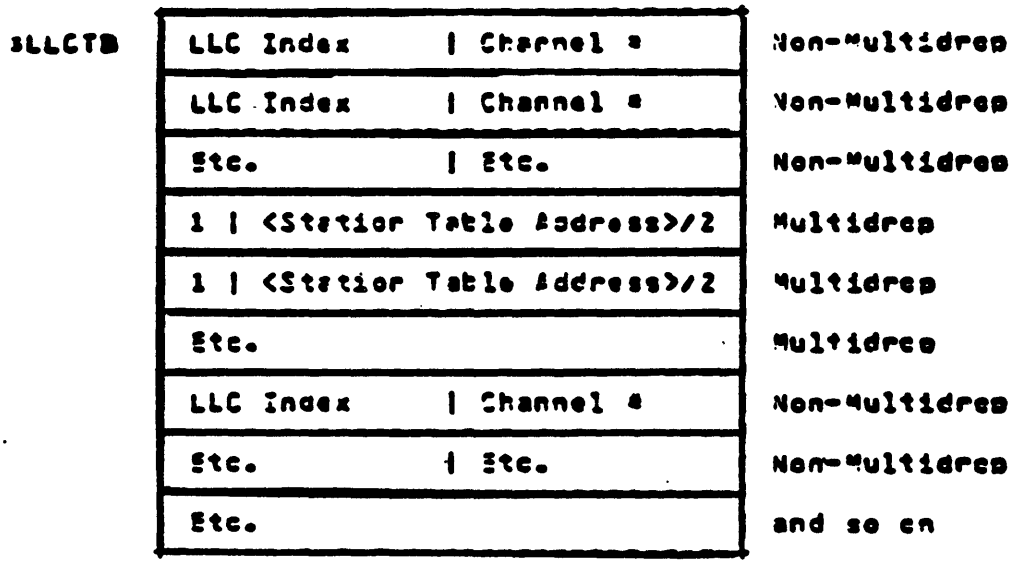


MKV84-0167

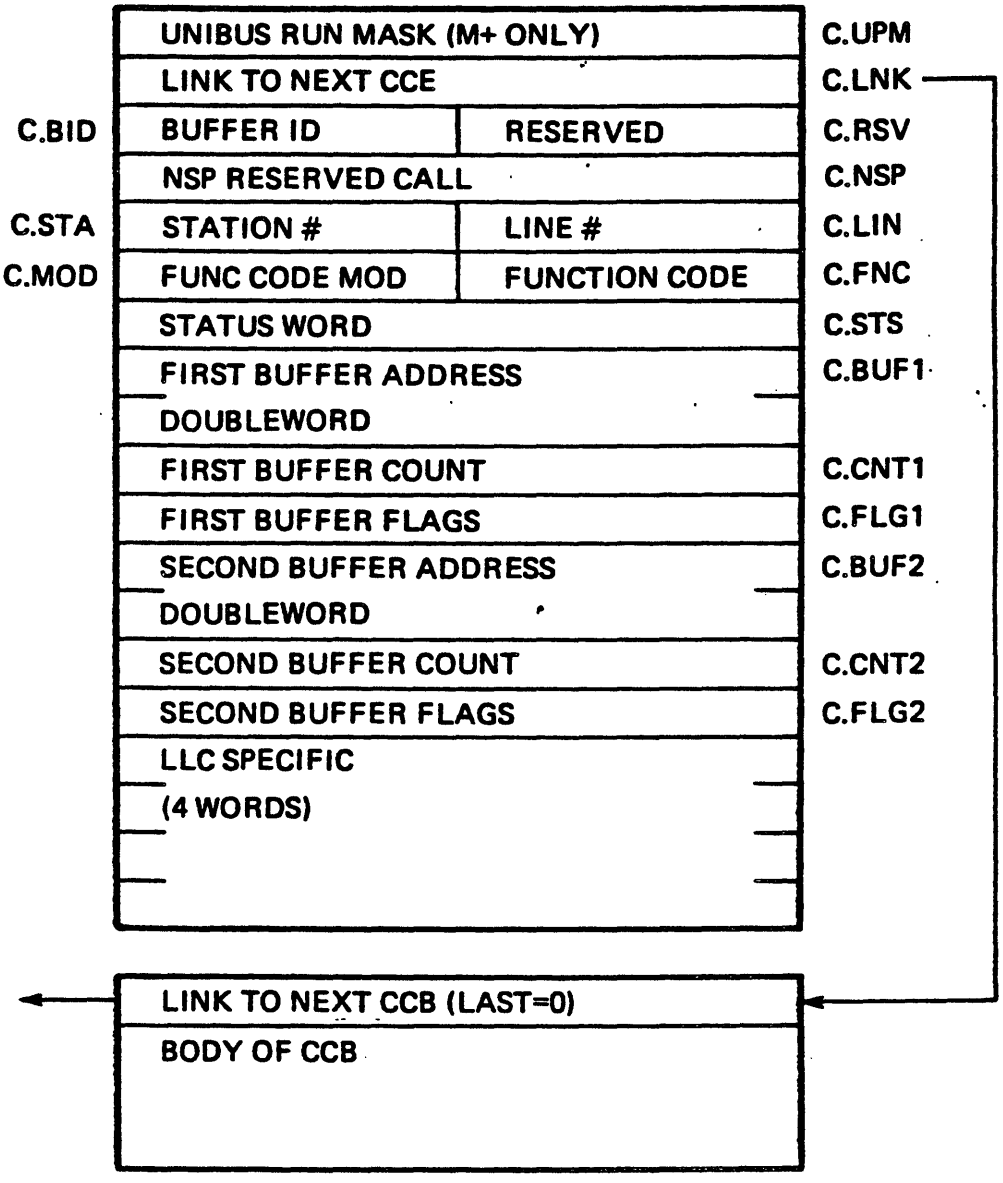
DECnet Structure: Reachability and Output Adjacency Table
Size (bytes): 2. per entry
Total Number: 1. entry for each node in the network
Allocated from: Process space
Pointed to by: XPT Database - N$ROA1
Use: Each entry contains an adjacency address if the target node is reachable and a 0 if not reachable



DECnet Structure: Reverse Mapping Table
Size (bytes): 2. per entry
Total Number: 1. entry for each XPT channel
Allocated from: DSR
Pointed to by: Indexed by SLN
Use: Maps from an LLC channel to a specific physical line

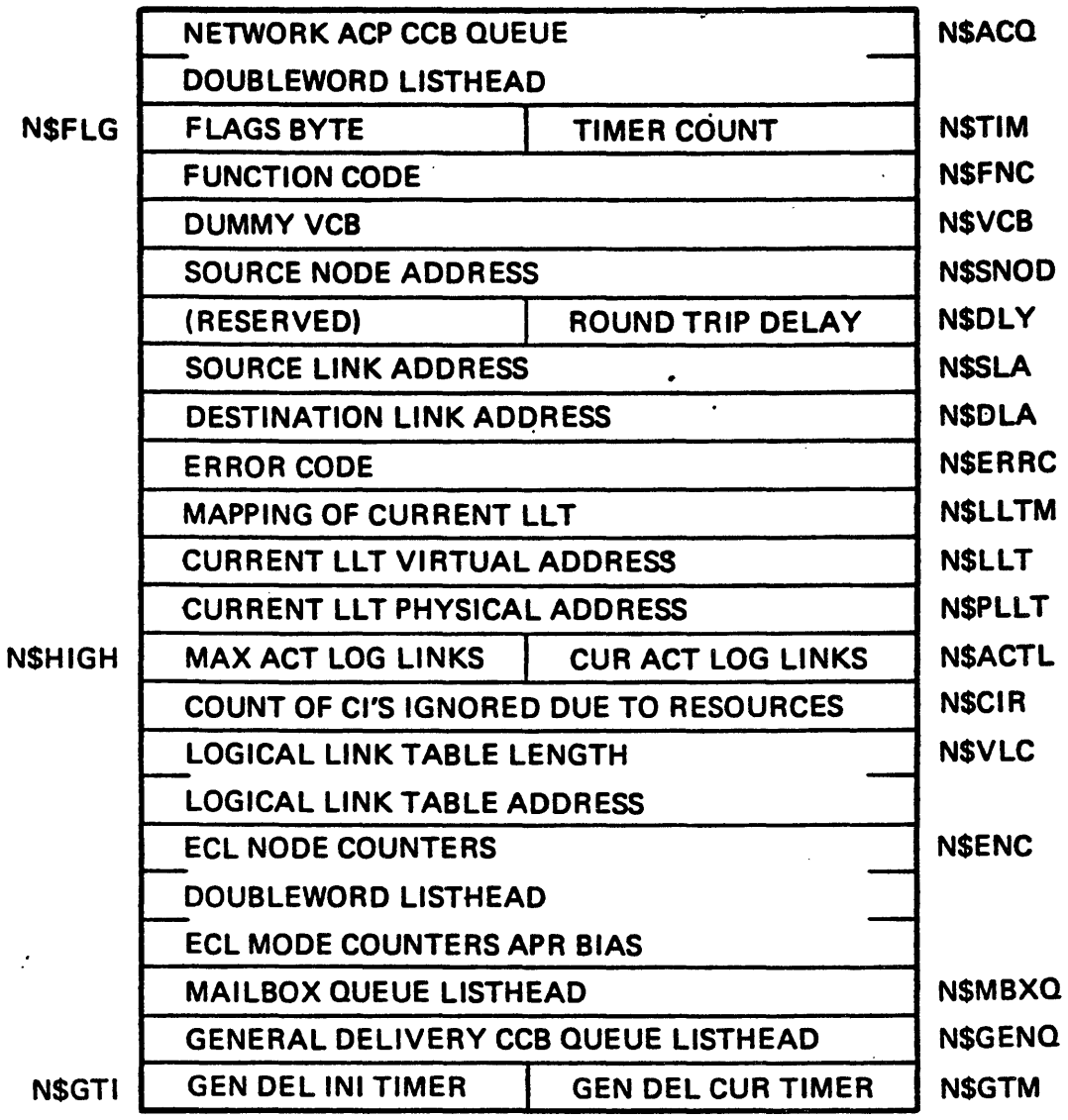


DECnet Structure: Communication Control Block (CCB)
Size (bytes): 38.
Total Number: Allocated dynamically
Allocated from: DSR
Pointed to by: CEXCM - $CCBLH
Use: Interprocess communication



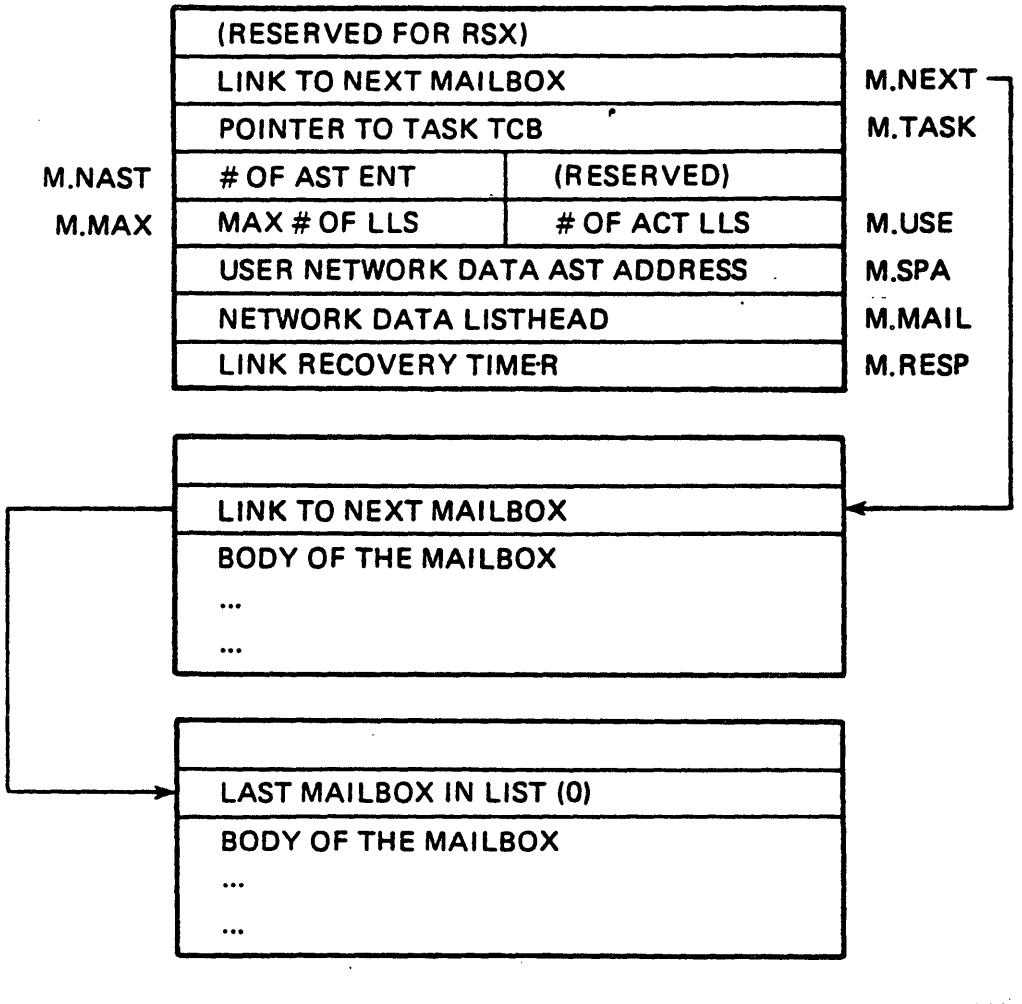
MKV84-0467

DECnet Structure: ECL Database
Size (bytes): 46.
Total Number: 1.
Allocated from: DSR
Pointed to by: ECL PDV - Z.DAT
Use: Contains data shared between the ECL process and NETACP



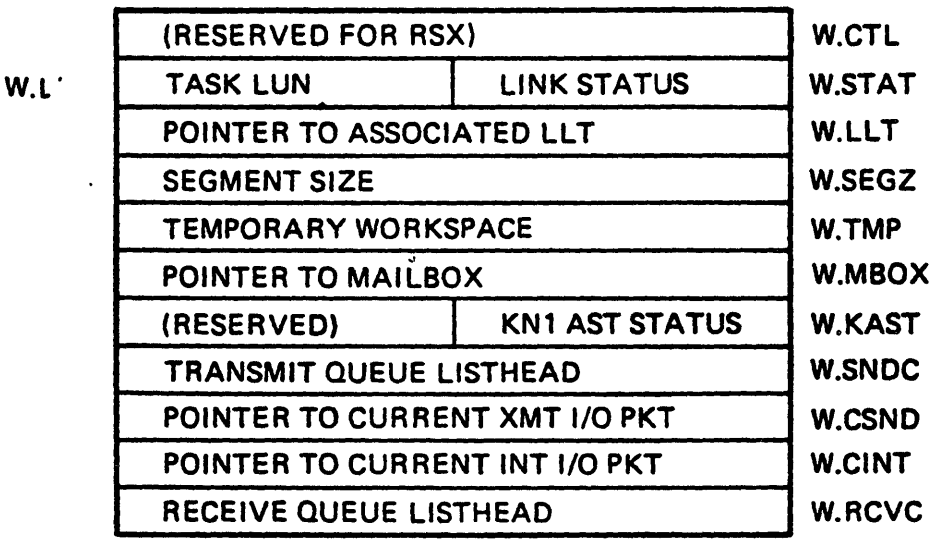
MKV84-0458

DECnet Structure: Mailbox
Size (bytes): 16.
Total Number: 1. per active network task
Allocated from: DSR
Pointed to by: Window Block - W.MBOX, Task Header - H.LUT
Use: Controls the tasks' overall network access



MKV84-0281

DECnet Structure: Window Block
Size (bytes): 40.
Total Number: 1. per logical link
Allocated from: DSR
Pointed to by: User Link Address table
Use: Data transmission and reception over established logical link.



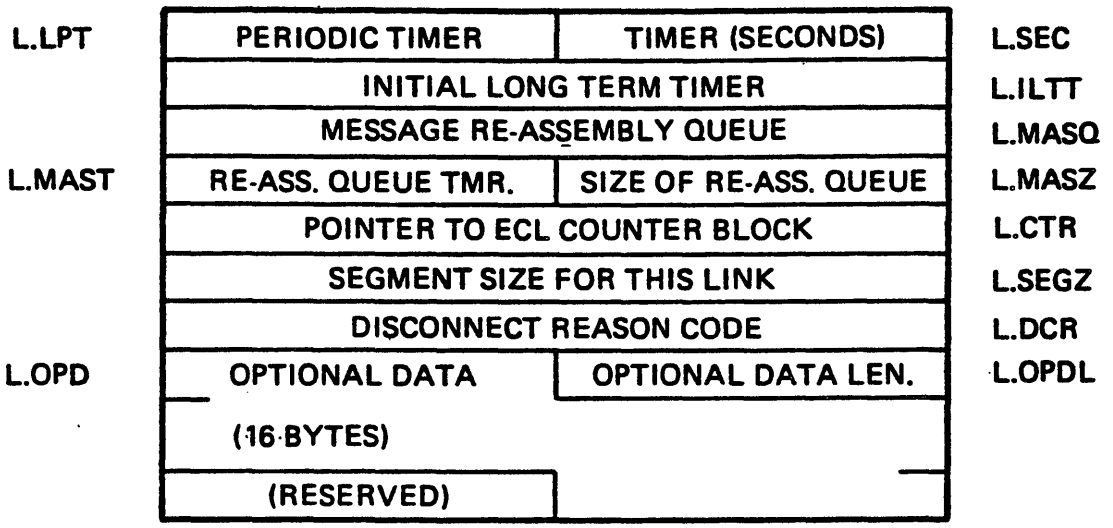
MKV84-0283

DECnet Structure: Logical Link Table (LLT)
Size (bytes): 42. - 62.
Total Number: 1. per logical link
Allocated from: DSR
Pointed to by: Logical Link Address Table
Use: Contains the source and destination logical link address

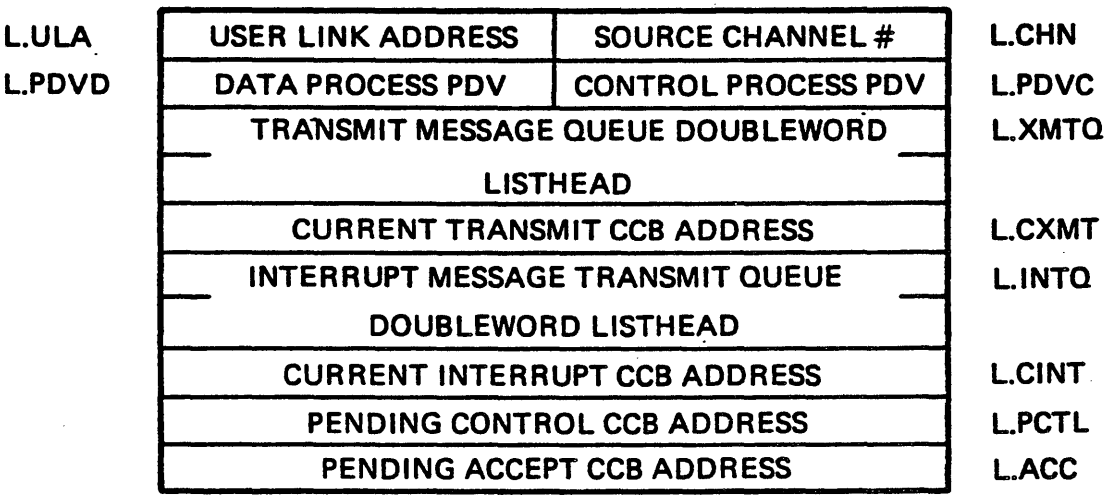
L.TYP	LINK TYPE	LINK STATE	L.STA
	LOCAL LOGICAL LINK ADDRESS		L.LLA
	REMOTE LOGICAL LINK ADDRESS		L.RLA
	REMOTE NODE 16 BIT ADDRESS		L.REM
L.TIPD	# DATA XMTS IN PROG.	# I/LS XMTS IN PROG.	L.TIPI
L.VER	REMOTE NSP VERSION	LINK FLAGS	L.FLAG
	NEXT DATA SEGMENT NUMBER TO BE ASSIGNED		L.NXN
	NEXT I/LS SEGMENT NUMBER TO BE ASSIGNED		L.NIN
	NEXT DATA SEGMENT NUMBER TO BE RECEIVED		L.RNO
	HIGHEST ACK # FROM USER ON DATA CHAN		L.USA
	NEXT I/LS SEGMENT NUMBER TO BE RECEIVED		L.LNO
	HIGHEST I/LS ACK # FROM USER		L.LSA
	LAST DATA SEGMENT NUMBER ACK'D		L.LDA
	LAST INT/LS SEGMENT NUMBER ACK'D		L.LIA
L.CSTA	NET DISC SUBSTATE	USER DISC SUBSTATE	L.USTA
	POINTER TO USER'S WINDOW BLOCK		L.WIND
L.TIC	INTERRUPT COUNT	TRANSMIT COUNT	L.TC
	FLOW CONTROL REQUEST COUNT (I/LS)		L.LSFI
	FLOW CONTROL REQUEST COUNT (DATA)		L.LSFD
	REMOTE FLOW CONTROL COUNT ESTIMATE		L.RCF
	I/LS PENDING ACK QUEUE		L.ILSQ
L.RTYD	RETRY CELL (DATA)	TIMER CELL (DATA)	L.TMRD
L.RTYI	RETRY CELL (I/LS)	TIMER CELL (I/LS)	L.TMRI
	MESSAGE AWAITING RETRANSMISSION		L.RTQ
	LONG TERM TIMER		L.LIT

MKV84-0460

DECnet Structure: Logical Link Table (continued)



THE FOLLOWING ADDITIONAL FIELDS ARE ONLY INCLUDED IF SYSTEM LEVEL INTERFACE (SLI) IS PRESENT.



MKV84-0456

CONTENTS

FIGURES

TABLES

NODE TROUBLESHOOTING

Node Troubleshooting

INTRODUCTION

A number of troubleshooting tools are available for dealing with Communications Server problems. This chapter describes the tools that may be used to verify that the server is working properly and to analyze any problems that occur. The available tools include:

- Diagnostic Tools
  - Server Self Test
  - Loadable Diagnostic Image (LDI)

- Network Management Facilities
  - Loopback tests
  - Examining Error Counters
  - Network Event Logging
  - Remote Console Facility

- On-line Debugging Tool (POD)

- Crash Dump Analysis

Some problems require more sophisticated procedures such as analyzing crash dumps or tracing through listings to retrieve specific information which may not be documented.

REFERENCES

RSX DECnet System Manager's Guide

DECnet VAX System Manager's Guide

Ethernet Communications Server Operations and Maintenance Manual

DECnet Router Installation and Operation Guide

DIAGNOSTIC TOOLS

To check the server hardware components, run the following diagnostic tests:

- Self Tests
- Loadable Diagnostic Image (LDI)
- On-line Tests

These diagnostics are installed at the same time as the server software and reside

- On the host node where the server software was originally installed.
- On any host node to which the server software was copied after installation.

Self Tests

The server self tests are stored in ROM and are executed:

- At power up of the server
- By depressing the START button on the server
- Through NCP commands at the host, eg.

```
NCP>LOAD NODE ROUTER

        or

NCP>TRIGGER NODE ROUTER
```

Node Troubleshooting

To load and run the diagnostics, the user must be sure that:

- The server is connected to the Ethernet.
- A host node is running and connected to the Ethernet.

NOTE

If either of these conditions is not met, then the server will Halt with an _02 LED display and wait.

The self tests do a complete check of the path over which the software is to be transferred, from the host node to the server (load path) for the:

- CPU
- Memory
- CBT
- DEUNA
- Ethernet
- Cursory check of the PAM(s)

Self Test Versions

There are two versions of the self tests: long and short.

- Long Self Test:
  - TEST button is pressed in
  - Tests the entire 512KB of memory
  - CBT display blinks at a slower rate, indicating the long version of the testing procedure
  - Displays the 48 bit Ethernet address of the server
  - 4 minute runtime
  - On completion, requests the LDI

- o Short Self Test:
  - TEST button out
  - Tests only the first 32KB of memory
  - Cursory check of the rest of memory
  - On completion, requests the server software image

Errors found during any portion of the diagnostic tests are reported by means of fault lamps and digital displays located on the server. The fault indicators when lit show which of the components is at fault: a line card, logic board, or Ethernet cable. The digital display identifies the faulty module (FRU). Figure 1 shows where these indicators are located and Table 1 lists the indications a user will see if the server has an error.

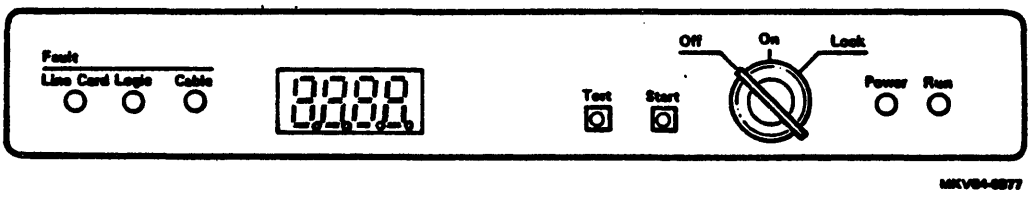


Figure 1 Communications Server Fault Indicators

Table 1 Testing and Diagnostics Error Indications

PANEL INDICATION TYPE	STATE	COMMENTS
Fault Light	ON	Line Card, Logic, or Cable Fault
Run Light	OFF	CPU Halted
Cyclic Pattern Display	OFF	Software not Working *
Line Card Light	ON	Line Card(s) Faulty (or not in use)**

NOTE

* This assumes that the server software was previously running.

** Line card lights are only valid when running the line card diagnostics.

Table 2 Self Test Indications

Indicator	Color	While Test is Running	After Test is Complete
Line Card Fault	Red	Blinks	Off
Logic Fault	Red	Blinks	*
Cable Fault	Red	Blinks	Off
Segment Display 1	Red	Blinking 8.	Underscore On
Segment Display 2	Red	Blinking 8.	Underscore On
Segment Display 3	Red	Blinking 8.	0
Segment Display 4	Red	Blinking 8.	2
Test	Red	Blinks	Off
Start	Red	On	On
Power	Green	On	On
Run	Green	On	Off
Line Card Light(s)	Red	On	On

NOTE

Blinking rates are 1 per second. Line card lights are located on each line card. The self test does not check any of the PAM line cards.

* This fault light will be on after completion of the self test and the server is not connected to the Ethernet. If the server is connected to the Ethernet, this light should be off at the completion of the self test.

Upon completion of the tests, a request of an "image" is made to the host. The image that is loaded will be either the Loadable Diagnostic Image (LDI), or the Router server software image depending on the TEST button setting.

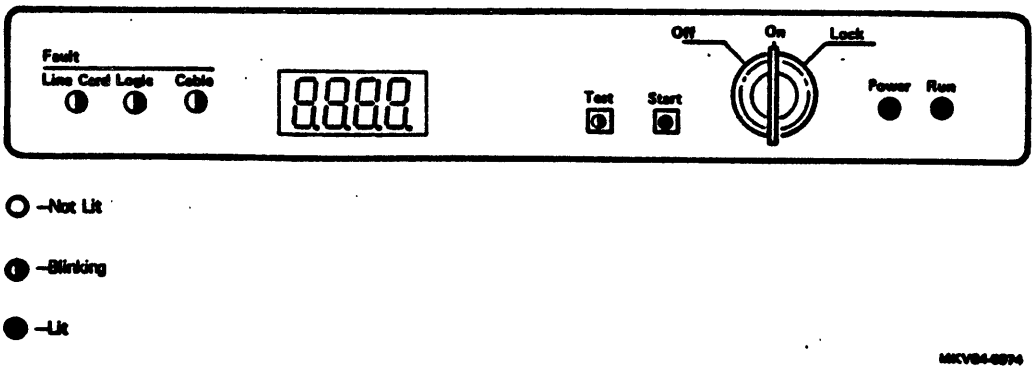


Figure 2 Front Panel Display During Self Test

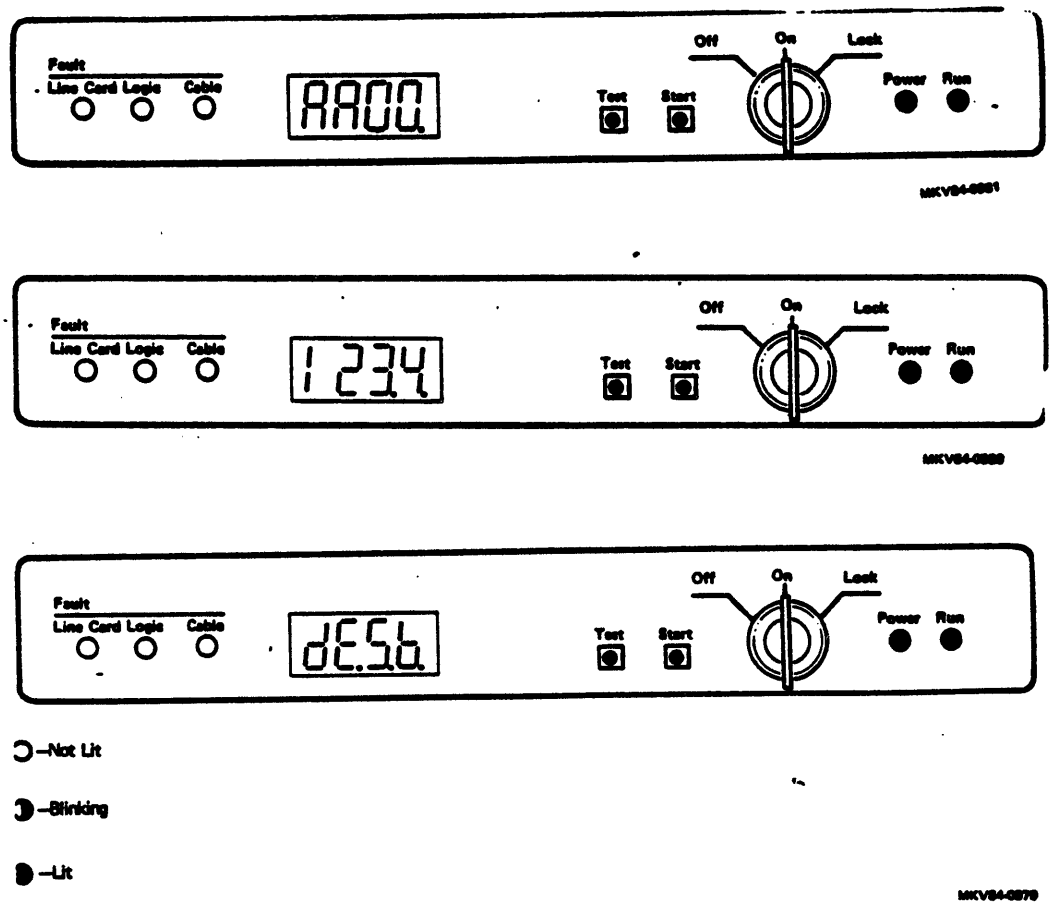
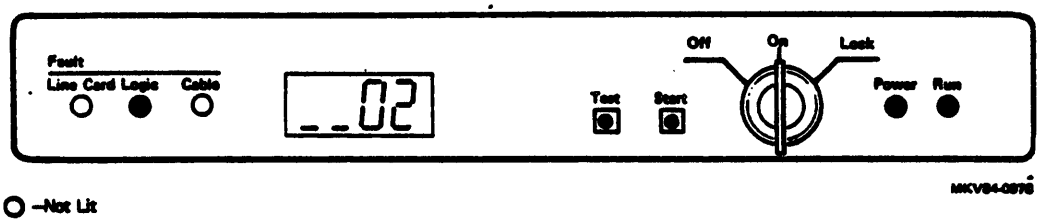


Figure 3 Ethernet Address Display (3 parts)



○ –Not Lit

◑ –Blinking

● –Lit

Figure 4 Example of a Logic Fault During Self Test

○ –Not Lit

◑ –Blinking

● –Lit

Figure 5 Example of a Cable Fault During Self test

Loadable Diagnostic Image (LDI)

The LDI is loaded and started if the "Test" button on the front panel is pressed to the in position. This image is down-line loaded over the Ethernet to the server. Results of LDI tests show on the LED display on the front panel of the server. The LDI runs dignostics on the PAM, Line cards, and CBT.

NOTE

The results of the LDI can also be displayed on a host console using the Console Carrier facility.

An integral part of the LDI is the SYSEXE (System Exerciser). This program, which automatically runs after the other diagnostics have completed, is designed to test the server in a manner close to actual use. Its main objective is to create a stressful situation by establishing communications between the server modules at very high data rates.

SYSEXE tests the following components:

- The PDP-11 UNIBUS to ensure proper data transfers
- PAM(s) for proper service to the line cards
- The line cards (up to 16) for proper data transfers
- The DEUNA is put into loopback mode and packets are checked for proper transfer from UNIBUS memory to and through the DEUNA

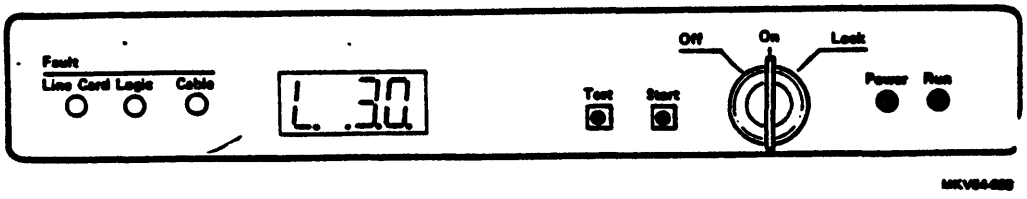


Figure 6 Front Panel Display During Loading of LDI

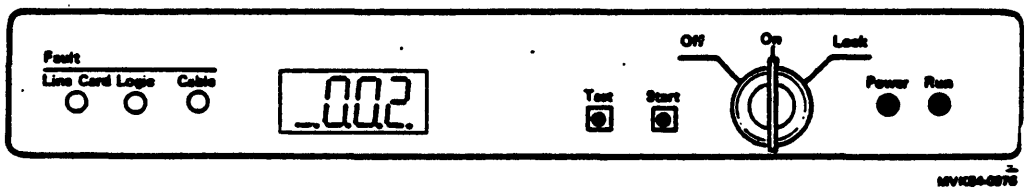


Figure 7 No Fault Display During LDI

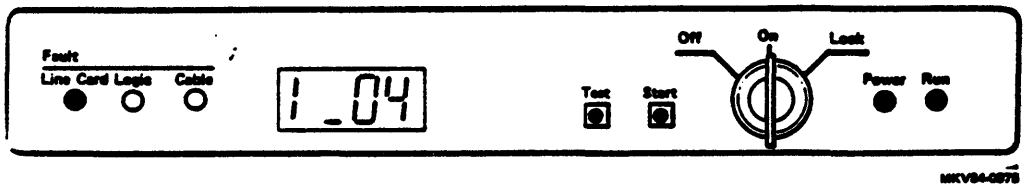


Figure 8 Line Card Fault Display During LDI

Network Management Facilities

The server provides standard NCP capabilities allowing a user to perform a variety of network management functions.

NETWORK CONTROL PROGRAM (NCP)

Table 3 NCP Troubleshooting Functions and Keywords

NCP Functions	Router Server	Router/X.25 Gateway
CONTROL FUNCTIONS /SET/TELL/LOOP		
Remote Command Execution	X	X
Setting and Clearing Executor Parameters	X	X
Setting and Clearing Passwords	X	X
Event Logging	X	X
Loopback	X	X
DISPLAYS /SHOW		
Line, Circuit, and Node Information	X	X
System Information	X	X
Logging Information	X	X

Table 4 NCP Troubleshooting Functions and Keywords (cont.)

NCP Functions	Router Server	Router/X.25 Gateway
Counters Information		
Line Counters	X	X
Circuit Counters	X	X
Node and Executor Counters	X	X
NODE NAME INFORMATION /SET/CLEAR/ZERO		
Changing a Node Name	X	X

LOOPBACK TESTING

NCP provides circuit and node level loopback tests that are used to determine:

- o The proper functioning of network components
- o The cause of a communications problem

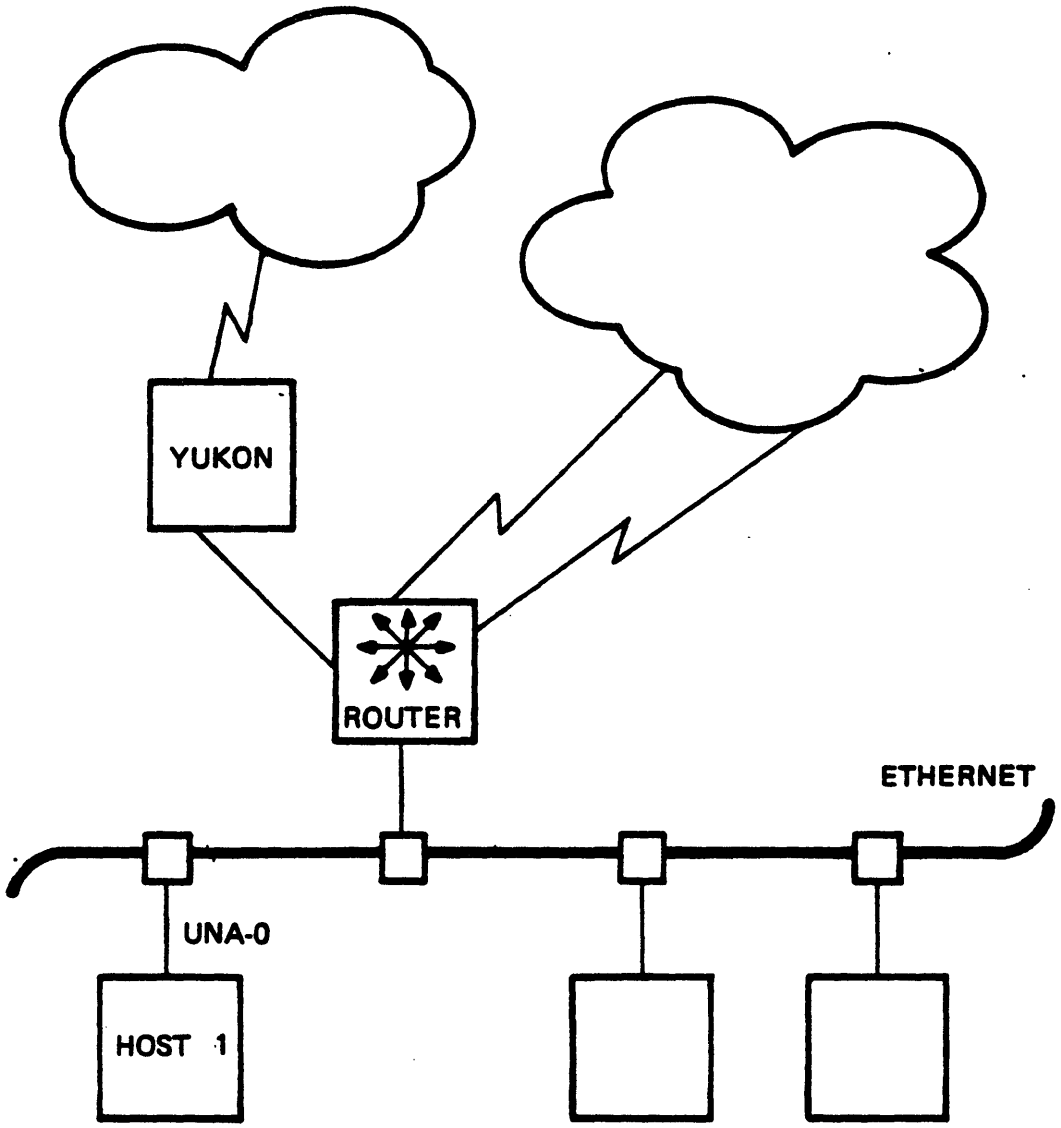


Figure 9 Circuit and Node Level Loopback Tests

Node, Circuit and Line Identification

- A DECnet node is identified by a node address, which is unique in the network, and by a name, which is local to the node (executor).

- A PSI node is identified by a 12-character DTE address (supplied by the PSN vendor), and by name, which is local to the node. The first four characters in the DTE address is an international code for a PSN (3110-TELENET, 2342-PSS, etc).

- An Ethernet node is identified by a unique Ethernet Physical Address. This address is set by the software at the Ethernet node. An Ethernet Physical Address is 48 bits in length and represented by six pairs of hexadecimal digits separated by hyphens.

  Example 1 AA-00-04-00-3B-04

- A circuit is the logical path between two adjacent nodes or DTEs. A DECnet circuit is identified by

  Device type-Controller number[-Unit number][.Tributary number]

  DMC-1

  DMP-1.3

  UNA-0

  LC-1

- A line is the physical path between two adjacent nodes. A line is identified by:

  Device type-Controller number[-Unit number]

  DUP-1

  SDP-1

  UNA-0

  LC-0

Node Level Loopback Tests

Node level loopbacks are used to verify the networking capabilities between the server and any other node. These nodes may be part of the LAN or remotely connected to the LAN via the Router server. A node level loopback is performed by using the TELL prefix with the NCP LOOP NODE command or by using NCP to SET EXECUTOR to the Router and performing the loopback tests.

Example 2 NCP>TELL SERVER LOOP NODE MAUDE

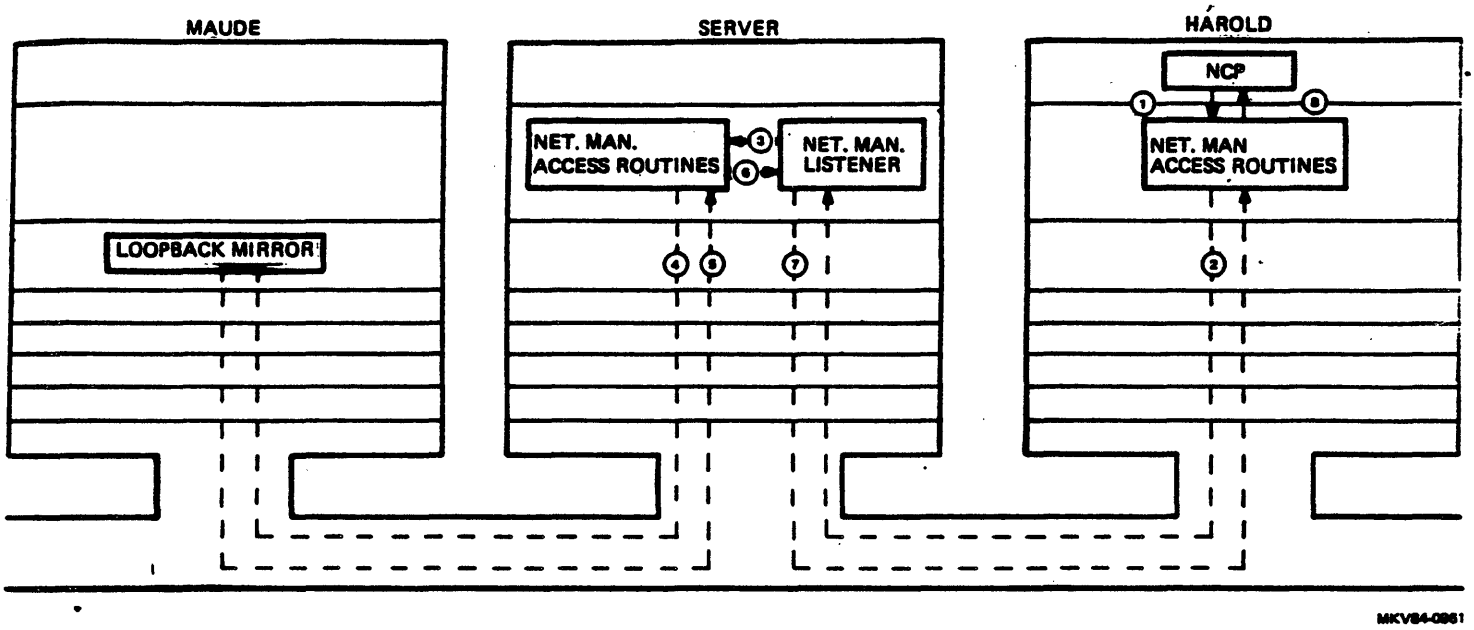


Figure 10 Node Level Loopback
(NCP>TELL SERVER LOOP NODE MAUDE)

Circuit Level Loopback Tests

Circuit level tests are used to verify either the Ethernet circuit or the point-to-point connections between the line cards on the Server and a remote node.

Point-to-Point Circuits:

These tests verify the point-to-point connection between the Router Server and a non-LAN node

Example 3 NCP>LOOP Circuit LC-1

Ethernet Circuits:

These tests verify:

- o the Ethernet cable
- o the interfaces, transceivers, and transceiver cables that connect the Server to the Ethernet

Example 4 NCP>LOOP CIRCUIT UNA-0

NOTE

The PHYSICAL ADDRESS needs to be specified on VMS hosts.

A user may specify a single Ethernet node or use the multicast address. When the multicast address is used, the first available Ethernet host will accept the data and loop it back.

Example 5 NCP>LOOP CIRCUIT UNA-0

A loop assist test may also be performed on the Ethernet circuits. These tests loop data between two Ethernet nodes, the initiating and destination nodes, with the help of a third, assistant node. There are three types of loop assist tests:

- o Transmit
  - Data is sent from the initiating node to the assistant node.
  - The assistant relays the data to the destination node.
  - The data is sent directly back to the initiating node.
- o Receive
  - Data is sent from the initiating node to the destination node.
  - The data is relayed to the assistant node.
  - The assistant relays the data to the initiating node.
- o Full (default)
  - Data is sent from the initiating node to the assisting node.
  - The assisting node relays the data to the destination node.
  - The destination node sends the data back to the assistant node.
  - The data is sent from the assistant node to the initiating node.

Example 6 NCP>LOOP CIRCUIT UNA-0 NODE SERVER ASSISTANT NODE MAUDE HELP FULL

NOTE

These tests are useful when the node being tested is receiving a signal too weak to respond to. The user needs to know the network layout to be sure that the assistant node is between the two nodes being tested.

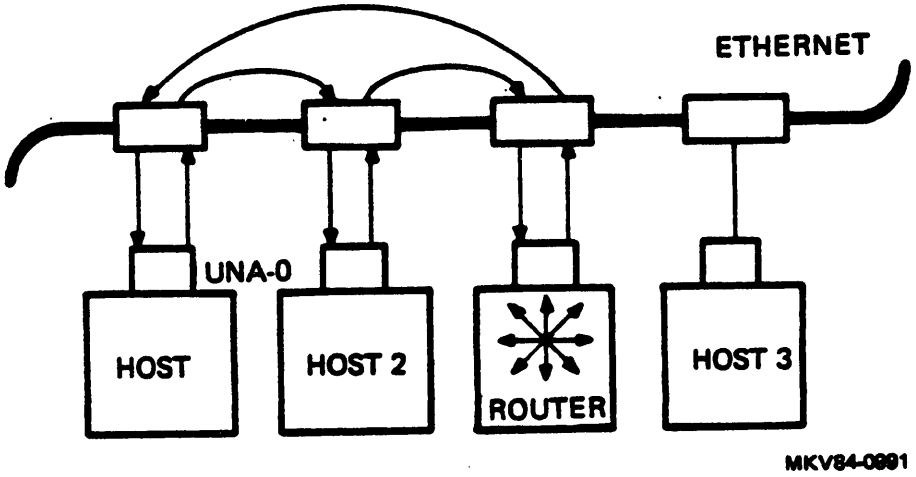


Figure 11 Loop Assist (Transmit)

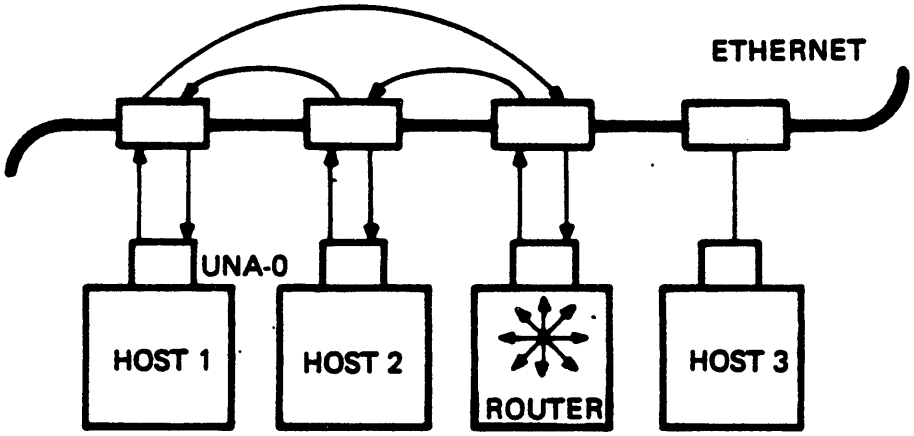


Figure 12 Loop Assist (Receive)

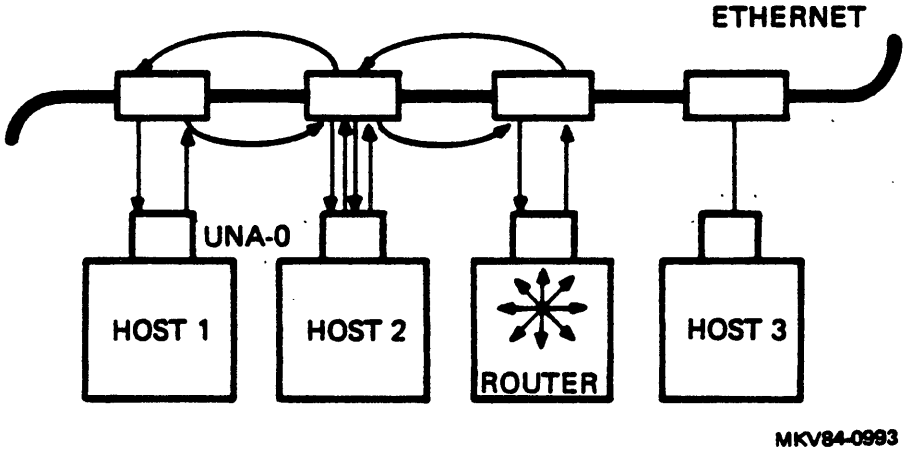


Figure 13 Loop Assist (Full)

EXAMINING NODE, CIRCUIT, AND LINE COUNTERS

DECnet software automatically maintains counters for the system, lines, circuits, and nodes. These counters are maintained by the current Executor and include information such as:

- The number of data packets sent and received over a line
- System buffer allocation failures
- Routing packet information

These counters can be displayed, zeroed and used along with loopback tests to try and locate the source of a network problem. The NCP SHOW commands display the various counters.

```
Node Counters as of  5-APR-1984 10:15:21

Executor node = 4.36 (OLEBOY)

          0  Aged packet loss
          0  Node unreachable packet loss
          0  Node out-of-range packet loss
          0  Oversized packet loss
          0  Packet format error
          0  Partial routing update loss
          0  Verification reject
          0  Control buffer failures
          0  Small buffer failures
          0  Large buffer failures
          0  Receive buffer failures
```

Example 7 Showing Executor (Router) Counters

```
Circuit Counters as of  5-APR-1984 10:16:16

Circuit = LC-8

        9404  Seconds since last zeroed
           0  Terminating packets received
           0  Originating packets sent
           0  Corruption loss
        7071  Transit packets received
       12926  Transit packets sent
           0  Transit congestion loss
           0  Circuit down
           0  Initialization failure
      243388  Bytes received
      511058  Bytes sent
        7730  Data blocks received
       13899  Data blocks sent
           0  Data errors inbound
           0  Data errors outbound
           0  Remote reply timeouts
           0  Local reply timeouts
           0  Remote buffer errors
           0  Local buffer errors
           0  Selection intervals elapsed
           0  Selection timeouts
```

Example 8 Showing Circuit Counter LC-8 on a DECnet Router

NETWORK EVENT LOGGING

The event logging facility records events that occur during network operation. The types of network events that are logged include:

- Changes in line, circuit, and node states
- Lost event reporting (too many events occuring simultaneously at the server)
- Passive loopback (the executor is looping back test messages - Router only)
- Errors in data transmission or in the line cards

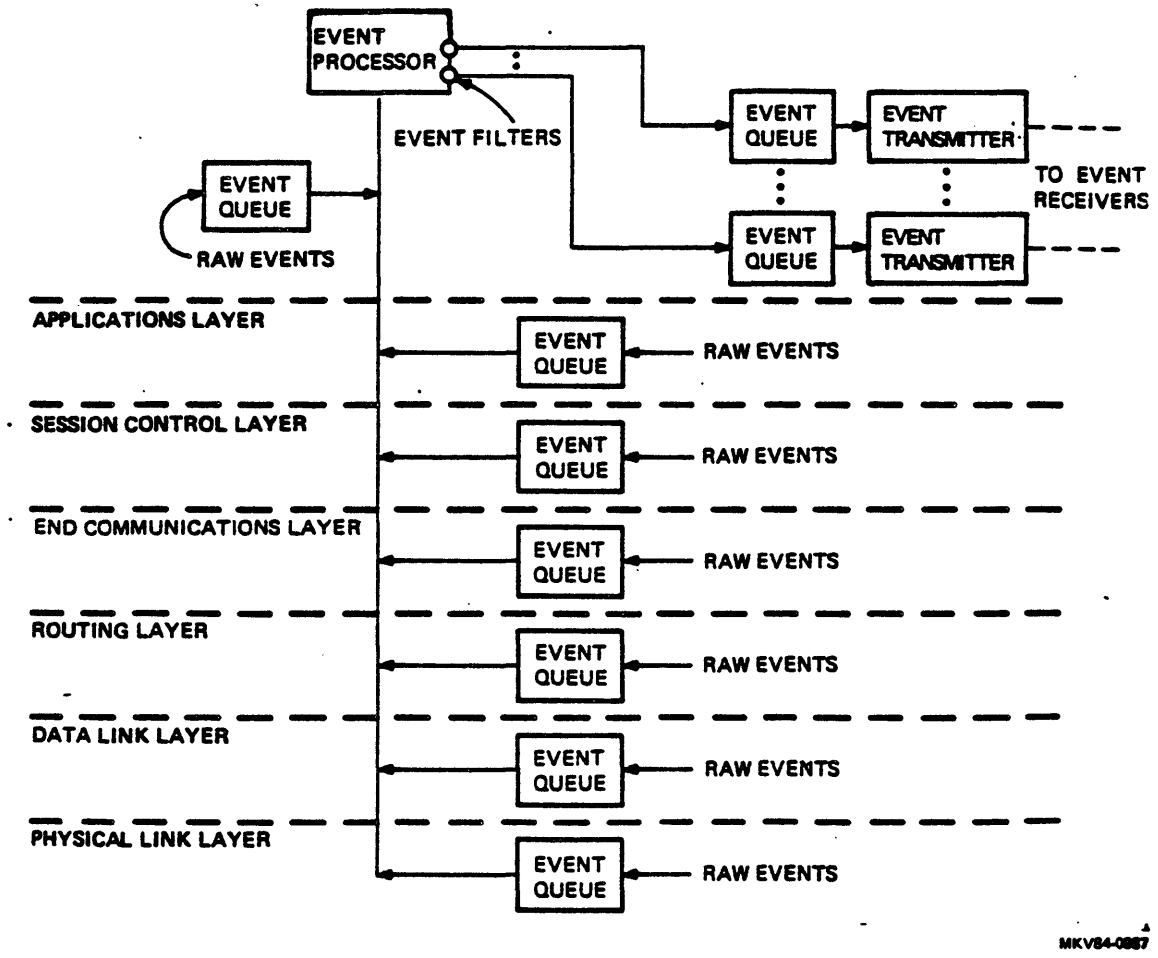


Figure 14 Event Logging Interface

Table 5 lists the supported event classes.

Table 5 Network Event Classes

Event Class	Source of the Event
0	Network Management Layer
1	Applications Layer
2	Session Control Layer
3	End Communications Layer
4	Routing Layer
5	Data Link Layer
6	Physical Link Layer
224 - 225	Server Base
226 - 227	DECnet Router
232 - 233	(Future use by SNA Gateway)
234 - 255	Reserved for future Server use

Example 9 shows some events as they appear on the host system console.

```
 Event type 0.3, Automatic service
Occurred 8-SEP-83 10:41:27 on node 4.25 (PIPPIN)
Circuit UNA-0
Service type = Load
Status = Requested
Node = 4.134 (MNCHKN), File = DL0:[100,54]PLUTO2.SYS
Software type = Secondary loader


 Event type 0.3, Automatic service
Occurred 8-SEP-83 10:41:27 on node 4.245 (PIPPIN)
Circuit UNA-0
Service type = Load
Status = Successful
Node = 4.134 (MNCHKN), File = DL0:[100,54]PLUTO2.SYS
Software type = Secondary loader


 Event type 0.3, Automatic service
Occurred 8-SEP-83 10:41:28 on node 4.245 (PIPPIN)
Circuit UNA-0
Service type = Load
Status = Requested
Node = 4.134 (MNCHKN), File = DL0:[100,54]PLUTO3.SYS
Software type = Tertiary loader


 Event type 0.3, Automatic service
Occurred 8-SEP-83 10:41:30 on node 4.245 (PIPPIN)
Circuit UNA-0
Service type = Load
Status = Successful
Node = 4.134 (MNCHKN), File = DL0:[100,54]PLUTO3.SYS
Software type = Tertiary loader


 Event type 0.3, Automatic service
Occurred 8-SEP-83 10:41:31 on node 4.245 (PIPPIN)
Circuit UNA-0
Service type = Load
Status = Requested
Node = 4.134 (MNCHKN), File = DL1:[4,4]MNCHKN.SYS
Software type = System
```

```
 Event type 0.3, Automatic service
Occurred 8-SEP-83 10:42:17 on node 4.245 (PIPPIN)
Circuit UNA-0
Service type = Load
Status = Successful
Node = 4.134 (MNCHKN), File = DL1:[4,4]MNCHKN.SYS
Software type = System


 Event type 4.15, Adjacency up
Occurred 8-SEP-83 10:42:32 on node 4.245 (PIPPIN)
Circuit UNA-0
Adjacent node = 4.134 (MNCHKN)
```

Example 9 Event Logging on the Host Console

Logging Network Events

Network events generated by the server are logged by default onto the logging console on RSX hosts or the logging monitor on VMS hosts. If the server's host is not available, the server sends the message to the first available node listed in the server's backup host list. Logging must be enabled at any host expected to receive a network event from a server.

```
NCP>SET LOGGING CONSOLE STATE ON (RSX)
```

Example 10 Using NCP to Set the Logging Console (RSX)

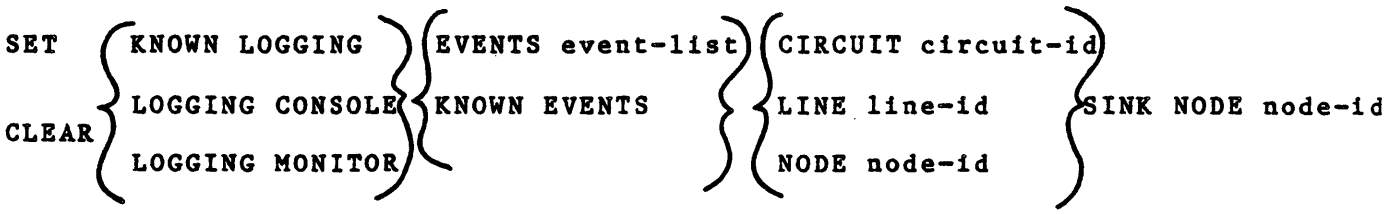
```
$NCP SET LOGGING MONITOR STATE ON (VMS)
$REPLY/ENABLE=NET
```

Example 11 Using NCP to Set the Logging Monitor (VMS)

User Options for Logging Events

The user has a number of options available for logging network events. Table 6 shows the logging options available to a user.

Table 6 User Options for Logging Events




```
NCP>SET KNOWN LOGGING KNOWN EVENTS CIRCUIT LC-2
```

Example 12 Using NCP to Set Up Network Event Logging

NOTE

Successive SET LOGGING commands applied to the same component and parameter have a cumulative affect. Each SET command issued will add new logging functions without overriding any functions previously set up. A user must explicitly cancel these functions with the CLEAR LOGGING commands.

The default SINK NODE when not explicitly stated is the servers' host or the backup host list.

ON-LINE DEBUGGING TOOL (POD)

The server on-line debugging tool (POD) is an analysis tool used to examine:

- Memory of a running server
- A server dump image
- The distribution image of a server product

For a running server system, POD may be used to make deposits into server memory.

Since the server does not support a console terminal, interactive support tools such as POD must be run on a Host system. The POD utility:

- Runs in cooperation with a server task, Remote Examine and Deposit (RED...), when examining the memory of a running server.
- Can only be run by a privileged user since it allows complete access to the running server's memory.
- Is transportable with the ability to run under RSX and VMS

When diagnosing problems, POD can be used to:

- Obtain information not provided by event logging or network management facilities. This includes examining:
  - Data structures
  - Trace buffers
  - Possibly corrupted code
- Insert "debug code" in a running server to determine if certain points in the code are reached, and to save the contents of pertinent data cells if needed. A scratch data area exists in the server's executive address space for this purpose.
- Tailor on-line and crash dump analysis report generation and allow symbolic, interactive examination of dump images and running server systems.

- Support command files and a data structure interpreter that allows analysis procedures to be pre-defined.

NOTE

Although POD can be used to insert DEBUG code, this feature is not intended to be used as a "patching mechanism" for the server products.

POD Help

```
  run pod
Pluto On-Line Debugger  -  V3.0.0
POD>help

          POD, (PLUTO On-Line Debugger)

The following is a list of available commands:

* @                          * SET LENGTH
* BIAS                       * SET LOG
* BREAK                      * SET OUTPUT
* CONVERT                    * SET REGION
* DEFINE                     * SET SERVER
* DEPOSIT                    * SET TYPE
* DISPLAY                    * SHOW IMAGE
* EXAMINE                    * SHOW LENGTH
* EXIT                       * SHOW LOG
* HELP                       * SHOW OUTPUT
* MAP                        * SHOW REGION
* MONITOR                    * SHOW SERVER
* NOBIAS                     * SHOW STRUCTURE
* NOMAP                      * SHOW SYMBOL
* SET ADDRESS                * SHOW TYPE
* SET IMAGE                  * UNDEFINE

POD>
```

Example 13 Available Help for POD

POD Functions

POD includes the following functionality:

- The capability to examine memory locations
- Make deposits into a running server system
- Monitor locations
- Command file support
- Log file support
- The capability to define/undefine user symbols
- Display data structures
- The ability to crash and reload a server

POD resides in the host system and is installed on the host during server installation. To run POD a user must be privileged on RSX or a VMS user must have either OPERATOR or DIAGNOSE privileges set.

POD Contexts

Before issuing any commands, a user must specify a set of contexts. A context indicates to POD where an examine or deposit is to take place. The user must specify the following contexts:

- Server - identifies the specific product (Router, Router/X.25 gateway, etc) currently running on the server in question.
- Image
  - A distribution image of a server product
  - A dump image from a crashed server
  - Memory of a running server (node name if the server is running)

- Region
  - The 22-bit address space in physical memory only (excludes the I/O page)
  - The executive address space (includes the I/O page)
  - The virtual address space of an RSX-11 task, common, DECnet process, etc.

Example 14 shows a sample POD session setting contexts:

```
>RUN POD
Pluto On-Line Debugger   -  V3.0.0
POD>SET SERVER ROUTER
POD>SET IMAGE NODE=MAUDE

                  DECnet Router Server V1.0
POD>SET REGION EXECUTIVE
POD>SHOW IMAGE
Current image is node MAUDE
Access: Examine
RED protocol version  1. 0. 0
Examine buffer size is  256 bytes
POD>
```

Example 14 Sample POD Session Setting Contexts (RSX)

POD Context Commands

- SET SERVER
  - Looks for a .DSD (Data Structure Definition) file.
  - Reads in the name of the directory where the structures are found.
  - Opens a .SYM (Symbol) file and loads the system symbol table which contains executive symbols.
  - These symbols can be used if the region is then set to the executive or to a region that is a privileged task.

- SET IMAGE
  - If the image is a running server node, POD communicates with RED...
  - If the image is a dump or distribution image, POD attempts to open the file.
  - POD displays the ID field in the DECnet Home Block to indicate whether the correct symbol (.SYM) file is being used.

- SET REGION
  - POD gets to the correct region, the symbol (.SYM) file for that region is read.

NOTE

For each SET command, there is a corresponding SHOW command (eg. SHOW SERVER, SHOW IMAGE, SHOW REGION).

POD Features and Commands

Command Syntax

Each POD command has the following format:

```
POD>Command [Keyword][/Qualifier][Parameter]
```

- Command - a verb indicating the general function to be performed
- Keyword - indicates the specific function to be performed
- Qualifier - modifies the effect of the command
- Parameter - qualifies the function in some way (eg. specifying a range of locations to be monitored)

Characters with special meanings (used as part of a qualifier or parameter):

- @ (contents operator) - requests POD to extract the contents of the location previously examined and use this as the next address to examine.
- : (range operator) - specifies a range of addresses for an EXAMINE command
- ! (comment indicator) - indicates that a comment follows

Examine

POD allows a user to display the contents of a specified address or range of addresses. The contents of the address can be displayed in:

- Numeric form (decimal, octal, or hexadecimal)
- ASCII
- RAD50
- Symbols (where possible)

The examine command may also be used to display formatted output as in the display of data structures.

NOTE

An EXAMINE range occurs as an atomic operation (uninterruptable) on a running server.

The EXAMINE command may be abbreviated to E.

All display formats are diplayed as either words or bytes, except for RAD50, which is displayed in words only.

```
POD>EXAMINE $LDBRT + 10

$LDBRT + 10/   177776
POD>EXAMINE/DECIMAL/RAD50 100:120

FE.PKT       /     7250.    DUJ
T.RDCT       /      224.     EX
T.EFLM       /      908.     V.
T.EFLM +   2/      225.     EY
T.GGF        /      908.     V.
T.LGTH       /      226.     EZ
T.LGTH +   2/      908.     V.
S$$IEN +   1/      227.     E$
S$$IEN +   3/    23264.    NUX
POD>EXAMINE .

FE.PKT       /    16122
T.RDCT       /      340
T.EFLM       /     1614
T.EFLM +   2/      341
T.GGF        /     1614
T.LGTH       /      342
T.LGTH +   2/     1614
S$$IEN +   1/      343
S$$IEN +   3/    55340
POD>E $PARHD

$PARHD       /   111734
POD>EXA/STRUCTURE=PCB @

        Partition Control Block
P.LNK =    111670
Priority =         0.
IO + IOSB count =          0.
Partition Name =    CEXPAR
P.SUB =    000000
P.MAIN =    111734
P.REL =    001120
P.SIZE =    000060
Wait Queue =    000000     111754
Swap Size =    111754
Busy Flags =    200     200
P.TCB =    000000
Starting APR =    0
        Common
P.HDR =    000000
Protection word =    177777
Attachment Descriptor Listhead =    000000     111772
POD>
```

Example 15 Using the Examine Command

Deposit

The DEPOSIT command allows a user to modify the contents of memory locations in a running server image. Deposited values may be in:

- Numeric form
- ASCII
- RAD50
- Symbolic format

The DEPOSIT facility is intended only to assist in diagnosing problems. Any inserted code should only be temporary. A scratch area exists in the server base for depositing Debug code (100 octal bytes starting at $PATCH). To use the DEPOSIT command, the /DEPOSIT switch must be specified when issuing the SET IMAGE command.

There are two modes for deposits: protected and unprotected. Protected deposits require that the user EXAMINE the location or range of locations before making a deposit. Unprotected deposits are made by RED... without any validation of the contents. This type of deposit is used when the server system changes the value of the desired location before deposits can be made. In this case, protected deposits are prevented, and the user is notified.

The DEPOSIT command can be abbreviated to D.

NOTE

The protected mode is the default value for deposits. The /UNPROTECT switch changes the default, but requires that the user specify the /UNPRODEP qualifier be used when issuing the SET IMAGE command.

```
POD>EXAMINE SYSPAS + 1
POD>DEPOSIT/BYTE/ASCII SYSPAS+1 T,E,S,T
```

Example 16 Depositing into Server Memory

RED... stores information on all deposits that have been made. To determine if a deposit has been performed use the following example:

```
POD>SET REGION RED...
POD>EXAMINE NUMDEP

NUMDEP      /        6
POD>EXAMINE DEPBAS

DEPBAS      /        1
POD>EXAMINE DEPADD

DEPADD      /      752
POD>EXAMINE DEPLEN

DEPLEN      /       14
POD>
```

Where:

NUMDEP - The number of deposits that have occurred.

The following contain data about the last deposit only.

DEPBAS - 0 = Physical Region
1 = Executive Region
or the base address of the partition

DEPADD (two words) - address of start of deposit

DEPLEN - length of the deposit (top bit set if unprotected)

Example 17 Examining RED... for Deposits

Monitor

The MONITOR command allows a user to monitor up to five locations during a specified time interval. The MONITOR command also indicates any changes in the contents of the location(s). The contents of the specified locations are displayed initially, and then again if the contents change (the locations are monitored periodically not continuously). This command is allowed on a running server system only.

NOTE

The default setting for the MONITOR command has POD re-examine the locations once per second for 30. seconds.

```
POD>MONITOR/INTERVAL=5/TIME=100 $ZTIME+2, $PATCH
```

Example 18 Monitoring Server Memory Locations

Command File Support

Allows a user to define standard procedures for analyzing data. Any valid POD command can be used in a command procedure.

NOTE

There is no indirect command support for POD command files (eg. there is no support for ifs, elses, etc.)

POD.CMD

```
SHO LOG
SET SERVER ROUTER
SET IMAGE/DEPOSIT/UNPROTECT/ NODE=BARNEY
SET REGION EXECUTIVE
EXAMINE $PARHD
EXAMINE/STRUCTURE=PCB @
EXIT
```

```
@POD
```

Example 19 POD Command File

Log File

The Log File allows users to send output to a terminal, log file, or both. There are two commands used to control where the output will go. The SET LOG command identifies the log file where the output will be sent. The SET OUTPUT command determines whether the output goes to the terminal, log file, or both.

```
POD>SET LOG ROUTER.LOG
POD>SET OUTPUT LOG
```

Example 20 Setting Up a Log File for Output

Symbol Table Support

If a user wishes to examine or deposit into a specific location in memory, it may be specified either in octal or by a symbolic expression. POD maintains three symbol tables:

- User Symbol Table
  - Contains symbols defined by the user using the DEFINE command.
- Region Symbol Table
  - Symbols that depend on the particular region specified and whose context changes with the region context.
  - When the region is changed (<SET REGION>), a file PODxxx.SYM (xxx = RTR, X25, etc.) is read to determine the proper .STB file for the new region. Since each server product includes its own set of regions, there is a specific .SYM file for each server product (PODRTR, PODX25, etc.).
- System Symbol Table
  - Contains Executive symbols.

- System Symbol Table
  - Contains Executive symbols.
  - This table never changes.

Define

The DEFINE command allows a user to define a symbol by assigning it a 16-bit alphanumeric value. If the user attempts to define a symbol that is already used, POD displays a warning message. User symbols may not be defined more than once without first being UNDEFINED. When a symbol is referenced, the User Symbol Table is searched first, then the Region Symbol Table. If an exact match for a symbol is not found, the symbol with the closest but lower value is displayed with an octal offset. If a symbol has been defined in the user symbol table as well as in either of the other two tables then the symbol in the user table takes precedence.

```
POD>DEFINE TEMP=10
```

Example 21 Defining POD Symbols

Data Structure Interpreter Support

- Displays data in a numerical format in octal, decimal, or hexadecimal numbers.
- Displays data in ASCII or RAD50 strings.
- Displays data in symbols or offsets from symbols.
- Assigns ASCII text to values that the data can have.
- Interprets data as a field of bits, bytes, words, or double words.

In example 22, the data structure ZBLOCK starting at address FOO is examined.

```
POD>EXAMINE/STRUCTURE=ZBLOCK FOO
```

Example 22 POD Command for Examining a Data Structure

When examining data structures, a user may wish to follow pointers from one structure to another. POD allows this through the CHAIN command.

```
POD>EXAMINE $PARHD

$PARHD       /   111734
POD>EXAMINE/STRUCTURE=PCB @

        Partition Control Block
P.LNK =     111670
Priority =          0.
IO + IOSB count =          0.
Partition Name =     CEXPAR
P.SUB =     000000
P.MAIN =     111734
P.REL =     001120
P.SIZE =     000060
Wait Queue =     000000      111754
Swap Size =     111754
Busy Flags =     200      200
P.TCB =     000000
Starting APR =     0
        Common
P.HDR =     000000
Protection word =     177777
Attachment Descriptor Listhead =     000000      111772
POD>CHAIN

        Partition Control Block
P.LNK =     111624
Priority =          0.
IO + IOSB count =          0.
Partition Name =     NTPOOL
P.SUB =     057524
P.MAIN =     111670
P.REL =     001200
P.SIZE =     003740
Wait Queue =     000000      111710
Swap Size =     111710
Busy Flags =     200      200
P.TCB =     000000
Starting APR =     0
        System controlled
P.HDR =     000000
Protection word =     000000
Attachment Descriptor Listhead =     000000      111726
POD>
```

Example 23 Using the Chain Command

CRASH DUMP ANALYSIS

The server software supports up-line crash dumps over the Ethernet. Whenever the server node fails, it automatically dumps its memory into a file on a host node. The file can be dumped to the designated host, or if the designated host is not available, the file can be dumped to any available host listed in the server's backup host list. Once dumped, the file can be analyzed by POD.

Prerequisites for Up-line Dump

Setting up the data base for performing the up-line dump depends on the host operating system. RSX and TOPS hosts use the CFE utility, VMS hosts use NCP.

RSX/TOPS host commands:

- Circuit UNA-0 must be set to the ON state and SERVICE ENABLED
- Enter the command:

```
CFE>DEFINE NODE 136 NAME BARNEY -
    DUMP FILE BARNEY.DMP -
    HARDWARE ADDRESS AA-00-03-00-00-06 -
    HOST RSXHOST -
    DIAGNOSTIC FILE LB:[100,100]CSVLDI.SYS -
    LOAD FILE LB:[100,100]CSVRTR.SYS -
    SERVICE CIRCUIT UNA-0 -
    SERVICE PASSWORD (HEX) -
    SECONDARY LOADER [100,100]PLUTO2.SYS
    TERTIARY LOADER [100,100]PLUTO3.SYS
```

Example 24 Setting Up the Up-line Dump File on RSX Hosts

VMS host commands:

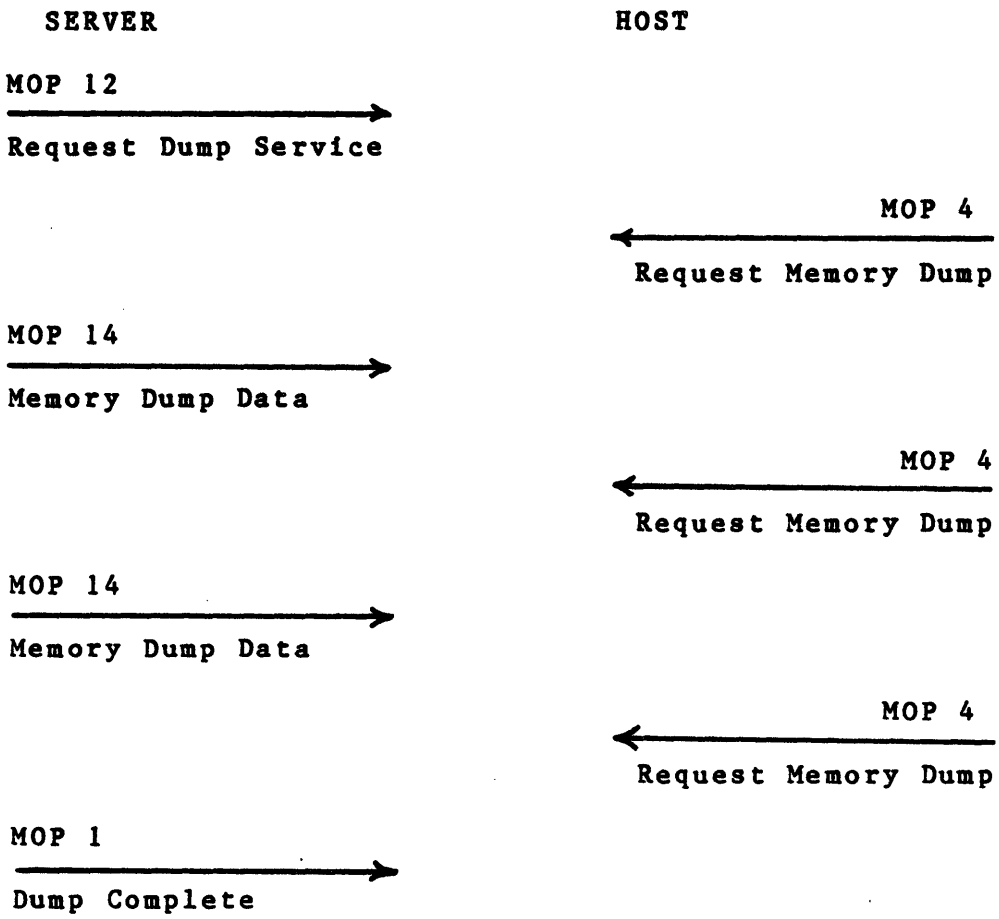
- Circuit UNA-0 must be ON and ENABLED for SERVICE functions
- Enter the command:

```
NCP>DEFINE NODE 134 NAME BARNEY -
    DUMP FILE BARNEY.DMP -
    HARDWARE ADDRESS AA-00-03-00-00-06 -
    HOST VMSHOST -
    DIAGNOSTIC FILE LB:[100,100]CSVLDI.SYS -
    LOAD FILE LB:[100,100]CSVRTR.SYS -
    SERVICE CIRCUIT UNA-0 -
    SERVICE PASSWORD (HEX) -
    SECONDARY LOADER LB:[100,100]PLUTO2.SYS -
    TERTIARY LOADER LB:[100,100]PLUTO3.SYS
```

Example 25 Setting Up the Up-line Dump File on VMS Hosts

Up-line Dump

When the server detects a condition that it cannot recover from, it requests the loading host to accept an up-line dump. If that host is unable to accept the dump, the server asks the first host on its backup list (obtained from the configuration file loaded with the server software). If every host on the backup list rejects the dump, the server sends it to a multicast address; the first available host that is able to accept the dump will respond.



Example 26 Message Exchange During an Up-line Dump

Forcing a Crash Dump

When the server is up and running but no activity is taking place (the server seems to "hang"), it is necessary to force a crash dump to determine the cause of the problem. Forcing the server to crash depends on the host operating system and also uses the Remote Console Facility of the server.

The Remote Console Facility (RCF) allows a user to set up a logical connection between a terminal connected to a host and the server hardware unit's console interface. This allows the remote terminal to act as the server's console. Table 7 shows the command format for running CCR.

Table 7 CCR Command Format

```
CCR NODE node-id [SERVICE PASSWORD password]
                 [SERVICE CIRCUIT circuit-id]
                 [PHYSICAL ADDRESS ethernet-address]
```

NOTE

The Service password, circuit, and address only need to be specified if they are not defined in the host's down-line load database for the server.

Invoking CCR on an RSX Host

CCR can be invoked by:

>RUN CCR (if it is not installed)

or

>CCR NODE node-id (if it is installed).

The server responds with:

CCR>

Once the connection to the server node has been established the following mesage will be displayed:

CCR -- Remote Console is Reserved

VMS Hosts

A terminal that is connected to a VMS host can remotely connect to the server's console interface by using the NCP CONNECT command. The CCR and the CCS image are installed in SYS$SYSTEM by VMSINSTAL. In VMS the CCS serves as the interface to NCP. To connect the VMS host to the server, one of the following commands is used:

```
NCP>CONNECT NODE node-id
```

or

```
NCP>CONNECT VIA CIRCUIT UNA-0 PHYSICAL ADDRESS AA-00-04-00-B6-04
```

Once the connection has been established, the server responds with:

```
Console Connected (press CTRL/D when finished)
```

Once the host (either RSX or VMS) is connected to the server's console; Table 8 shows the procedure that will force a crash dump.

Table 8 Commands and Server Responses to Force a Crash Dump

User Command	Server Display	Server Response
<CNTRL/B>	Current PC address	Initiates a break in processing
	@	Indicates the server has halted
@100/	@100/address	An octal address
<RET>	@	
@address/	@address/contents	Contents of the location is displayed
@addr/cont 3 <RET> *		
@P	[Server node crashes]	Causes the server processor to continue

NOTE

The 3 that is entered is the address of a breakpoint trap. When the trap is encountered, the server will begin to dump. The front panel of the server will diplay "d" followed by a series of digits. There will also be event messages on the host indicating that the server memory is being dumped. Example 27 shows a sample VMS session forcing a crash dump:

```
$NCP CONNECT NODE ROUTER

Console Connected (type ^D when done)

^B

001260

@100/016122

@016122/004567 3 <RET>

@P

^D

 $
```

Example 27 Forcing a Crash Dump from a VMS Host

Analyzing a Crash Dump

When the server crashes, a dump file is sent to a host node. This name of the dump file is specified in the down-line load/up-line dump database. This file can be analyzed by using the POD facility.

```
>RUN POD
POD>SET SERVER ROUTER
POD>SET IMAGE DUMP = [Accnt]BARNEY.DMP
```

Example 28 Analyzing a Server Dump using POD

Draft Software Product Description

PRODUCT NAME: DECnet Router, Version 1.0 SPD # 30.34.00

DESCRIPTION:

The DECnet Router Server is a software product that runs on an Ethernet Communications Server hardware unit to provide DECnet routing functions in a network of one or more host computers -- be they Phase IV routing nodes or endnodes, or Phase III routing nodes or endnodes (e.g., DECnet-RT V2.0 and DECnet/E V3.0). The DECnet Router Server connects directly to the Ethernet to provide routing to nodes off the Ethernet and connected via the unit's synchronous lines. These nodes can be remote Phase III/IV routing nodes or endnodes, or other DECnet Router Servers connected to other Ethernets. Endnodes connected directly to an Ethernet must use DECnet Router Servers or Phase IV host routing nodes connected to the same Ethernet to enable message routing off that Ethernet. A routing node is not required on an Ethernet if the endnodes connected to that Ethernet communicate only with each other. Use of the DECnet Router Server enables the offloading of certain communications processing on host nodes that would otherwise serve as routing nodes on the Ethernet.

The DECnet Router Server implements Phase IV DECnet routing and network management. Through the use of Phase IV DECnet protocols, DECnet computer networks can contain up to 1023 nodes given proper network planning. Along with providing Ethernet support, the DECnet Router Server is compatible with Phase III implementations; this provides migration of Phase III networks with connectivity to Phase IV Ethernet nodes.

The DECnet Router Server is warranted for use only with supported Phase III and Phase IV products supplied by DIGITAL.

Adaptive Routing

The DECnet Router Server functions as a dedicated routing node with one or more communications lines connecting nodes off an Ethernet to nodes on that Ethernet. With adaptive routing, a message received from a Phase IV/III node addressed to another Phase IV/III node will be forwarded by the intermediate Phase IV/III nodes -- in this case the Router Server. Routing is not supported to or from Phase II nodes.

Although two adjacent routing nodes can be connected by more than one physical link, messages will be sent over only one of the links. All other lines will serve as "hot standbys" such that the least cost path available between two nodes is the one that will be used for message traffic. A line cost parameter set by the system manager determines the line over which all messages will be sent from one node to an adjacent node.

Network Management

The Network Management Utility on any Phase IV DECnet host node can execute commands remotely at the DECnet Router Server to perform three primary functions: display statistical and error information, control the operation of the server, and test network operation. All network management functions are performed from a node logically connected to the server, rather than from the server itself.

A remote operator can choose to display statistics related to the server node itself or the communications lines, including traffic and error data. The remote operator can also perform many network control functions such as starting and stopping lines, and downline loading the Router software into the server.

A remote operator can log network events (for the DECnet Router server node) to a terminal device or other event sink on a Phase IV DECnet host node supporting this capability. The Network Management utility can be used to enable and disable the event logging facility at the host.

Network Management can also be used to test components of the network. A remote operator can transmit and receive test messages over individual server lines between the server and remote nodes. The messages can then be compared for possible errors. Network Management allows a logical series of tests to be performed, thereby aiding in isolating network problems.

Communications

The DECnet Router Server uses Ethernet Communications Server line cards to connect network nodes. The Communications Server line cards connect to an intelligent controller within the unit that provides DMA access to server memory and implements line control and error recovery procedures. This module implements the Digital Data Communications Message Protocol (DDCMP) to provide full- or half-duplex communications over point-to-point synchronous lines. The Communications Server line cards interface to standard DIGITAL synchronous line controllers (e.g., DMR11) via DIGITAL-supplied EIA RS-232C/CCITT V.24 and V.35 cables. Note that the Communications Server does not support asynchronous DDCMP lines or multipoint lines.

The Ethernet Communications Server interfaces to the Ethernet via an intelligent Ethernet controller packaged with the hardware unit.

DECnet Router Server Operation

The Ethernet Communications Server hardware provides the necessary maintenance operation protocols for loading DECnet Router software from an Ethernet load host, over the Ethernet, and into server memory. All software, including diagnostics, are downline loaded into the unit. In the event of certain hardware or software malfunction, the unit will attempt to upline dump the Router server software and

automatically invoke reload.

A configuration file on one or more host load nodes defines parameters for the operation of the Router server. This information is defined at installation time and can be modified by the System/Network Manager for reconfiguring the server upon subsequent reloading.

Router Server Configuration and Performance

The process of configuring the DECnet Router node is based primarily on tradeoffs of cost and performance within the realm of satisfying the users' application requirements. It can be expected that network applications will range from low-speed, low-cost situations (e.g., connecting few remote nodes over low speed communications lines) to those of relatively high performance (e.g., connecting two Ethernets). Primarily, the performance of a given DECnet Router node is a function of the expected network traffic and resultant processing pursuant to the dedicated function of the unit. Thus node performance depends on several factors:

- Communication line characteristics
- Size of buffers
- Quantity and frequency of route-through traffic

It is important to note that the rate at which user data can be transmitted (throughput) over a communications line can sometimes approach, but will never reach, the actual line speed. The actual throughput is a function of many factors, including the line quality, protocol overhead, topology, and network application(s), as well as the factors cited in this section.

The table below describes the physical hardware configurations supported by the DECnet Router server in terms of communications interface.

	Maximum Line Speed (Kilobits/sec)			
Line Card Type	up to 19.2	56	250	500
DCSAX-LA (FDX/HDX)	8	N/A	N/A	N/A
DCSAX-LB (FDX/HDX)	N/A	8	2	1

Note that the total number of line cards supported by the hardware unit is eight, and that the aggregate line throughput cannot exceed 500 kilobits per second.

In order to achieve a viable configuration, the user and /or a DIGITAL software specialist should perform a level of application analysis which addresses the factors above. In the preceding table, the following definition applies: Maximum line speed -- the fastest clock rate at which the number of line cards specified can be driven under the DECnet Router server. The actual maximum data throughput can not be calculated by multiplying the number of lines by the line speed, since many factors already discussed in this section will reduce the actual throughput.

MINIMUM HARDWARE REQUIRED:

The DECnet Router software runs on the following Ethernet Communications Server packaged hardware option:

- o DECSA-EA — Ethernet Communications Server hardware, including one DCSAX-LA line card (single line EIA RS-232C/CCITT V.24 synchronous to 19.2K bps, FDX/HDX).

One of the following cables should be used with the DCSAX-LA line card:

- o BC17D — Null modem cable for local connections
- o BC17C — EIA extension cable for remote (modem) connections

The Ethernet Communications Server hardware requires both a transceiver drop cable and Ethernet interface -- H4000 or DELNI -- to connect to the Ethernet physical channel.

Because all software to run on the Ethernet Communications Server is downline loaded from a DECnet Phase IV Ethernet load node, this node must have a magnetic tape or RL02 disk drive to read the distribution media. This node should also have 1200K bytes of disk space to store the Router software for subsequent downline loading.

OPTIONAL HARDWARE:

Additional Ethernet Communications Server line cards (DCSAX-LA and/or DCSAX-LB), up to the maximum defined in the Configuration and Performance section of this SPD, can be added to the packaged hardware. Line card types:

- o DCSAX-LA — Single line EIA RS-232C/CCITT V.24 synchronous to 19.2K bps, FDX/HDX. Use BC17D (null modem) or BC17C (EIA extension) cables.
- o DCSAX-LB — Single line CCITT V.35 synchronous to 500K bps,

FDX/HDX. Use BC17E (full modem) cables.

PREREQUISITE SOFTWARE:

Phase IV DECnet support on one or more Ethernet Communications Server load hosts (on the Ethernet). These include:

DECnet-RSX (V4.0 - 11-M/V2.0 - 11-M-PLUS)
DECnet-VAX V3.1

OPTIONAL SOFTWARE:

None

TRAINING CREDITS:

No training credits are included with an Ethernet Communications Server software license. Training courses on DECnet software are scheduled at regular intervals in DIGITAL's Training Centers. Arrangements should be made directly with DIGITAL's Educational Services Department.

SOFTWARE CATEGORY:

DIGITAL SUPPORTED

The DECnet Router Server is a DIGITAL Supported Software Product.

If DIGITAL Installation Service has been provided, then during the ninety (90) day period following installation, DIGITAL will provide the following standard services if the customer encounters a problem with the Software Product:

(a) If DIGITAL also determines the problem to be a defect in the Software Product, DIGITAL will provide remedial service on site if necessary (1) to apply a temporary correction or make a reasonable attempt to develop an emergency bypass if the software is inoperable, and (2) assist the customer in preparing a Software Performance Report (SPR).

If DIGITAL Installation Service has not been provided, then during the ninety (90) day period following installation, either from the date of first use by the customer or thirty (30) days after delivery (F.O.B. Digital's plants), whichever occurs first, if the customer encounters a non-installation related problem with the software product, DIGITAL will provide the following warranty services.

(b) If customer diagnosis indicates the problem is caused by a defect in the Software Product, it may submit an SPR to DIGITAL.

DIGITAL will respond to problems reported in SPRs that are caused by defects in the current, unaltered release of the Software Product in a manner appropriate to the severity of the problem.

Telephone support is available, at no additional charge, from the DIGITAL Telephone Support Center for a ninety (90) day period commencing either from the date of first use of this Software Product by the customer or thirty (30) days after delivery, whichever comes first.

Any Updates to the Software Product released by DIGITAL during the ninety (90) day period will be provided to the customer on DIGITAL standard distribution media as specified in this SPD.

These services will be provided in locations within the United States, except Alaska. Otherwise, service will be provided in the country of purchase. Service required because of customer use of other than the current, unaltered release of the Software Product operated in accordance with the SPD will be provided at DIGITAL's then current rates, terms, and conditions.

SOFTWARE INSTALLATION:

CUSTOMER INSTALLED

The DECnet Router Server is a Customer Installed Software Product. There is an optional installation service available for Server software products.

Optional DIGITAL installation is recommended for this software product. This service is described under the "Additional Services" section of this Software Product Description.

DIGITAL recommends that only those customers who have sufficient technical resources should attempt its installation without DIGITAL's assistance.

DIGITAL Installation Service is available, provided the customer requests this service within 30 days of product delivery (F.O.B. Digital plants).

WARRANTY LIMITATIONS:

The customer may purchase DECnet Router Server licenses with options that do not include support services. The category of support applicable to such software is Customer Supported. When a network contains DECnet product options, which were purchased without warranty services and/or are not currently covered by a DIGITAL Software Product Service contract, DIGITAL will respond to only those problems that occur or can be demonstrated by the customer to occur among nodes that are under warranty service or a current DIGITAL Software Product Service Contract.

PREREQUISITE SUPPORT:

A Network Profile and DECnet Customer Support Plan covering all intended network nodes and their support may be required.

ORDERING INFORMATION:

All binary licensed software, including any subsequent updates, is furnished under the licensing provisions of DIGITAL's Standard Terms and Conditions of Sale, which provide in part that the software and any part thereof may be used on only the single server CPU on which the software is first executed, and may be copied, in whole or in part (with the proper inclusion of the DIGITAL copyright notice and any DIGITAL proprietary notices on the software) only for use on such CPU.

Options with no support services are available only after the purchase of one supported license.

A single-use, license-only option is a license to copy the software previously obtained under license.

The following key (H, M, Z) represents the distribution media for the product and must be specified at the end of the order number, e.g., QE725-AH = binaries on RL02 disk cartridge.

H = RL02 disk cartridge
M = 9-track 1600 BPI magtape (PE)
Z = No hardware dependency

Refer to the following table for available options. Note that the options are defined by the load host upon which the software will be installed for subsequent Ethernet loading into the Ethernet Communications Server.

Table

Options	Load Hosts			
	VAX/VMS			RSX-11M/ M-PLUS
	11/730	11/750	11/780	
Single use license	QC725-AH	QD725-Ax	QE725-Ax	QP725-Ax
Single use license-only	QLA01-DZ			
Update/Unsupported	QC725-HH	QD725-Hx	QE725-Hx	QP725-Hx

Update, license-only	QC725-HZ	QD725-HZ	QE725-HZ	QP725-HZ
Documentation	QC725-GZ	QD725-GZ	QE725-GZ	QP725-GZ

ADDITIONAL SERVICES:

The following post-warranty Software Product Services for the software product are available to licensed customers:

- Self-Maintenance Service
- Basic Service
- DECsupport Service

Customers should contact their local DIGITAL office for additional information on the availability of these services.

Installation Services

For a fixed price a DIGITAL Software Specialist will assure that the customer's system is ready for installation, install the software, and familiarize the customer with its operation.

Installation for the DECnet Router Server will consist of the following:

- Verification that all components of the DECnet Router Server have been received.
- Verification that the necessary versions of the host software and documentation are available.
- Creation of the necessary DECnet Router Server accounts and directories.
- Creation of the Network Configuration File.
- Installation of DECnet Router Server software.
- Verification of the proper installation for the DECnet Router Server by running a series of tests to show connectivity to a hosts connected to the DECnet Router.

  Connectivity to all other nodes within the network is the responsibility of the customer.

Pre-Installation Procedures Required

Before DIGITAL can install the software, the customer must:

- Ensure that the system meets the minimum hardware and software

requirements (as specified in the SPD).

- Obtain, install, and demonstrate as operational any modems and other equipment and facilities necessary to interface DIGITAL's communication equipment.

- Designate one node to verify installation/connectivity.

- Make available for a reasonable period of time, as mutually agreed upon by DIGITAL and the customer, all hardware communications facilities and terminals that are to be used during installation.

Delays caused by any failure to meet these responsibilities will be charged at the prevailing rate for time and materials.

Pluto On-Line Debugger

User's Guide

Version 3.0.0

Date: 12-Apr-84

Kim Pyle

Distributed Systems Software Engineering, TWO/E07

1.0 OVERVIEW

The Pluto On-line Debugger (POD) is an analysis tool for examining the memory of a running server system, a server dump image, or the distribution image for a server product. And in the case of a running server systems, deposits are allowed into memory.

The Pluto hardware configuration does not include a console terminal, nor does the Server Base include any software which communicates with a console terminal. Therefore, any general interactive support tools must be run from a host system. POD runs on such a host system in cooperation with a Remote Examine/Deposit server task (RED...) which is part of the Server Base.

POD has been developed to support the following areas of diagnosing software problems in Pluto nodes:

1. Obtain information not provided by event logging or by network management commands. This includes the examination of data structures, trace buffers, possibly corrupted code, etc.

2. Insertion of debug code in a running server system to determine if (and perhaps how often) certain points in the code are reached, and to save the contents of pertinent data cells at such an event. Insertion of debug code can also be used to force a crash dump at a selected point. A scratch data area exists in the Server Base for this purpose. Starting at location $PATCH, there are 100 octal bytes available for code insertion.

3. Allow the tailoring of on-line and crash dump analysis report generation and to allow symbolic, interactive examination of dump images and running Pluto systems.

POD supports command files and formats data structures to allow analysis procedures to be pre-defined and easily invoked. References to memory locations may be made using symbols. Several data types are supported for the entry and display of data. POD is implemented in BLISS and runs under RSX-11M/M-Plus, and VMS. POD.TSK or POD.EXE reside in a standard Pluto directory on the host node, along with a .SYM file (containing symbol information) and a .DSD file (containing data structure information).

Before issuing any examine commands a server type, image and region context must be established.

A server type may be set to one of the following:

1. Router
2. Terminal Server
3. X25 Gateway/Router
4. SNA Gateway

An image context may be set to one of the following:

1. a distribution image of a server product
2. a dump image from a crashed Pluto node
3. memory in a running Pluto node

A region context may be set to one of the following:

1. the 22-bit physical address space (Excluding the I/O page)
2. the executive virtual address space (Including the I/O page)
3. the virtual address space of an RSX-11S task, common, DECnet process, etc.

2.0 SECURITY

The POD utility allows its user to examine or deposit into any memory location in a running Pluto. An unauthorized user must be prevented from engaging in such activities as "spying" on a data entry session at a Voyager terminal, or from modifying such input data, or from crashing the node at will. Therefore, POD access to a running server node will require that the user have an appropriate level of privileges at the host. On a VMS host, the user must have operator privilege, and on RSX, the user must be running POD from a privileged account.

Examine/deposit requests from POD are serviced over a logical link by the RED... task running in a server. Only one such logical link may be open at a time.

3.0 DATA TYPES FOR ENTRY AND DISPLAY

Data that is entered and displayed during POD operations is interpreted under the current "TYPE" and "LENGTH" settings. These are user-settable parameters which initially default to "octal" and "word" respectively. These defaults may be changed with the SET TYPE and SET LENGTH commands. Additionally, a type and/or length to be used during the execution of a particular command can be specified by applying a qualifier to that command. See the description of the EXAMINE or DEPOSIT commands for specific information on the data types and lengths supported.

When data is being interpreted numerically on output, all decimal values are treated as signed numbers and all other numeric values handled as unsigned numbers. On input, however, numerical data supplied as command parameters are treated as unsigned numbers and are translated from ASCII according to the input radix specified in the SET TYPE command.

4.0 COMMAND FILE SUPPORT

POD includes command file support. This gives the user a mechanism for defining standard procedures for analyzing data. Any valid POD command can be used in a command procedure.

5.0 DATA STRUCTURE SUPPORT

POD allows the user to examine data structures producing an appropriately formatted display. This syntax supports the following capabilities:

1. Displaying data in a numerical format as octal, decimal, or hexadecimal numbers.

2. Displaying data as ASCII or RAD50 strings.

3. Displaying data as symbols or offsets from symbols.

4. The user can also assign ASCII text to values that the data can have.

5. Data can be interpreted as a field of bits, a byte or a word.

6.0 SYMBOL TABLE SUPPORT

When a location is specified in an examine or deposit operation, that address may be specified in terms of an octal or symbolic expression. Such an expression is comprised of one or two terms, separated by a "+" or "-". A term may be an octal number or a symbol.

POD maintains three symbol tables - a User Symbol Table, containing symbols that can be DEFINEd and UNDEFINed by the user, a Region Symbol Table, whose content changes with region context, and a System Symbol Table. When region context changes (by issuing a SET REGION command), the .SYM file will be read to determine what symbol table information should be loaded for a particular region. Since each server product will include a different set of regions, there will be a specific .SYM file for each server product (PODRTR.SYM, PODSNA.SYM, PODX25.SYM, PODTS.SYM).

Symbol naming must follow the conventions stated in the description of the DEFINE command. Each symbol has a 16-bit value associated with it. To resolve a symbol reference, the User Symbol Table is searched first, then the Region Symbol Table and finally the System Symbol Table. When a value is being displayed in symbolic mode, and an exact match cannot be found in any symbol table, that symbol which has the closest but lower value (within 100 octal) will be displayed with an octal offset. (eg. $XYZ + 40)

7.0 COMMAND SYNTAX AND FUNCTIONALITY

The following section describes the commands available in POD, including a detailed description of what the command does for the user and the particular syntax for the command.

Each command has the following format:

command [keyword] [/qualifier] [parameter]

command - A verb indicating the general function to be performed.

keyword - Indicates, in conjunction with command verb, the specific function to be performed by the command.

/qualifier - Modifies the effect of the command.

parameter - Qualifies the function in some way, such as specifying a range of locations to be monitored.

Also the following characters, if used as a part of a qualifier or parameter, are interpreted to have special meaning:

1. @ (Contents Operator) - requests the debugger to extract the contents of the location previously examined and use this as the next address to examine.

2. : (Range Operator) - is used to specify a range of addresses for an EXAMINE command.

3. ! (Comment Indicator) - indicates that a comment is to follow.

8.0 SUMMARY OF COMMANDS

```
@
BREAK
CHAIN
CONVERT
DEFINE
DEPOSIT
DISPLAY
EXAMINE
EXIT
HELP
MAP
MONITOR
NOBIAS
NOMAP
SET
        IMAGE
        LENGTH
        LOG
        OUTPUT
        REGION
        SERVER
        TYPE
SHOW
        IMAGE
        LENGTH
        LOG
        OUTPUT
        REGION
        SERVER
        STRUCTURE
        SYMBOL
        TYPE
UNDEFINE
```

The following sections describe the commands supported. (For examples of their use see Appendix B.)

9.0 COMMANDS

9.1 @

Description:

The @ command instructs POD to begin taking commands from the indicated file. Command procedures, also called indirect command files, can be invoked wherever any other command can be given and can contain any valid POD command.

Format:

```
@file-specification
```

Command Parameters:

file-specification

Specifies the name of the command procedure to be executed. If a file type is not specified, the debugger will search for a file with the .COM type on a VMS host and a .CMD type on an RSX host. The file specification must be in the appropriate format for the host operating system that POD is running under.

Filespecs must be entered as quoted strings, unless the user is just specifying a filename and type.

Command Qualifiers:

None.

Example:

```
@POD.COM
```

9.2 BIAS

Description:

Specifies a value to be used for mapping into physical memory. It is intended to be used to emulate the mapping provided by the extended pool mechanism ($XBIAS et. al.) in the Server Base. Once the bias command has been issued, any odd address(es) specified during a command (such as EXAMINE and DEPOSIT), causes the virtual address to be translated into a 22-bit address. The low bit of the virtual address is cleared and the bias value specified is used as a block number in translating to 22-bit address, and the location referenced will be examined.

Note: If a byte length has been specified on an EXAMINE command (either by a qualifier or a previous SET LENGTH command), then a /BIAS switch must also be specified. Otherwise the virtual address will just be treated as a byte address, not as a bias specific address.

(NOTE: The BIAS command can be abbreviated to one character, B.)

Format:

BIAS bias-value

Command Parameters:

bias-value

The bias-value specified can either be a 16-bit octal number or a symbol. If a symbol is specified, the contents of the symbol is used as the bias-value, not the symbol's address. And in the case of an octal number, the number itself is used as the bias-value. (Note: $XBIAS is usually used as the bias-value. And no check is performed to determine whether the bias-value could cause the physical address(es) to go beyond the physical memory limit. If this occurs, an error message will be displayed when attempting to EXAMINE or DEPOSIT using bias.)

Command Qualifiers:

None.

Example:

BIAS $XBIAS

9.3 BREAK

Description:

The BREAK command will transmit a message to a running server node which will cause the system to crash and hence up-line dump. This command is only applicable when working against a running node and when the /BREAK has been specified on the last SET IMAGE command. (See the SET IMAGE command for further details.)

Format:

BREAK

Command Parameters:

None.

Command Qualifiers:

None.

Example:

BREAK

9.4 CHAIN

Description:

Allows the user to chain through data structures. A reference name must have been previously specified as part of the structure definition. If this has been done, you can use the reference name specified to look at a structure of a different type, which is pointed to by the structure you just examined.

If you did not previously examine a structure, this command is invalid and an error message will be displayed.

The CHAIN command can also be used to look at the next structure of the same type you are currently examining. This is done by not specifying a reference name when you enter the command. (Note: This is equivalent to performing a EXAMINE <cr> (next operation) if your display type is STRUCTURE).

(Note: Structure names and reference names are described in Appendix A.)

Format:

CHAIN [structure-reference-name]

Command Parameters:

structure-reference-name

The reference name is optional, and is the name of the field through which the referecne occurs. (This is defined when the data structure definition is written.)

Command Qualifiers:

None.

Example:

CHAIN P.TCB

9.5 CONVERT

Description:

The CONVERT command displays the octal, decimal, hexadecimal, ASCII, RAD50, and symbolic equivalent for the specified value. The value specified is interpreted based on the current input TYPE.

Format:

CONVERT value

Command Parameters:

value

An input value that is interpreted according to the first type parameter specified in the last SET TYPE command. (see the SET TYPE command for further details.)

Command Qualifiers:

None.

Example:

CONVERT 100

9.6 DEFINE

Description:

The DEFINE command lets you define a symbol and assign a 16 bit value to the symbol. A warning will be displayed if a user symbol, being DEFINEd, is previously defined in either of the other two symbol tables. A user symbol cannot be defined more than once. The user must first UNDEFINE the symbol before attempting to redefine it.

The debugger always searches for user defined symbols first. Consequently, symbols in the User Symbol Table have precedence over symbols in the remaining symbol tables with the same name. To reference a symbol in another symbol table which has the same name as a symbol you have defined, you have to first UNDEFINE the user defined symbol.

Note: If you change regions, the symbol in the User Symbol Table still takes precedence over a symbol in another symbol table which has the same name. If this occurs, the user will be informed that there's a multiply defined symbol. The user then has the option of UNDEFINing that symbol from the User Symbol Table.

Format:

DEFINE [/qualifier] symbol = data

Command Qualifiers

/ASCII

/DECIMAL

/HEXADECIMAL

/OCTAL

/RAD50

/SYMBOL

Command Parameters:

symbol

Specifies the name of the symbol to be defined. The following rules must be followed when defining a user symbol.

- Symbols can be composed of alphanumeric characters (A-Z and 0-9), dollar signs ($) or periods (.).

- The first character must not be numeric (0-9).

- The symbol cannot exceed 6 characters in length.

data

Specifies the data to use for the symbol definition. The data is interpreted according to the qualifier specified, ie. ASCII, DECIMAL, HEXADECIMAL, OCTAL, RAD50, SYMBOL. If no type or length qualifier is specified the current input setting, specified on the last SET TYPE command, is used. (See the SET TYPE command for further details.)

Command Qualifiers:

/ASCII

Specifies that the data is interpreted as ASCII characters. If two ASCII characters were not specified, the remaining portion of the word is blank filled.

/DECIMAL

Specifies that the data is to be interpreted as an unsigned decimal number.

/HEXADECIMAL

Specifies that the data is to be interpreted as hexadecimal digits.

/OCTAL

Specifies that the data is interpreted as an unsigned octal value.

/RAD50

Specifies that the data is to be interpreted as RAD50 characters. Any time a RAD50 string doesn't fill up a word (three RAD50 characters can fit in a word), the remainder of the word will be blank filled.

/SYMBOL

Specifies that the data is to be interpreted as a symbolic expression. A symbolic expression can be a one or two term expression, separated by a plus (+) or a minus(-) sign. A term can be either a symbol or an octal constant. The value of the symbol is used in defining the user symbol.

Example:

```
DEFINE TEMP = 10
```

9.7 DEPOSIT

Description:

The DEPOSIT command allows you to modify the contents of memory locations in a running server image. If you enter more than one value, the values are deposited at sequential memory locations starting with the specified location. Deposits into sequential memory locations occur as an atomic (ie. uninterruptable) operation.

Data values to be deposited can be specified in numeric, ASCII or RAD50 string, or symbol format.

The DEPOSIT feature is not intended to be used as a patching facility. It should be used only to assist in diagnosing problems. Any code inserted should only be temporary. (A scratch data area exists in the Server Base for depositing debug code. Starting at location $PATCH, there are 100 octal bytes available.) To use this command the /DEPOSIT switch must be specified on the SET IMAGE command.

The default mode for deposits is "protected". This requires the users to first EXAMINE a location, or range of locations, before attempting a DEPOSIT. This supports the presumption that the user really only wishes to change the content of a memory location to a specific value based on his knowledge of its current content. If the running server system changes the value before the deposit can be performed, the above will prevent the deposit from taking place and the user is informed.

In the case mentioned above, where the server system changes the values of locations before a deposit can be made, an "unprotected" mode of operation can be enabled. This is done by including the /UNPROTECT qualifier to the DEPOSIT command. In this case RED... will perform the deposit without any validation of current content. (NOTE: To use this capability the /UNPRODEP switch must also have been specified on the SET IMAGE command.)

(NOTE: The DEPOSIT command can be abbreviated to D.)

Format:

DEPOSIT [/qualifier(s)] expression = data [,data...]

Command Qualifiers

/ASCII

/BYTE

/DECIMAL

/HEXADECIMAL

/OCTAL

/RAD50

/SYMBOL

/UNPROTECT

/WORD

Command Parameters:

expression

Specifies the location in which to deposit the data. This location may be specified as either an octal or symbolic expression. Such an expression is comprised of one or two terms, separated by a "+" or "-". A term may be an octal number or a symbol.

data

Specifies the data to be deposited. The data is interpreted according to the qualifier(s) specified. Qualifiers are of two categories. They are either type qualifiers (ASCII, DECIMAL, HEXADECIMAL, OCTAL, RAD50, SYMBOL) or length qualifiers (BYTE, WORD). If no qualifiers are specified the default is the current settings for "TYPE" and "LENGTH". (See the SET LENGTH and SET TYPE commands for further details.)

Command Qualifiers:

/ASCII

Specifies that the data is ASCII characters. When the length specified is word and there aren't an even number of ASCII characters (two ASCII characters can fit in each word) in the string, the remaining portion of the last word will be blank filled.

/BYTE

Specifies that the unit of data being deposited is in bytes. For each data item specified the debugger deposits one byte of data. An error will occur when a value is entered that exceeds what can be stored in a byte (a value greater than 255, unsigned).

/DECIMAL

Specified that all data is to be interpreted as unsigned decimal numbers.

/HEXADECIMAL

Specified that all data following is to be interpreted as hexadecimal digits.

/OCTAL

Specifies that the data will be interpreted as unsigned octal values.

/RAD50

Specifies that the data is to be interpreted as RAD50 characters. The smallest unit that a RAD50 string can be deposited into is a word. Therefore, if the length has been specified as a byte, either by using a qualifier or by the SET LENGTH command, an error message will be displayed. Also, any time a RAD50 string doesn't fill up a word (three RAD50 characters can fit in a word), the remainder of the word will be blank filled.

/SYMBOL

Specifies that the data is to be interpreted as a symbolic expression. A symbolic expression can be a one or two term expression, separated by a plus (+) or a minus(-) sign. A term can be either a symbol or an octal constant. The value of the symbol is used in depositing into the location specified.

/UNPROTECT

Specifies that the current deposit operation is to be performed without checking the current contents (ie. there is no protection against inadvertant mistakes). This requires that the last SET IMAGE command included a "/UNPRODEP" qualifier.

/WORD

Specifies that the unit of data being deposited is a word. For each data item specified the debugger deposits two bytes of data. If a value is specified which exceeds what can be stored in a word (greater than 65,535, unsigned), an error will occur.

Example:

```
DEPOSIT/BYTE/ASCII SYSPAS + 1 = T,E,S,T
```

9.8 DISPLAY

Description:

The DISPLAY command can be used following an EXAMINE command to redisplay the contents of a specified address or range of addresses which where specified in the last EXAMINE command.
The DISPLAY command does not cause the target image to be accessed. The contents of the addresses obtained from the last EXAMINE command can be redisplayed many times over with the DISPLAY command in numeric form, ASCII or RAD50 string form, as symbols where possible.

An error message is displayed if the user attempts to execute a DISPLAY command before executing an EXAMINE command.

Format:

```
DISPLAY [/qualifier(s)]

Command Qualifiers

------------------

/ASCII

/BYTE

/DECIMAL

/HEXADECIMAL

/OCTAL

/RAD50

/STRUCTURE

/SYMBOL

/WORD
```

Command Qualifiers:

/ASCII

Specifies that the contents of the memory location(s) should be displayed as ASCII characters. (Note: Control characters are displayed as spaces.)

/BYTE

Specifies that the data should be displayed in byte entities.

The format of how this data is displayed will be determined by either a type qualifier, if specified, or what was specified in the SET TYPE command. The initial default is octal. (See the SET TYPE command for further information.)

/DECIMAL

Specifies that all data should be displayed in decimal as signed numbers.

/HEXADECIMAL

Specifies that all data should be displayed as hexadecimal digits.

/OCTAL

Specifies that the data will be displayed as octal digits.

/RAD50

Specifies that the data is to be displayed as RAD50 characters. It must be displaying a field with a minimum length of a single word and the addresses specified must start on an even byte boundary. Otherwise an error will occur.

/STRUCTURE

Specifies that a tailored output format will be used for displaying the data. The format has been predefined by a structure definition. The output format will be the same as what was specified for the previous EXAMINE command. If the previous EXAMINE command did not specify a structure, an error message is displayed.

/SYMBOL

Specifies that the data is to be displayed symbolically, whenever possible. Otherwise, it will be displayed as an octal offset from a symbol. If this not possible, the default will be to output the value as an octal number.

/WORD

Specifies that the data should be displayed as word entities. The format of how this data is displayed will be determined by either a type qualifier, if specified, or what was specified in the SET TYPE command. The initial default is octal. (See the SET TYPE command for further information.)

Example:

```
DISPLAY/BY/DEC/HEX
```

9.9 EXAMINE

Description:

The EXAMINE command displays the contents of a specified address or range of addresses. If an address is omitted the next sequential location in memory is examined. The contents of the addresses can be displayed in numeric form, ASCII or RAD50 string form, or as symbols where possible.

The contents may also be displayed as formatted output, When a data structure display format is specified, the data won't be displayed in any of the other display formats (see Appendix A for further details).

An examine of a range of addresses occurs as atomic (ie. uninterruptable) operation on a running server.

(NOTE: The EXAMINE can be abbreviated to E.)

Format:

EXAMINE [/qualifier(s)] [expression[:expression]]

Command Qualifiers

/ASCII

/BIAS

/BYTE

/DECIMAL

/HEXADECIMAL

/OCTAL

/RAD50

/STRUCTURE = structure-name

/SYMBOL

/WORD

Command Parameters:

expression

Specifies the location to be examined. This is an optional parameter, which if specified can be either an octal or a symbolic

expression. Such an expression is comprised of one or two terms, separated by a "+" or "-". A term may be an octal number or a symbol.

If an expression is not specified the location is incremented by the appropriate amount and the next location is examined.

If a range is specified, (expression:expression) the first address must be less than the second address, whether they are specified symbolically or as an octal numbers. Otherwise, an error message will be displayed.

Command Qualifiers:

/ASCII

Specifies that the contents of the memory location(s) should be displayed as ASCII characters. (Note: Control characters are displayed as spaces.)

/BIAS

Indicates that the addresses are to be translated into 22-bit physical addresses, using the bias value specified in a preceeding BIAS command. (See the BIAS command for further details.)

/BYTE

Specifies that the data should be displayed in byte entities. The format of how this data is displayed will be determined by either a type qualifier, if specified, or what was specified in the SET TYPE command. The initial default is octal. (See the SET TYPE command for further information.)

/DECIMAL

Specifies that all data should be displayed in decimal as signed numbers.

/HEXADECIMAL

Specifies that all data should be displayed as hexadecimal digits.

/OCTAL

Specifies that the data will be displayed as octal digits.

/RAD50

Specifies that the data is to be displayed as RAD50 characters. It must be displaying a field with a minimum length of a single word and the addresses specified must start on an even byte boundary. Otherwise an error message will be displayed.

/STRUCTURE = structure-name

Specifies the name of a structure definition to be used in examining locations in memory. Implicit in the structure definition is the range of memory locations to be examined. You only need to specify the starting location to be examined. If a range is specified the second expression will be ignored. (See Appendix A for further details.)

/SYMBOL

Specifies that the data is to be displayed symbolically, whenever possible. Otherwise, it will be displayed as an octal offset from a symbol. If this not possible, the default will be to output the value as an octal number.

/WORD

Specifies that the data should be displayed as word entities. The format of how this data is displayed will be determined by either a type qualifier, if specified, or what was specified in the SET TYPE command. The initial default is octal. (See the SET TYPE command for further information.)

Examples:

```
EXA $LDBRT+10

EXAMINE/DEC/RAD 100:120

EX .

EXA $PARHD

E/STR=PCB @

EX
```

9.10 EXIT Or ^Z

Description:

Ends the debugging session and returns control to the operating system.

Format:

EXIT or ^Z

Command Parameters:

None.

Command Qualifiers:

None.

Example:

```
EXIT
```

9.11 HELP

Description:

The HELP command displays a list of available debugger commands. (NOTE: There is no intention to provide a full help capability in POD for security reasons.)

Format:

HELP

Command Parameters:

None.

Command Qualifiers:

None.

Example:

HELP

9.12 MAP

Description:

Specifies that a mapping into physical memory is to occur for any address(es) within the APR context specified. This mapping is performed transparently by POD. If it is determined that the virtual address specified falls within the APR(s) specified for mapping, the map value entered is used as a block number and the virtual address specified is converted to a 22-bit physical address using this block number.

If an APR is not specified as a switch on this command, the default is assumed to be APR 6.

(Note: Mapping can be specified for more than one APR at a time.)

Format:

MAP [/qualifier] map-value

Command Qualifiers

/APR = apr-number

Command Parameters:

map-value
An 16 bit octal value to be used as a bias into physical memory, for address(es) within the specified APR context. (Note: No check is performed to determine whether the map-value could cause the physical address(es) to go beyond the physical memory limit. If this occurs, an error message will be displayed when attempting to EXAMINE or DEPOSIT using mapping.)

Command Qualifiers:

/APR = apr-number
Indicates the APR for which mapping is to be used. The default if this switch is not specified is to use APR 6 for mapping.
Example:

MAP/APR=4 14030

9.13 MONITOR

Monitors a location or locations and indicates if the contents change during a specific time interval. (This is done by examining locatiion(s) periodically, not continuously.) The contents of all specified locations are displayed initially, based on the current display type setting. After this, any location and its contents are displayed only when changes are detected. This command is allowed against a server running system only. By default, POD will re-examine the locations, word/byte based on the current length setting, every second for a period of 30. seconds.

Command Format:

MONITOR [/qualifier(s)] expression [,expression...]

Command Qualifiers

/WORD

/BYTE

/TIME

/INTERVAL

Command Parameters:

expression

Specifies the location(s) to be monitored. A maximum of five locations are allowed. A location may be either an octal expression or a symbolic expression. Such expressions are comprised of one or two terms, separated by a "+" or "-". A term may be an octal number or a symbol.

Command Qualifiers:

/BYTE
Specifies that the location(s) to be monitored are in byte lengths.

/INTERVAL=seconds
Specifies the interval between re-examination of the monitored location(s). A value between 1 and the time specified in the time qualifier is allowed. The default is 1 second.

/TIME=seconds
Specifies the duration in which to monitor the location(s). A

value between 1 and 300. seconds is allowed. The default is 30. seconds.

/WORD

Specifies that the location(s) to be monitored are in word lengths.

Example:

```
MONITOR/INT=5/TIME=100 $ZTIME+2,$PATCH
```

9.14 NOBIAS

Description:

Cancels a previous BIAS command and indicates that no bias operations are to be performed.

Format:

NOBIAS

Command Parameters:

None.

Command Qualifiers:

None.

Example:

NOBIAS

9.15 NOMAP

Description:

Cancels a previous MAP command. Indicating that a mapping operation is not to be performed for the specified APR. If an APR is not specified as a switch on this command, the default is APR 6.

Format:

```
NOMAP [/qualifier]
```

Command Qualifiers

```
/APR = apr-number
```

Command Parameters:

None.

Command Qualifiers:

/APR = apr-number
Indicates the APR for which mapping is to be cancelled. The default if this switch is not specified is to cancel APR 6 mapping.

Example:

```
NOMAP/APR=4
```

9.16 SET

Description:

Enables the user to set the server context within which examines are to take place, by specifying the generic type of server product, the specific server image and the region within this image. The user can also indicate how to display addresses, where to output any information from a debugging sessions, the name of a log file for output, how to display the examined data.

See the individual command descriptions following for more information.

(NOTE: The SET command can be abbreviated to S.)

Format:

SET keyword [/qualifier] parameter

Command Keywords:

ADDRESS, IMAGE, LENGTH, LOG, OUTPUT, REGION, SERVER or TYPE

Command Parameters:

Depends on keyword specified.

Command Qualifiers:

Depends on keyword specified.

9.16.1 SET ADDRESS -

Description:

Sets the display type for addresses which are being examined. The address types control whether the addresses are displayed in octal and/or symbolic form.

The default display of addresses is in symbolic form. The parameter specified in the SET ADDRESS command will override the current defaults. The addresses may be displayed in octal form, symbolic form, or both.

Format:

SET ADDRESS address-keyword [,address-keyword]

Commamnd Parameters:

address-keyword

This can be OCTAL or SYMBOLIC.

Command Qualifiers:

None.

Example:

```
SET ADDRESS OCTAL,SYMBOLIC
```

9.16.2 SET IMAGE -

Description:

The SET IMAGE command specifies the target image upon which POD operations are to be applied. Such an image may be a running server node, a server product distribution image, or a crash dump of a server node. An appropriate keyword must be applied to identify the node name or filename of the image.

The qualifiers of the SET IMAGE command specify how the target image can be accessed. They are only valid when the target image is a running server node. They specify whether the user can make use of the BREAK command and what deposit types (protected or unprotected) are allowed. See the descriptions of the BREAK and DEPOSIT commands for further details.

Format:

SET IMAGE [/qualifier(s)] image-keyword [node-qualifiers]

Command Parameters:

image-keyword

Specifies the target image upon which POD operations are to be performed. Valid image-keywords are NODE, DUMP, or DISTRIBUTION and the syntax associated with each is described below.

NODE=nodename

Specifies the name of a running server node to be examined. A valid nodename can consists of up to six alpha or numeric characters and the first character cannot be a number.

Node-qualifiers are optional qualifiers to allow access control information to be specified.

USER = username
PASSWORD = password

DUMP=file-specification

Specifies the name of a dump image file. The default file type is .CDA.

Filespecs must be entered as quoted strings, unless the user is just specifying a filename and type.

DISTRIBUTION=file-specification

Specifies the name of a distribution image for a server product. The default file type is .SYS.

Filespecs must be entered as quoted strings, unless the user is just specifying a filename and type.

Command Qualifiers:

NOTE:
These qualifiers are only valid when the target image is a running server node.

/DEPOSIT
Specifies that the user can do protected deposits, which requires the user to first examine the location(s) before making a deposit to the location(s).

/UNPRODEP
Specifies that the user can perform an unprotected deposit, which means that a deposit can be made without any validation of the data at the location(s) specified. (Note: If you specify this switch, you are automatically given the ability to do protected deposits, as well.)

/BREAK
Specifies that the user has can initiate a BREAK command which will cause a running server system to crash.

Examples:

```
SET IMAGE/DEP NODE=BOOJUM USER=SYSTEM PASSWORD=PRIV

SET IMAGE DUMP = "LB:[40,6]EOMER.CDA"

SET IMAGE DIST = RSX11S.SYS
```

9.16.3 SET LENGTH -

Description:

Sets the default length for commands such as DEPOSIT and EXAMINE. The initial length is WORD. A qualifier may be specified in the DEPOSIT or EXAMINE command, to override the setting for one command.

Format:

SET LENGTH length-keyword

Command Parameters:

length-keyword

Specifies the default length. The length-keyword can either be BYTE or WORD.

BYTE
Specifies that the default length is byte.

WORD
Specifies that the default length is word.

Command Qualifiers:

None.

Example:

```
SET LENGTH BYTE
```

9.16.4 SET LOG -

Description:

Specifies the file specification of the log file that the debugger uses when the output is directed to a log file.

The SET LOG command only controls the name of the log file. It does not control whether to output information to a log file. This is done by the command SET OUTPUT.

If no log file name is specified, the debugger will default to a file specification of POD.LOG. If the debugger is currently creating a log file and you enter SET LOG, the debugger does not closes the existing file. You must turn logging off and back on again to cause the debugger to close the file you are currently logging to (see the SET OUTPUT command for further details). If a file name is specified and that file already exists, a new version will be generated.

Format:

SET LOG file-specification

Command Parameters:

file-specification

Specifies the name for the log file. The file specification must be in the appropriate format for the host operating system that POD is running under. The default file specification is POD.LOG.
Filespecs must be entered as quoted strings, unless the user is just specifying a filename and type.

Command Qualifiers:

None.

Example:

SET LOG KIM.TST

9.16.5 SET OUTPUT -

Description:

Controls where the debugger sends its output: to the terminal, a log file, or both.

When the debugger is initiated, all debugger responses are displayed at the terminal and no log file is created.

Format:

```
SET OUTPUT option [,option...]
```

Command Parameters:

option

Specifies the output mode. Valid options are LOG, TERMINAL, or their negatives, NOLOG or NOTERMINAL.

LOG

Enables output to a log file. The log file contains all the commands that are entered at the terminal and all debugger responses. NOLOG, which is the default setting, inhibits output to a log file.

TERMINAL

Enables output to the terminal. This is the default option. NOTERMINAL causes the debugger to stop displaying commands and responses to the terminal.

Command Qualifiers:

None.

Example:

```
SET OUTPUT LOG
```

9.16.6 SET REGION -

Description:

The SET REGION command establishes an address space context within the current image. This may be the 22-bit physical memory space or some 16-bit virtual address space, such as for a specific task.

The Region Symbol Table is reloaded when setting a region. POD will load one or more .STB files, symbol information, from the current .SYM file that is opened.

Format:

SET REGION region-name

Command Parameters:

region-name

Supported regions are "PHYSICAL" (22-bit address space), "EXECUTIVE" (virtual address space of RSX executive, Comm Exec and I/O page) or "name", where "name" is an existing task, common, driver, dynamic region, or DECnet process. "Name" must have an associated Partition Control Block in RSX-11S DSR (ie. task name, common name, or device name).

Setting a region to "PHYSICAL" will result in an empty Region Symbol Table. Setting a region to "EXECUTIVE" will result in the loading of CEX.STB (the Executive and Communications Executive symbol information) into the Region Symbol Table and the enabling of virtual address references corresponding to APR's 0-4 and 7.

When a region name is specified whose base APR is 5 or 6 (eg. for a privileged task or region), POD also allows the user access to the "EXECUTIVE" address space and corresponding symbols transparently (APR's 0-4 and 7).

Command Qualifiers:

None.

Example:

SET REGION NT.XPT

9.16.7 SET SERVER -

Description:

This command establishes a context to a specific server product, and hence names a .DSD and .SYM file for this context. Different node and file images can be set to within this context.

Format:

SET SERVER server-type

Command Parameters:

server-type

This can be ROUTER, TERMINAL, SNA, X25.

ROUTER
Refers to the Router Server product.

TERMINAL
Refers to the Terminal Server product.

SNA
Refers to the SNA Gateway.

X25
Refers to the X25 Gateway and Router product.

Command Qualifiers:

None.

Example:

SET SERVER ROUTER

9.16.8 SET TYPE -

Description:

Sets the default entry type and display types. The entry type and display types control whether the data is interpreted in a radix form, as ASCII or RAD50 strings, symbolically or in the case of display only, as a data structure.

More than one type keyword can be specified for displaying the data and the data will be output in the various forms. However, when one of the display types is STRUCTURE, the data will only be displayed as a data structure. A warning message will be issued if the user attempts to set type to structure and something else.

In the case of the entry type (ie. how to interpret input data) only one form is allowed. If more than one type keyword has been specified, the first keyword will be interpreted as the default entry mode.

You can override the current type by using a qualifier with commands such as DEPOSIT, DISPLAY or EXAMINE. If the type has not been set, using either the SET TYPE command or a qualifier, the default is octal, for both input data and display data.

If the type has been set to SYMBOL, values are displayed as symbols or offsets from symbolic locations. The debugger first looks for an exact match with a symbol. It will first search the User Symbol Table, then the Region Symbol Table, and if necessary the System Symbol Table. If no exact match can be found, the debugger searches for the symbol closest to the value, but not further away than a 100(octal). If it cannot find any symbolic definition, it displays the value in octal.

Format:

SET TYPE type-keyword [,type-keyword]

Command Parameters:

type-keyword

Specifies the entry and display type(s). The type-keyword(s) can be ASCII, DECIMAL, HEXADECIMAL, OCTAL, RAD50, STRUCTURE or SYMBOL. Whenever one of these display types is invalid for interpreting the data, the data will be interpreted in octal.

ASCII

Specifies that the data is to be interpreted as ASCII characters. (Note: Control characters are displayed as spaces.)

DECIMAL

Sets the current radix to decimal.

HEXADECIMAL
Sets the current radix to hexadecimal.

OCTAL
Sets the current radix to octal.

RAD50
Specifies that the data is to be interpreted as RAD50 characters.

STRUCTURE = structure-name
Specifies that the data is to be displayed as a data structure and structure-name indicates which data structure definition is to be used for displaying data. Structure definitions have been predefined and are included when the server context is specified (see Appendix A for further details.)

(Note: This is not a valid entry form, the input type will default to octal.)

SYMBOL
Specifies that the data is to interpreted symbolically.

Command Qualifiers:

None.

Examples:

```
SET TYPE DEC,HEX,SYM

SET TYPE STRUCT=RNN
```

9.17 SHOW

Description:

Allows the user to look at what type of server product is being examined, the specific image being looked at, and the region within this image that the user is set to. The user may also show where the output information from the debugging session is being placed, the name of a log file for output, the type being used to interpret the entry and display of data, and the length for handling the data. Also, the user can display the octal value associated with a symbol or symbols.

See the individual command descriptions following for more information.

Format:

SHOW keyword [/qualifier] parameter

Command Keywords:

IMAGE, LENGTH, LOG, OUTPUT, REGION, SERVER, SYMBOL or TYPE

Command Parameters:

Depends on keyword specified.

Command Qualifiers:

Depends on keyword specified.

9.17.1 SHOW IMAGE -

Description:

The SHOW IMAGE command displays the current image context, including the identifying nodename or filename.

Format:

```
SHOW IMAGE
```

Command Parameters:

None.

Command Qualifiers:

None.

Example:

```
SHOW IMAGE
```

9.17.2 SHOW LENGTH -

Description:

Shows the default length (ie. words or bytes) being used by the DEPOSIT, EXAMINE and DISPLAY commands.

Format:

SHOW LENGTH

Command Parameters:

None.

Command Qualifiers:

None.

Example:

SHOW LENGTH

9.17.3 SHOW LOG -

Description:

Displays the name of the log file. Format:

SHOW LOG

Command Parameters:

None.

Command Qualifiers:

None.

Example:

SHOW LOG

9.17.4 SHOW OUTPUT -

Description:

Shows where the output is being sent.

Format:

SHOW OUTPUT

Command Parameters:

None.

Command Qualifiers:

None.

Example:

SHOW OUTPUT

9.17.5 SHOW REGION -

The SHOW REGION command displays the current region context, including the virtual address space and the number of symbols that have been loaded into the Region Symbol Table.

Format:

SHOW REGION [qualifier(s)]

Command Parameters:

None.

Command Qualifiers:

/ALL
In addition to the current region context, a list of all known regions for the current image is displayed

Examples:

SHOW REGION

SHOW REG/ALL

9.17.6 SHOW SERVER -

Description:

Displays the current server product type being accessed, the name of the .SYM file, from which the symbols are loaded, and the number of symbols in the System System Table.

Format:

SHOW SERVER

Command Parameters:

None.

Command qualifiers:

None.

Example:

SHOW SERVER

9.17.7 SHOW STRUCTURE -

Description:

Displays the names of structures which have been defined for formatting output. (see Appendix A for further details).

Format:

SHOW STRUCTURE

Command Parameters:

None.

Command qualifiers:

None.

Example:

```
SHOW STRUCTURE
```

9.17.8 SHOW SYMBOL -

Description:

The SHOW SYMBOL command is used to display, in octal, the value of a symbol or symbols, if defined. Wildcard characters can be used in the symbol specification. The wildcard character, "%", will match on any single character. The wildcard character, "*", will match on any string, including null strings.

Format:

```
SHOW SYMBOL [/qualifier] symbol-name
```

Command Parameters:

symbol-name
Specifies a string whose value from the User, Region, and/or System Symbol Tables will be displayed, depending on the qualifier used. If no qualifier is specified, all three symbol tables will be searched for matching symbol names. The symbol-name specified may include wildcard characters.

Command Qualifiers:

One and only one, of the following qualifiers can be used. If no qualifier is specified, the default will be to reference all three symbol tables.

/USER
Indicates that the symbol table to be referenced will be limited to the USER Symbol Table

/REGION
Indicates that the symbol table to be referenced will be limited to the REGION Symbol Table

/SYSTEM
Indicates that the symbol table to be referenced will be limitd to the SYSTEM Symbol Table

Example:

```
SHOW SYMBOL $XBIAS

SHOW SYMBOL/SYSTEM $TKPS

SHO SYM $T*

SHO SYM %.*
```

9.17.9 SHOW TYPE -

Description:

Displays the current types for interpreting and displaying the data.

Format:

SHOW TYPE

Command Parameters:

None.

Command Qualifiers:

None.

Example:

```
SHOW TYPE
```

9.18 UNDEFINE

Description:

The UNDEFINE command clears the definition of a user defined symbol.

Format:

UNDEFINE symbol

Command Parameters:

symbol

Specifies the name of the symbol to be cleared.

Command Qualifiers:

None.

Example:

```
UNDEFINE TEMP
```

APPENDIX A

DATA STRUCTURE DEFINITIONS

POD Data Structures for the Server Base Executive

The following data structures have been defined for the Server Base executive (RSX-11S v4.0). For each data structure, its name, associated pointers, source macro name (from EXEMC.MLB) and list header or other access information is provided.

CRASH

The crash macro is used to print out the contents of the machine registers that were saved when the system crashed. Only those registers that correspond to the PLUTO hardware are printed out (for example, no Supervisor Mode registers or D Space APRs are shown).

Access to the crash registers is via the exec location $CRSST, viz:

```
POD>examine/structure=crash $CRSST
```

There are no structures referenced from structure CRASH.

There is no corresponding EXEMC.MLB macro name.

Example:

```
        Crash Registers
Before Crash
 PC =    016070
 PS =    000010
After Crash
  PS   =    000340
 SP(K) =    123036
 SP(U) =    000410
Registers
 R0 =    177434
```

```
 R1 =     177777
 R2 =     177777
 R3 =     000000
 R4 =     000000
 R5 =     000000
Memory Management Registers
 MMR0 =     000017
 MMR1 =     000000
 MMR2 =     024356
 MMR3 =     000020
User I Space
   PDRs
 APR 0 =     077406
 APR 1 =     077406
 APR 2 =     077406
 APR 3 =     077406
 APR 4 =     077506
 APR 5 =     077406
 APR 6 =     077406
 APR 7 =     077506
   PARs
 APR 0 =     000000
 APR 1 =     000200
 APR 2 =     000400
 APR 3 =     000600
 APR 4 =     000577
 APR 5 =     000600
 APR 6 =     001400
 APR 7 =     177600
Kernel I Space
   PDRs
 APR 0 =     077506
 APR 1 =     077506
 APR 2 =     077406
 APR 3 =     077406
 APR 4 =     077406
 APR 5 =     077506
 APR 6 =     077406
 APR 7 =     077506
   PARs
 APR 0 =     000000
 APR 1 =     000200
 APR 2 =     000400
 APR 3 =     000600
 APR 4 =     001000
 APR 5 =     017710
 APR 6 =     001400
 APR 7 =     177600
```

CLKQ

The clock queue macro is used to print Clock Queue Control Blocks. The last 3 words displayed (Type-Dependent Area) have different interpretations depending on the value of C.RQT (clock request type).

Access to the head of the list of clock queue entries is at $CLKHD,

```
POD>examine $CLKHD
POD>examine/structure=clkq @
```

Chaining down the list of clock queue entries is done by

```
POD>examine
     or
POD>chain
```

If the clock queue entry relates to a task, the corresponding TCB can be accessed by

```
POD>chain C.TCB
```

The EXEMC macro is CLKDF$.

Example:

```
        Clock Queue Control Block
C.LNK =       $INITL+000026
C.RQT =    Internal Routine-Ident
C.EFN =         0.
C.TCB =       062714
C.TIM =         31005400000
Type-Dependent Area =    122040     015040     101054
```

DCB

The DCB structure describes Device Control Blocks. The head of the list is at $DEVHD.

```
POD>examine $DEVHD
POD>examine/structure=DCB @
```

Chaining down the list of DCBs is done by

```
POD>examine
     or
POD>chain
```

The list of UCBs corresponding to the DCB can be accessed via D.UCB,

```
POD>chain D.UCB
```

The PCB for the driver (if any) is accessed via D.PCB,

```
POD>chain D.PCB
```

The EXEMC macro is DCBDF$.

Example:

```
Device Control Block
D.LNK =        063004
First UCB =        064236
Device Name =     NS
Unit numbers (low,high) =     000      000
UCB length =     000040
D.DSP =     122232
Legal, Control, NOP, ACP - (0-15) =     000001      000001      000000      000000
Legal, Control, NOP, ACP - (16-31) =     037000      020000      000000      017000
Driver PCB =     064410
```

TSKHDR

The TSKHDR structure describes the task header. The task header is accessed from P.HDR in the corresponding PCB. The structure display always shows entries for LUNs 1 to 10, even though not all LUNs may be in use.

The various AST blocks can be accessed from the task header, viz:

```
POD>chain H.PFVA
POD>chain H.FPVA
POD>chain H.RCVA
POD>chain H.RRVA
```

The Window Blocks for the task may be accessed via H.WND - H.WND points to a word containing the number of window blocks, followed immediately by the window blocks themselves.

The EXEMC macro is HDRDF$.

Example:

```
        Task Header
Current Stack Pointer =    000000
Header Length =    000206
Event Flag mask,addr =    107117     131722
Current UIC =    001     001
Default UIC =    001     001
Initial PS =    170017
Initial PC =    124572
Initial SP =    121270
ODT SST vector length,addr =    000000     000000
Task SST vector length,addr =    000000     000000
Powerfail AST block =    000000
FP AST block =    000000
Receive AST block =    000000
Event flag save =    000000
FP save =    140206
Window Blocks =    056156
Directive Status =    000000
FCS impure ptr =    000000
Fortran impure ptr =    000000
Overlay impure ptr =    000000
Work Area ptr =    000000
Swapping Priority =         0.
Mailbox LUN =    000
Receive by reference AST block =    000000
X.25 byte =    000
Guard word =    056220
Number of LUNs =    000011
LUN table
 LUN 1 =    064612     000000
 LUN 2 =    064612     000000
```

```
LUN 3 =     064612      000000
LUN 4 =     064612      000000
LUN 5 =     064612      000000
LUN 6 =     064612      000000
LUN 7 =     064612      000000
LUN 10 =     064612      000000
```

HDRWND

The HDRWND structure describes the Window Blocks in the Task Header for a task. The Window Blocks are accessed via H.WND in the Task Header.

Access to the next (adjacent) Window Block is available by:

```
POD>examine
     or
POD>chain
```

but the user must know when to stop, by the count in the word preceeding the first Window Block (and pointed to by H.WND)

The PCB corresponding to the task can be accessed via W.BPCB,

```
POD>chain W.BPCB
```

The attachment descriptor corresponding to the Window Block can be accessed via

```
POD>chain W.BATT
```

The EXEMC macro is HDRDF$.

Example:

```
Window Block
PCB address =    106720
Low VA =    120000
High VA =    142077
Attachment Descriptor =    106704
Blocks in window =    000221
Physical Block Address =    000000
First PDR =    001212
Number of PDRs =    003002
Last PDR value =    010006
```

ADB

The ADB structure describes Attachment Descriptor Blocks. The ADBs are accessed via W.BATT in the corresponding Window Block.

Successive ADBs that correspond to the same Partition (PCB) can be obtained by

```
POD>examine
     or
POD>chain
```

Successive ADBs that correspond to the same Task (TCB) can be obtained by following down the TCB attachment thread, subtracting 6 from the address contained therein.

The TCB corresponding to the task can be accessed via A.TCB,

```
POD>chain A.TCB
```

The PCB for the task is accessed via A.PCB,

```
POD>chain A.PCB
```

The EXEMC macro is PCBDF$

Example:

```
        Attachment Descriptor
PCB Attachment thread =    000000
Task Priority =       205.
I/O count =         0.
TCB of attached task =    107130
TCB Attachment thread =    000000
        Delete Access
        Write Access
        Read Access
Mapping Count =         1.
PCB of attached task =    106720
```

PCB

The PCB structure describes Partition Control Blocks. The head of the list is at $PARHD.

```
POD>examine $PARHD
POD>examine/structure=PCB @
```

Chaining down the list of PCBs is done by

```
POD>examine
     or
POD>chain
```

The main partition or next sub-partition can be accessed via P.MAIN or P.SUB, viz:

```
POD>chain P.MAIN
     or
POD>chain P.SUB
```

The Wait Queue is chained through P.WAIT,

```
POD>chain P.WAIT
```

The TCB for the task is accessed via P.TCB,

```
POD>chain P.TCB
```

The Task Header is accessed via P.HDR,

```
POD>chain P.HDR
```

The listhead for the Attachment Descriptors for the partition is accessed via P.ATT,

```
POD>chain P.ATT
```

The EXEMC macro is PCBDF$.

Example:

```
        Partition Control Block
P.LNK =    000000
Priority =       205.
IO + IOSB count =          0.
Partition Name =    GEN
P.SUB =    110304
P.MAIN =    111230
P.REL =    007245
P.SIZE =    000045
```

```
Wait Queue =     000000      000000
Swap Size =     000000
Busy Flags =     000      000
P.TCB =     111114
Starting APR =     0
         System controlled
P.HDR =     110740
Protection word = 000000
Attachment Descriptor Listhead = 110660 110320
```

AST

The AST structure describes AST Blocks. AST blocks are referenced from a number of places, including the Task Header (see TSKHDR structure).

AST blocks may be queued up, so the list may be traversed by

```
POD>examine
     or
POD>chain
```

The EXEMC macro is PKTDF$.

OCB

The OCB structure describes Offspring Control Blocks. OCBs are accessed via the TCB and field T.OCBH.

Chaining down the list of OCBs is done by

```
POD>examine
     or
POD>chain
```

The TCB for the parent task is accessed via O.PTCB,

```
POD>chain O.PTCB
```

The EXEMC macro is PKTDF$.

IORB

The IORB structure describes I/O Packets. IORBs are referenced, for example, from the SCB field S.PKT. The I/O parameters are displayed in both octal and decimal, for convenience.

Chaining down the list of IORBs is done by

```
POD>examine
     or
POD>chain
```

The TCB originating the I/O request can be accessed via I.TCB,

```
POD>chain I.TCB
```

The UCB for the unit on which the request is queued can be accessed via I.UCB,

```
POD>chain I.UCB
```

The EXEMC macro is PKTDF$.

Example:

```
I/O Packet
I.LNK =    000000
Priority =        36.
EFN =         0.
TCB of requestor =    000020
2nd LUN word =    123066
UCB addr =    000001
Function code =    121202
IOSB VA, bias, offset =    121202     012773     140002
AST addr =    123066
reserved =    012774
Parameters (octal) =  140072   000371   000000   000000   000010   000000
Parameters (decimal) =  -16326.     249.       0.       0.       8.       0.
```

SRDB

The SRDB structure describes Send/Receive Data Blocks.

Chaining down the list of SRDBs is done by

```
POD>examine
     or
POD>chain
```

The Sender's Terminal UCB can be accessed via SRUCB,

```
POD>chain SRUCB
```

Example:

```
Send/Receive Data Block
Link =    000000
Sender Task Name =     ...VNP
Send Data = 030001  054374  103240  103240  000000  111242  000000  000000
000000  107130  100000  022000  050000
Sender's Terminal UCB =    030070
UIC of Sender =    001     001
```

SRRDB

The SRRDB structure describes Send/Receive by Reference Data Blocks.

Chaining down the list of SRRDBs is done by

```
POD>examine
     or
POD>chain
```

The TCB for Sender's Task is accessed via SRRTCB,

```
POD>chain SRRTCB
```

SCB

The SCB structure describes Status Control Blocks. Successive SCBs may be examined by

```
POD>examine
     or
POD>chain
```

but the user is expected to know when to stop.

The I/O queue listhead can be accessed by S.LHD,

```
POD>chain S.LHD
```

The current I/O packet is referenced by S.PKT,

```
POD>chain S.PKT
```

The EXEMC macro is SCBDF$.

Example:

```
        Status Control Block
I/O queue =     000000      027624
Device priority =     240
Vector/4 =     014
Timeout - current, initial =          0.           5.
Controller =     000
Status =     000
CSR =     177560
Current I/O packet =     000000
Fork Block =     000000
Fork PC, R5, R4 =     000000      000000      000000
```

TCB

The TCB structure describes Task Control Blocks. The head of the list of installed tasks is at $TSKHD.

```
POD>examine $TSKHD
POD>examine/structure=TCB @
```

The head of the list of active tasks is at $ACTHD, and the pointer to the currently active task is in $TKTCB.

Chaining down the list of tasks in the System Task Directory (ie. installed tasks) is done by

```
POD>chain T.TCBL
```

Chaining down the list of active tasks is via T.ACTL,

```
POD>chain T.ACTL
```

The TCB has a number of pointers to other data structures, each of which can be accessed by the CHAIN command in POD:

CHAIN argument	Structure
T.CPCB	Checkpoint PCB
T.RCVL	Receive Queue Listhead
T.ASTL	AST Queue Listhead
T.UCB	UCB for TI:
T.LDV	UCB for the task load device
T.PCB	PCB for the task
T.RRFL	Receive by Reference Queue
T.OCBH	OCB Queue Listhead

The Attachment Descriptor blocks may be accessed using the pointer displayed, subtracting 6 from the address, as structure ADB.

The EXEMC macro is TCBDF$.

Example:

Task Control Block

```
T.LNK =     000000
Task Priority =         205.
I/O Pending =           0.
Checkpoint PCB =     000000
Task Name =     NTINIT
Receive queue =     000000      111126
AST queue =     000000      111132
Local Event Flags =           37777400001
TI: UCB =     030070
Task list thread =     107020
T.STAT =     -EXE
T.ST2 =     -CHK!FXD
T.ST3 =     -PMD!PRV
Default Priority =         205.
LBN for task load =     000      031      011
UCB for load device =     030140
PCB for task =     110674
Max task size =     000045
Next task in ATL =     107020
Specified AST list =     000000
Buffered I/O =           0.
Task size, blocks =     000045
Attachment descriptors =     110666      110666
Offset in partition =     000000
Send by reference count =           0.
Receive by reference queue =     000000      111206
OCB queue =     000000      111212
OCB count =           0.
Event Flag mask =     000001
Event Flag mask address =     111136
Group Global use count =     000000
```

UCB

The UCB structure describes Unit Control Blocks. UCBs are referenced from a number of places.

The DCB corresponding to the UCB can be accessed via U.DCB,

```
POD>chain U.DCB
```

The Redirect UCB can be accessed via U.RED,

```
POD>chain U.RED
```

The SCB can be accessed via U.SCB,

```
POD>chain U.SCB
```

The TCB for the attached task (if any) is accessed via U.ATT,

```
POD>chain U.ATT
```

The EXEMC macro is UCBDF$.

Example:

```
        Unit Control Block
DCB address =     027506
Redirect UCB =    027552
Control flags =    074
U.CTL =     UC.QUE!UC.PWF!UC.ATT!UC.KIL
Unit status =     001
U.STS-1 =
Unit =    000
Unit status extension =     000
U.ST2-1 =
Characteristics words =     000007      000010      000000      000110
U.CW1 =     DV.TTY!DV.CCL!DV.REC
SCB pointer =     027624
Attached task TCB =     000000
Current I/O request - bias, offset, count =     000000      000000      040001
```

POD Data Structures for the Server Base Communications Executive

The following data structures have been defined for the Server Base communications executive (DECnet-RSX/11S V4.0)
For each data structure, its name, associated pointers, source macro name (from NETLIB.MLB) and list header or other access information is provided.

DECHB

The DECHB structure is used to display the DECnet Home Block. $DECPT points to the start of the DECnet Home Block.

```
POD>examine $DECPT
POD>examine/structure=dechb @
```

The remote node name list can be accessed via D$RNN, but first you have to bias into extended pool.

```
POD>bias $xbias
POD>chain D$RNN
```

The NETLIB macro is DHBDF$.

Example:

```
DECnet Home Block Format
Alias Node Name Listhead =    000000
Remote Node Name Listhead =    101074
Pointer to End of Remote Node Name List =    004031
Local Node Name =    SBONE
Local Node Number
     Area: =    4.
     Node Number: =    161.
Local Node Identification =    Server Base BL8 PhIV End Node
Host Node Address
     Area: =    4.
     Node Number: =    19.
Hi-order 4 bytes of Ethernet Address =    $00AA     $0004
Number of Routing Channels =         2.
DLL Service Database Address =    000000
DLL Database FNB Address =    000000
ECL Segment Size =      1466.
```

FEATR

The FEATR structure is used to display the Feature Word for DECnet. It is the third characteristic word found in the UCB for NS:. (See the UCB description found in the Executive data structure section).
Once you've determined the address of the feature word, is the following command:

```
POD>examine/structure=featr x
```

The NETLIB macro is NSFDF$.

Example:

```
        Features Word for DECnet
Event Logging Support
Logical Link Integrity Support
System is an End Node
System Level Interface Support
```

RNN

The RNN structure describes the Remote Node Name List. The list can be accessed through the DECnet Home BLock via D$RNN, but you must be mapped into extended pool to chain throught the list.

Chaining down the remote node name list is done by

```
POD>examine
     or
POD>chain
```

The NETLIB macro is RNBDF$.

Example:

```
        Remote Node Name List
R.LNK =    004031
Node Name =    SBONE
Address
     Area: =    4.
     Node Number: =    161.
```

PDV

The PDV structure is used to display the Process Descriptors. $PDVTA points to the process descriptor vector table, which contains the PDV addresses.
You must determine the address you wish to examine before you can display a PDV.

```
POD>examine $PDVTA
$PDVTA/    x
POD>examine x:x+20
x/         a
x+2/       b
 .         .
 .         .
 .         .
x+20/      c
POD>examine/str=pdv a
```

After examining the PDV for XPT you can chain to the XPT Database, viz:

```
POD>chain xpt
```

If you examine the PDV for ECL, you can chain to the ECL Database, viz:

```
POD>chain ecldb
```

Chaining to the associated PCB is also possible by

```
POD>chain z.pcb
```

The NETLIB macro is PDVDF$.

Example:

```
Process Descriptor
Process =     AUX
Type =    LLC
Priority =    340
Flags =    000614
Z.FLG =    ZF.MTM!ZF.TIM!ZF.MFL
     Process Relocation Bias =    014265
     Dispatch Table Address =    120000
     Free Space Pointer =    000000
     PCB Address =    074650
     LLC Data Base Virtual Address =    074244
     No. of Channels =         0.
```

System Line Number = 325
Station Number = 30
Channel =

ECLDB

The ECLDB structure describes ECL's database. The database can be accessed through ECL's PDV via a chain operation specifying ECLDB. Or an examine can be done using the address specified as the "LLC Data Base Virtual Address".

From here you can chain to the mailbox queue or ECL's node counters.

Issue the following command to chain to the mailbox queue.

```
POD>chain n$mbxq
```

To look at ECL's node counters, you must first set up mapping through APR 6, using the value displayed in ECL's database as "ECL Node Counters APR Bias".

```
POD>map x
POD>chain n$enc
```

The NETLIB macro is ECDDB$.

Example:

```
ECL Data Base
ACP CCB Queue Listhead =    000000     074054
Timer Count =          0.
Flags Byte =    040
Flags =    NF$MOU
Function Code =    000002
Dummy VCB =    000021
Source Node Address
     Area: =    4.
     Node Number: =    10.
Round Trip Delay =        10.
Source Link Address =    024150
Destination Link Address =    044401
Error Code =    000051
Mapping of Current LLT =    014174
Current LLT Virtual Address =    034230
Current LLT Physical Address =    034230
Current Active Logical Links =          1.
Current Maximum Logical Links =          2.
Count of CI's Ignored Due to Resources =          0.
Logical Link Table Length =         40.
Logical Link Table Address =    030360
```

```
ECL Node Counters Listhead =     140050      140000
ECL Node Counters APR Bias =     014170
Mailbox Queue Listhead =     030220
General Delivery CCB Queue Listhead =     000000
General Delivery Current Timer =          4.
General Delivery Initial Timer =          5.
```

ECLNC

The ECLNC structure describes the ECL Node Counters. The node counters are accessed through ECL's database, a bias and address is displayed which can be used for exmaning the node counters or you can use a chain function described under ECLDB. Successive node counters may be examined by

```
POD>examine
     or
POD>chain
```

The NETLIB macro is CTRDF$.

Example:

```
       ECL Node Counters List
E$NLNK =     140120
Node Address
     Area: =     4.
     Node Number: =     10.
Number of Active Logical Links =          1.
Retransmit Period =         10.
Message Start Time =     000000
Segment Number of the Packet Being Timed =          0.
Logical Link Address of Segment =     000000
Number of Bytes Received =        156.           0.
Number of Bytes Sent =       341.           0.
Number of Messages Received =         66.           0.
Number of Messages Sent =         66.           0.
Number of Connects Received =          1.
Number of Connects Sent =          0.
Maximum Logical Links =          1.
Number of Response Time-outs =          0.
Number of Connects Ignored =          0.
Time this Counter Block Last Zeroed =     001250
```

NOTE

The counters are displayed (32 bit decimal values) as <low 16 bits.>, <high 16 bits.>. The actual number is [<high 16 bits.> * 32767.] + <low 16 bits.>.

MBX

The MBX structure describes Mailbox Queue. You can display successive entries by issuing the following:

```
POD>examine
     or
POD>chain
```

Issue the following command to chain to the TCB for the associated task.

```
POD>chain m.task
```

The NETLIB macro is MBXDF$.

Example:

```
        Mailbox Queue
M.NEXT =     000000
Pointer to Task TCB =     106160
Number of AST Entries =          0.
Number of Active Logical Links =          1.
Maximum Number of Logical Links =          1.
User Network Data AST Address =     122342
Network Data Listhead =     000000
Link Recovery Timer =          0.
```

LLT

Logical Link Tables are pointed to off of the ECL Database. ECL's database displays both a "Current LLT Virtual and Physical Address" which can be used for examining the current LLT. It also contains a "Logical Link Table Address and Length", which can be examined to determine the addresses of all LLTs.

Once you know the address you wish to examine, issue the following command:

```
POD>examine/str=llt x
```

Issue the following command to chain to the window block associated with this LLT,

```
POD>chain 1.wind
```

The NETLIB macro is LLTDF$.

Example:

```
Logical Link Table Entry
Link State =    ST$DAT - Normal Data Transfer
Link Type =    000
Logical Link Addresses - Local, Remote =    046001     024031
Remote Node 16 bit Address
     Area: =    4.
     Node Number: =    10.
No. of Transmits in Progress - I/LS, Data =         0.          0.
Link Flags =    200
L.FLAG =    LF.MMF
Remote NSP Version =         0.
Next Segment Numbers to be Assigned - Data, I/LS =        12.         17.
Next Data Segment Number to be Received =        13.
Highest Ack # From User on Data Channel =    12.
Next I/LS Segment Number to be Received =         4.
Highest I/LS Ack # from User =    3.
Last Segment Numbers Ack'd - Data, INT/LS =        11.         16.
User Disconnect Sub-state =    US$DON - Disconnect Complete
Network Disconnect Sub-state =    NS$DON - Disconnect Complete
Window Block =    033524
Transmit Count, Interrupt Count =         0.          1.
Flow Control Request Count (I/LS) =    0.
Flow Control Request Count (Data) =    0.
Remote Flow Control Count Estimate =         0.
I/LS Pending ACK Queue =    000000
Timer and Retry Cell (Data) =         0.          5.
```

```
Timer and Retry Cell (I/LS) =          0.          5.
Message Awaiting Retransmission =          0.
Long Term Timer =          0.
Timer (seconds) =          0.
Periodic Timer =          5.
Initial Long Term Timer =          0.
Message Re-assembly Queue =     000000
Size of Re-assembly Queue =          0.
Re-assembly Queue Timer =          0.
Pointer to ECL Counter Block =     140050
Segment Size for this Link =        561.
Disconnect Reason Code (In Octal) =     000400
Length of Optional Data (In Decimal) =          0.
Optional Data (In Octal) =     001      000      000      001      000      001      000
000
Source Channel Number =          1.
User Link Address =     000
Control Process PDV =     000
Data Process PDV =     000
Transmit Message Queue Listhead =     000000      034356
Current Transmit CCB Address =     000000
Interrupt Message Transmit Queue Listhead =     000000      034364
Current Interrupt CCB Address =     000000
Pending Control CCB Address =     000000
Pending Accept CCB Address =     000000
```

WBBLK

The WDBLK structure describes a Window Block.
You must first examine a LLT to determine the
window block address, it will be shown in
octal as "Window BLock" in the output of an LLT.

Issue the following command to chain
to the LLT associated with this window block,

```
POD>chain w.llt
```

And if you wish to examine the associated
mailbox, you can type

```
POD>chain w.mbox
```

The NETLIB macro is LLWDF$.

Example:

```
Window Block
Link Status =    000
W.STAT =
Task Lun =         2.
Pointer to Associated LLT =    034230
Segment Size =       561.
Temporary Workspace =    000000
Pointer to Mailbox =    030220
Kernal AST Status =    000
W.KAST =
```

XPT

The XPT structure describes XPT's database. The database can be accessed through ECL's PDV via a chain operation specifying XPT. Or an examine can be done using the address specified as the "LLC Data Base Virtual Address".

From here you can chain to the NI Cache, Transport Line Counters, or Transport Node Counters.
In all cases your region must have previously been set to NT.XPT.

The commands are as follows:

For NI Cache - . .

```
POD>chain niche
```

To look at XPT's line counters, you must first set up mapping through APR 6, using the value displayed in XPT's database as "Transport Line Counter's Bias" -

```
POD>map x
POD>chain xlncou
```

And for XPT's node counters -

```
POD>chain xndcou
```

The NETLIB macro is XPDDB$.

Example:

```
XPT Database
TCB Address of Level 1 Router =    000000
Level 1 Routing Message Listhead =    000000     073516
TCB Address of Level 2 Router =    000000
Level 2 Routing Message Listhead =    000000     073524
Level 1 Routing Timer =          0.
Level 2 Routing Timer =          0.
Physical Link Block Vector Size =          2.
Physical Link Block Vector Address =    072414
Reachability Vector Size =          0.
Reachability Vector - Bias,Address =    000000     000000
Area Reachability Vector Size =          0.
Area Reachability Vector - Bias,Address =    000000     000000
Minhop/Mincost Vector Size =          0.
```

```
Minhop/Mincost Vector - Bias,Address =     000000      000000
Area Minhop/Mincost Vector Size =          0.
Area Minhop/Mincost Vector - Bias,Address =     000000      000000
Hops/Cost Matrix in RCP - Bias,Address =     000000      000000
Area Hops/Cost Matrix in RCP -  Bias,Address =     000000      000000
NI Cache Size =         40.
NI Cache Address =     130406
Number of Password Database Entries =          0.
Password Database Address =     073634
Bias of Adjacency Database Part 1 =     120052
Bias of Adjacency Database Part 2 =     120056
Broadcast Router Priority Table Address =     030260
Number of Transport Line Counter Blocks =          1.
Transport Line Counter - Bias,Address =     014167      140000
Trace Control Word =     000000
Trace Collector TCB Address =     000000
Trace Control Block Address =     000000
Transport Node Counter Size =          4.
Transport Node Counter Address =     127240
```

NICHE

The NICHE structure displays NI Cache. The address to this location can be found by examining the XPT Database. You can also chain to it from this database. See XPT description for further details.

To follow down the entries, you can issue either of the following commands: (Note: your region must have been set to NT.XPT)

```
POD>examine
     or
POD>chain
```

The NETLIB macro is CACDF$.

Example:

```
     NI Cache
Node ID of Destination
     Area: =    4.
     Node Number: =    10.
Node Id of Next Hop =    $AA     $00     $04     $00     $11     $10
Lifetime Timer =        60.
```

XPTNC

The XPTNC structure displays the Transport node counters.
The address to this location can be found by examining the XPT Database. You can also chain to it from this database.
See XPT description for further details.

Once address has been determined, you can issue the command:

```
POD>examine/str=xptnc x
```

The NETLIB macro is CTRDF$.

Example:

```
        Transport Node Counters
Message Format Error Counter =          0.
Verification Reject Counter =          0.
Node Unreachable Packet Loss Counter =          0.
Aged Packet Loss Counter =          0.
Node Out of Range Packet Loss Counter =          0.
Oversized packet Loss Counter =          0.
Partial Routing Update Loss Counter =          0.
```

XPTLC

The XPTLC structure displays the Transport line counters.
The address to this location can be found by examining the XPT Database. You must first set up a mapping value in APR 6, this is displayed as part of the XPT database also.
You can also chain to it from the XPT database.
See XPT description for further details.

Once bias and address have been determined, you can issue the commands:

```
POD>map z
POD>examine/str=xptlc x
```

Chaining down the XPT line counters is done by

```
POD>examine
     or
POD>chain
```

The NETLIB macro is CTRDF$.

Example:

```
        Transport Line Counters
System Line Number Station Number =     000      000      042      004
Output Packets Received Counter =      1058.           0.
Input Packets Sent Counter =       1068.           0.
Line Down Count =          0.
Line Initialization Failure =          0.
Transit Packets Received Counter =           0.           0.
Transit Packets Sent Counter =           0.           0.
Transit Congestion Loss Count =           0.
Time Since Counters Last Zeroed =           0.
Initial Hello Timer =         15.
Initial Listener Timer =         30.
SVC Descriptor - Bias,Address  =     000000      000000
Routing Priority for Circuit =          64.
Number of Routers per NI =         10.
DLM Flags =    000
Corruption Loss =    000
```

NOTE

The counters are displayed (32 bit decimal values) as <low 16 bits.>, <high 16 bits.>. The actual number is [<high 16 bits.> * 32767.] + <low 16 bits.>.

SLT

The SLT structure is used to display the System Line Table.
$SLTMA points to the system line table, which
contains the addresses of the SLTs.
You must determine the address you wish to examine before
you can display a SLT.

```
POD>examine $SLTMA
$SLTMA/    x
POD>examine x:x+20
x/         a
x+2/       b
  .        .
  .        .
  .        .
x+20/      c
POD>examine/str=slt a
```

The NETLIB macro is SLTDF$.

Example:

```
        System Line Table
Flags Word =     142410
Buffer Wait Queue Count =     0
L.FLG =     LF.ACT!LF.RDY!LF.ENA!LF.BRO!LF.TIM
DDM PDV Index =     020
DDM Line Table Address =     130532
DLC PDV Index =     014
DLC Line Table Bias =     015054
DLC Line Table Virtual Address =     125750
     Line Owner =     000
     Controller No. =     000
     Unit No. =     000
     No. of Stations =     005
     Cost  =          1.
     State =     On
     Sub-State =     Closed
```

EPMLT

The EPMLT structure describes the EPM Line Table. To access this your region must be set to NT.EPM. You must find the appropriate SLT entry for EPM because this structure contains the address of the EPM Line Table. If you look through the SLT entries you will find a "DLC PDV Index". This value can be used as an offset into the PDV vector. If you examine the address at this offset as a PDV structure, you will be able to tell by the Process Name whehter the SLT entry is for EPM. When you have the correct SLT entry, EPM will show up as the Process Name in the corresponding PDV.

Once you've determined that you are looking at the correct SLT, use the "DLC Line Table Bias" as mapping through APR 6 and examine the "DLC Line Table Virtual Address" as the address to EPM's Line Table.

```
POD>examine/structure=EPMLT  x
```

The NETLIB macro is EPMDF$.

Example:

```
        EPM Line Table
Current Timer =          1.
Initial Timer =          1.
System Line Number =          0.
Controller Flags =    014
C$FLG =    CF$CHN!CF$SID
Active Ports - 1:  =          0.
Started Ports - 1:  =          0.
Local Workspace =    000000     000034
Destination Address From Last Received Message =    $00AA     $0004     $10A1
Source Address From Last Received Message =    $00AA     $0004     $1011
Protocol from Last Message =    001540
Pending Control Function Queue Listhead =    000000
Active Control Function =    000000
Pending Network Management Function Queue =    000000     126006
DDM PDV Index =    000
Timer Cell for System ID Message =          0.
Timer Refresh Value =          0.
Device Hardware Address =    $00AA     $0004     $10A1
Line Table Extension for Network Management =    000000     000000     125150
Data Overrun Counter =          0.
Unrecognized Frame Destination Counter =       4983.
Port Table Listhead =    126052     126242
Protocol Table Listhead =    125700     125700
```

EPMPT

The EPMPT structure describes the EPM Port Tables. To access this structure, your region must be set to NT.EPM. The address for the Port Tables, can be found by examining the EPM Line Table. The address is referenced by "Port Table Listhead" You use the first address specified to examine the first entry in the port table.

```
POD>examine/structure=EPMPT x
```

Chaining down the port table is done by

```
POD>examine
     or
POD>chain
```

The NETLIB macro is EPMDF$.

Example:

```
EPM Port Table
System Line Number =          0.
Station Number =          0.
Port Flags =    200
Port is Active
Multicast Address Chain Listhead =    125714     125714
Seconds Since Last Zeroed =          0.
Data Blocks Received =      2175.           0.
Bytes Received =     29750.           1.
Data Blocks Sent =      2183.           0.
Bytes Sent =      5828.           1.
User Buffer Unavailable =          0.
```

NOTE

The counters are displayed (32 bit decimal values) as <low 16 bits.>, <high 16 bits.>. The actual number is [<high 16 bits.> * 32767.] + <low 16 bits.>.

EPMPRT

The EPMPRT structure describes the EPM Protocol Table. To access this structure, your region must be set to NT.EPM. The address for the Protocol Table, can be found by examining the EPM Line Table. The address is referenced by "Protocol Table Listhead" You use the first address specified to examine the first entry in the protocol table.

```
POD>examine/structure=EPMPRT x
```

Chaining down the protocol table is done by

```
POD>examine
     or
POD>chain
```

The NETLIB macro is EPMDF$.

Example:

```
EPM Protocol Table
Protocol flags =    005
L$FLG =    LF$PAD!LF$EXC
```

PLB

The PLB structure describes the Pysical Link Block. The XPT database contains an address, "Physical Link Block Vector Address", which points you to a vector of PLB addresses. Use the first address in this vector to start examining PLBs.

```
POD>examine/structure=PLB x
```

Chaining down the physical link block is done by

```
POD>examine
     or
POD>chain
```

The NETLIB macro is PLBDF$.

Example:

```
        Physical Link Block
Link State =     PS$UP  - Up
Adjacent Node Type =    201
     Broadcast Channel
     Level 2 Routing Node
Link Recovery Flag =
Count of Outstanding Control Requests =     0.
Recovery Timer =          0.
XPT Channel Number =          1.
Number of Messages Queued =          0.
Pending Control function Queue =     000000
General Protocol Timer =         13.
Flags =     000005
     Link has been Enabled
Desired Link State:    =     Up
Input Packet Limit =          0.
Designated Router Countdown Timer =          0.
Maximum Delay for Routing Message Level 1 =          0.
Maximum Delay for Routing Message Level 2 =          0.
Level 1 State =     000
Level 2 State =     000
Transport Block Size =        576.
Count of Adjacent Nodes of Low TSIZ =          1.
Store and Forward Queue Listhead =     000000     072522
Transport Counter Block Address =     140000
16 Bit Address of Designated Router =     010176
Number of Routers on NI =         10.
Circuit Routing Priority =         64.
```

PAMLTB

The PAMLTB structure describes the PAM Line Table. To access, set region to NT.PAM. You must find the appropriate SLT entry for PAM because this structure contains the address of the PAM Line Table. From the SLT entry, use the "DDM Line Table Address" as the address to examine.

```
POD>examine/structure=PAMLTB  x
```

The NETLIB macro is PMTDF$.

Example:

```
PAM Line Table

General timer: current, reset value =     000      002
Hotswap timer: current, reset value =     000      036
Transmit timer: current, reset value =     000      012
Physical unit number =     000
System line number =     001
Pending control CCB address =     000000
Controller table address =     136274
Cursor position and page count =     000      000      000

Line powerfailed or never enabled

Receive Segment Descriptor Block Controls

Descriptor block starting address =     000000
Descriptor block ending address =     000000
Address of next descriptor to complete =     000000

Transmit Segment Descriptor Block Controls

Descriptor block starting address =     000000
Descriptor block ending address =     000000
Address of next descriptor to build =     000000
Address of next descriptor to complete =     000000
Transmit CCB listhead =     000000      136564

Line Characteristics

Line card type =     **  000000  **
Line protocol =     DDCMP
Asynchronous line transmit speed =     19200 baud
Asynchronous line receive speed =     19200 baud
Synchronous line speed =     **  000037  **
Character length =     8 bits
Station number =     000001
```

If ASYNC protocol, two stop bits used
Line card synchronous clock enabled
AST dispatches to DLC enabled

PAMCTB

The PAMCTB structure describes the PAM Controller Table. To access, set region to NT.PAM. Use the "Controller Table Address" in the PAm Line Table as the address to examin.

```
POD>examine/structure=PAMCTB  x
```

The NETLIB macro is PMTDF$.

Example:

```
PAM Controller Table

Controller number =     000
First CSR address =     171300
Second CSR address =     171302
Status interrupt priority mask =     000100
Error interrupt priority mask =     000100
Network pool allocation bias =     014124
Network pool allocation address =     140000
Controller powerfailed or uninitialized

Command Descriptor Block Access Controls

Number of descriptors in block =     000004
Descriptor block starting address =     000000
Descriptor block ending address =     000000
Address of next descriptor =     000000

Status Descriptor Block Access Controls

Number of descriptors in block =     000020
Descriptor block starting address =     000000
Descriptor block ending address =     000000
Address of next descriptor =     000000

Small Free Segment Descriptor Block Access Controls

Number of descriptors in block =     000020
Descriptor block starting address =     000000
Descriptor block ending address =     000000
Address of next descriptor =     000000

Large Free Segment Descriptor Block Controls

Number of descriptors in block =     000020
Descriptor block starting address =     000000
Descriptor block ending address =     000000
Address of next descriptor =     000000
```

Buffering and Interrupt Timer Controls

```
Number of available lines =    000
Number of lines enabled =    000
Minimum receive buffering level =    002
Maximum receive buffering level =    004
Current receive buffering level =    000
Maximum transmit buffering level =    012
Next line to post buffer request on =    017
RDBs: number required =    000
RDBs: number allocated =    000
SDBs: number required =    000
SDBs: number allocated =    000
Receive CCB listhead =    000000     136466

LAT protocol timer =    000
CTERM protocol timer =    017
Line card status timer =    377
Driver process index =    000000
```

Line Table Address List

```
Line table 0 address =    000000
Line table 1 address =    000000
Line table 2 address =    000000
Line table 3 address =    000000
Line table 4 address =    000000
Line table 5 address =    000000
Line table 6 address =    000000
Line table 7 address =    000000
Line table 8 address =    000000
Line table 9 address =    000000
Line table 10 address =    000000
Line table 11 address =    000000
Line table 12 address =    000000
Line table 13 address =    000000
Line table 14 address =    000000
Line table 15 address =    000000
```

Default ASYNC Protocol Parameters

```
Line width =    120
Page length =    030
Stop length =    030
Carriage return fill count =    000
Line feed fill count =    000
Transmit work mask zero =    377
Transmit work mask one =    377
XON  character =    021
XOFF character =    023
Receive work mask =    375
```

Default DDCMP Protocol Parameters

Select timer value = 011
Reply timer value = 011
Number of SYNC characters = 010
Number of PAD characters = 007
Maximum abutted messages = 002

Default SDLC Protocol Parameters

Number of trailing FILL characters = 002

APCLTB

The APCLTB structure describes the APC Line Table. To access, set region to NT.APC. You must find the appropriate SLT entry for the PAM because this structure contains the address of the APC Line Table. From the SLT entry, use the "DLC Line Table Virtual Address" as the address to examine.

```
POD>examine/structure=APCLTB  x
```

Example:

```
APC Line Table

Process timer: current, reset =    000     000
Address of pending enable/disable CCB =    000000
Address of pending start/stop CCB =    000000
Number of pending transmits =    000000
System line number =    001
Line protocol =    ASYNC
Line state =    Halted
Network management listhead =    000000     123670
Time since counters zeroed (unformatted): line =    000000
Time since counters zeroed (unformatted): circuit =    000000
```

NOTE

There is 1 other data structures which has not been documented. This will be made available at a later date.

CIBLK - Describes the Connect Initiate Block

If you wish to examine this structure, reference your Communications Executive Documentation.

APPENDIX B

SAMPLE SESSIONS

```
$ !
$ !
$ !      Command File To Show Current Settings
$ !      (Used in following two sample sessions.)
$ !
$ !
$ TYPE SHOW.COM
SHOW SERVER
SHOW IMAGE
SHOW REGION
SHOW TYPE
SHOW LENGTH
```

```
$!
$!      Example Of Using Pod To Look At A Distribution Image
$!
$!
$ RUN POD
Pluto On-Line Debugger  -  V3.0.0
POD>SET SERVER BASE
POD>SET IMAGE DIST=SBBL51

               ServerBaseBL5.0 Phase IV Endnode
POD>SET REGION EXECUTIVE
POD>@SHOW
POD>
Current server is Server Base
Symbol library file is PODSB.SYM
Executive symbol table contains     908 symbols
POD>
Current image is server distribution SBBL51.SYS
Access: Examine
POD>
Current region is EXECUTIVE
POD>
Default type setting(s):    OCTAL
POD>
Default length setting:    WORD
POD>
POD>
POD>EXA $TKPS

  5726/        74
POD>EXA/DEC $TKPS

  5726/         60.
POD>SET TYPE HEX,OCT,DEC
POD>SHO TYP
Default type setting(s):    DECIMAL    HEXADECIMAL    OCTAL
POD>EXA $TKPS

  5726/         60.    $  3C        74
POD>SET TYPE OCTAL
POD>EXA 0:10

     0/      1614
     2/      4625
     4/     15564
     6/       340
    10/     15460
POD>EXIT
```

```
$ !
$ !
$ !     Example Of Examining A Dump File
$ !
$ !
$ RUN POD
Pluto On-Line Debugger  -  V3.0.0
POD>SET SERVER X25
POD>SET IM DU=EOMER1

               X25 Gateway( Router )
POD>SET REG SBI...
POD>SET LEN BYTE
POD>@SHOW
POD>
Current server is PSI Gateway
Symbol library file is PODX25.SYM
Executive symbol table contains     908 symbols
POD>
Current image is crash dump file EOMER1.CDA
Access: Examine
POD>
Current region is SBI...
Virtual address base is 120000
Region symbol table contains   311 symbols
POD>
Default type setting(s):    OCTAL
POD>
Default length setting:    BYTE
POD>
POD>
POD>SHO SYM SYSPAS
Value is   120722
POD>E/O/A SYSPAS:SYSPAS+4

120722/     4    '
120723/   120    'P
120724/   122    'R
120725/   111    'I
120726/   126    'V
POD>EXA BKPHST

120644/     5
POD>EXA/AS BKPHST+1:BKPHST+5

120645/    'S
120646/    'M
120647/    'A
120650/    'U
120651/    'G
POD>SHO SYM $EXSIZ
Value is     5462
POD>EXA/WORD $EXSIZ
```

```
  5462/    112000
POD>^Z
```

```
$ !
$ !
$ !
$ !      Example Of Using Pod To Examine And Deposit Into
$ !      A Running Server Image
$ !
$ !
$ RUN POD
Pluto On-Line Debugger  -  V3.0.0
POD>sho log
Current log file is:       EXAM.LOG
POD>sho output
Current output is:       TERMINAL  LOGGING
POD>set ser router
POD>set imag/dep/unp nod=eomer

                DECnet Router Server V1.0
POD>set reg executive
POD>!
POD>!
POD>! Examine PCB structure
POD>!
POD>exa $parhd

$PARHD       /   111734
POD>ex/st=pcb @

Partition Control Block
P.LNK =     111670
Priority =          0.
IO + IOSB count =           0.
Partition Name =     CEXPAR
P.SUB =     000000
P.MAIN =     111734
P.REL =     001120
P.SIZE =     000060
Wait Queue =     000000
Swap Size =     111754
Busy Flags =     200      200
P.TCB =     000000
Starting APR =     0
Common
POD>!
POD>! Chain to next pcb
POD>!
POD>ch

Partition Control Block
P.LNK =     111624
Priority =          0.
IO + IOSB count =           0.
Partition Name =     NTPOOL
P.SUB =     057524
P.MAIN =     111670
```

```
P.REL =     001200
P.SIZE =     003740
Wait Queue =     000000
Swap Size =     111710
Busy Flags =     200      200
P.TCB =     000000
Starting APR =     0
System controlled
POD>!
POD>! Chain down the sub-partition
POD>!
POD>ch p.sub

Partition Control Block
P.LNK =     000000
Priority =          0.
IO + IOSB count =          0.
Partition Name =     SBPOOL
P.SUB =     000000
P.MAIN =     111670
P.REL =     001200
P.SIZE =     003703
Wait Queue =     000000
Swap Size =     000000
Busy Flags =     000      000
P.TCB =     000000
Starting APR =     0
Not shuffleable
Common
System controlled
POD>!
POD>!
POD>! Redisplay this data in another output format
POD>!
POD>disp/oct/dec/by

 57524       /      0.      0
 57525       /      0.      0
 57526       /      0.      0
 57527       /      0.      0
 57530       /     32.     40
 57531       /    119.    167
 57532       /     36.     44
 57533       /     96.    140
 57534       /      0.      0
 57535       /      0.      0
 57536       /    184.    270
 57537       /    147.    223
 57540       /    128.    200
 57541       /      2.      2
 57542       /    195.    303
 57543       /      7.      7
 57544       /      0.      0
 57545       /      0.      0
```

```
 57546       /      0.      0
 57547       /      0.      0
 57550       /      0.      0
 57551       /      0.      0
 57552       /      0.      0
 57553       /      0.      0
 57554       /    160.    240
 57555       /      1.      1
POD>!
POD>! Show structures which can be displayed
POD>!
POD>SHO STRUCT


   Data Structures

DECHB
FEATR
ECLDB
MBX
WDBLK
XPT
NICHE
RNN
ECLNC
XPTNC
XPTLC
PLB
CIBLK
PDV
SLT
LLT
EPMLT
EPMPT
EPMPRT
ETHADD
ETHADD
CRASH
CLKQ
DCB
TSKHDR
HDRWND
ADB
PCB
AST
OCB
IORB
SRDB
SRRDB
SCB
TCB
UCB

POD>EXA $DECPT
```

```
$DECPT      /   102200
POD>!
POD>!
POD>! Examine DECnet Home Block Structure
POD>!
POD>E/ST=DECHB @

     DECnet Home Block Format
Alias Node Name Listhead =    000000
Remote Node Name Listhead =    102204
Pointer to End of Remote Node Name List =    030235
Local Node Name =    EOMER
Local Node Number
     Area: =    4.
     Node Number: =    126.
Local Node Identification =    DECnet Router Server V1.0
Host Node Address
     Area: =    4.
     Node Number: =    19.
Hi-order 4 bytes of Ethernet Address =    $00AA     $0004
Number of Routing Channels =       10.
DLL Service Database Address =    000000
DLL Database FNB Address =    000000
ECL Segment Size =       558.
POD>!
POD>!
POD>! Must set up bias into extended pool, so that can chain through
POD>! remote node name list
POD>!
POD>BIAS $XBIAS
POD>CH D$RNN

     Remote Node Name List
R.LNK =    030235
Node Name =    EOMER
Address
     Area: =    4.
     Node Number: =    126.
POD>CH

     Remote Node Name List
R.LNK =    030221
Node Name =    VORTEX
Address
     Area: =    2.
     Node Number: =    1.
POD>CH

     Remote Node Name List
R.LNK =    030205
Node Name =    ECHO
Address
     Area: =    2.
```

```
     Node Number: =    2.
POD>CH

     Remote Node Name List
R.LNK =    030171
Node Name =    JEDI
Address
     Area: =    2.
     Node Number: =    3.
POD>!
POD>!
POD>! Cancel Bias Operation
POD>!
POD>NOBIAS
POD>
POD>!
POD>!
POD>! Put a patch in to count the number of times the routine
POD>! $LDBRT is entered. (There is a patch area in the Server Base
POD>! starting at location $PATCH)
POD>!
POD>E $LDBRT:$LDBRT+6

$LDBRT       /    10246
$LDBRT +   2/    10346
$LDBRT +   4/    10546
$LDBRT +   6/    13746
POD>EXA $PATCH:$PATCH+20

$PATCH       /        0
$PATCH +   2/        0
$PATCH +   4/        0
$PATCH +   6/        0
$PATCH +  10/        0
$PATCH +  12/        0
$PATCH +  14/        0
$PATCH +  16/        0
$PATCH +  20/        0
POD>!
POD>! Set address display to octal, so can use octal
POD>! address values when depositing patch
POD>!
POD>SET ADDRESS OCTAL
POD>EXA $LDBRT:$LDBRT+6

112170/    10246
112172/    10346
112174/    10546
112176/    13746
POD>EXA $PATCH:$PATCH+20

   752/        0
   754/        0
   756/        0
```

```
   760/         0
   762/         0
   764/         0
   766/         0
   770/         0
   772/         0
POD>DEP $PATCH=5227,0,10246,10346,137,112174
POD>EXA $PATCH:$PATCH+6

   752/      5227
   754/         0
   756/     10246
   760/     10346
POD>EXA $PATCH:$PATCH+12

   752/      5227
   754/         0
   756/     10246
   760/     10346
   762/       137
   764/    112174
POD>!
POD>!
POD>! You must examine a location performing attempting deposit
POD>! An error message will occur if you don't
POD>!
POD>DEP $LDBRT=137,752
?Must Examine Before Deposit
POD>EXA $LDBRT:$LDBRT+6

112170/     10246
112172/     10346
112174/     10546
112176/     13746
POD>DEP $LDBRT=137,752
POD>!
POD>!
POD>! Now monitor the location which is counting the number of times
POD>! the routine is entered.
POD>!
POD>DEP/UNP $PATCH+2=0
POD>MONITOR/TIM=100/INT=5 $PATCH+2
 Monitor started at 21-Mar-1984 14:13:12


   754/      1106

 ...in progress...

   754/      1703
   754/      2337
   754/      3015
   754/      3510
   754/      4247
```

```
   754/      4721
   754/      5424
   754/      6003
   754/      6651
   754/      7540
   754/     10222
   754/     10653
   754/     11420
 Monitor finished at 21-Mar-1984 14:14:19

POD>!
POD>! Reset the counter location, because it is constantly changing
POD>! the only way to do this is by using the /UNPRO switch on
POD>! the DEPOSIT command.
POD>!
POD>DEP $PATCH+2 =0
?Previous Contents Don't Match
POD>DEP/UNP $PATCH+2=0
POD>EXA $PATCH+2

   754/       255
POD>!
POD>!
POD>! Use the BIAS operation to look at object list in extended pool
POD>!
POD>BIAS $XBIAS
POD>EX $OBJHD

115676/     31565
POD>SET TYP RAD,OCT
POD>EXA 31565:31575

 1642564/     31551    HIA
 1642566/      1000     L2
 1642570/         0
 1642572/         0
 1642574/         0
POD>EX 31551:31561

 1642550/     31535    HH
 1642552/      1023     MK
 1642554/         0
 1642556/     54353    NIC
 1642560/    131574    ...
POD>EX 31535:31545

 1642534/         0
 1642536/      1031     MQ
 1642540/         0
 1642542/     51272    MIR
 1642544/    131574    ...
POD>NOBIAS
POD>!
POD>! Without BIAS set the examine would fail
```

```
POD>!
POD>EX 31535:31545
?Invalid Odd Address Specified
POD>!
POD>! You could set region to physical and look at same locations
POD>!
POD>SET REG PHYSICAL
POD>SHO REG
Current region is PHYSICAL
POD>EXA 1642534:1642544

 1642534/         0
 1642536/      1031    MQ
 1642540/         0
 1642542/     51272   MIR
 1642544/    131574   ...
POD>SET REG EXECUTIVE
POD>!
POD>!
POD>! Next example uses the MAP function to look at ECL's node
POD>! counters off of the ECL data base. To find ECL's data
POD>! base must find its PDV (process descriptor first)
POD>!
POD>E $PDVTA

115470/   117100   YL2
POD>SET TYP OCTAL
POD>EX $PDVTA

115470/   117100
POD>EX 117100:117120

117100/   117474
117102/   117514
117104/   117534
117106/   117554
117110/   117620
117112/   117674
117114/   117712
117116/   117730
117120/   117746
POD>ex/st=pdv 117534

     Process Descriptor
Process =    ECL
Type =    LLC
Priority =    340
Flags =    001214
Z.FLG =    ZF.COU!ZF.TIM!ZF.MFL
     Process Relocation Bias =    016533
     Dispatch Table Address =    123644
     Free Space Pointer =    000000
     PCB Address =    056744
     LLC Data Base Virtual Address =    056664
```

```
     No. of Channels =          0.
          System Line Number =     304
          Station Number =    35
          Channel =
POD>CH ECLDB

     ECL Data Base
ACP CCB Queue Listhead =    000000     056664
Timer Count =          0.
Flags Byte =    040
Flags =    NF$MOU
Function Code =    000002
Dummy VCB =    005622
Source Node Address
     Area: =    4.
     Node Number: =    10.
Round Trip Delay =          4.
Source Link Address =    012130
Destination Link Address =    107401
Error Code =    000051
Mapping of Current LLT =    016301
Current LLT Virtual Address =    140054
Current LLT Physical Address =    017155
Current Active Logical Links =          1.
Current Maximum Logical Links =         20.
Count of CI's Ignored Due to Resources =         48.
Logical Link Table Length =         20.
Logical Link Table Address =    030374
ECL Node Counters Listhead =    140120     140000
ECL Node Counters APR Bias =    016100
Mailbox Queue Listhead =    030220
General Delivery CCB Queue Listhead =    000000
General Delivery Current Timer =         4.
General Delivery Initial Timer =         5.
POD>!
POD>! Map through APR 6 to get to node counters
POD>! Using ECL Node Counters APR Bias above
POD>!
POD>MAP 16100
POD>CH N$ENC

     ECL Node Counters List
E$NLNK =    140360
Node Address
     Area: =    4.
     Node Number: =    24.
Number of Active Logical Links =          0.
Retransmit Period =          4.
Message Start Time =    177777
Segment Number of the Packet Being Timed =          3.
Logical Link Address of Segment =    000000
Number of Bytes Received =       135.          0.
Number of Bytes Sent =      2491.          0.
Number of Messages Received =       236.          0.
```

```
Number of Messages Sent =       243.          0.
Number of Connects Received =        20.
Number of Connects Sent =          0.
Maximum Logical Links =          1.
Number of Response Time-outs =          0.
Number of Connects Ignored =          0.
Time this Counter Block Last Zeroed =    002156
POD>NOMAP
POD>
POD>!
POD>! Demonstrate HELP command
POD>!
POD>HELP


        POD, (PLUTO On-Line Debugger)


The following is a list of available commands:

* @                          * SET LENGTH
* BIAS                       * SET LOG
* BREAK                      * SET OUTPUT
* CONVERT                    * SET REGION
* DEFINE                     * SET SERVER
* DEPOSIT                    * SET TYPE
* DISPLAY                    * SHOW IMAGE
* EXAMINE                    * SHOW LENGTH
* EXIT                       * SHOW LOG
* HELP                       * SHOW OUTPUT
* MAP                        * SHOW REGION
* MONITOR                    * SHOW SERVER
* NOBIAS                     * SHOW STRUCTURE
* NOMAP                      * SHOW SYMBOL
* SET ADDRESS                * SHOW TYPE
* SET IMAGE                  * UNDEFINE


POD>!
POD>!
POD>! Demonstrate SHO SYMBOL command
POD>!
POD>SHO SYM T.*

Symbol      Value    Symbol Table

T.ACTL     000052       SYSTEM
T.ASTL     000016       SYSTEM
T.ATT      000062       SYSTEM
T.CPCB     000004       SYSTEM
T.DPRI     000040       SYSTEM
T.EFLG     000022       SYSTEM
T.EFLM     000104       SYSTEM
T.EXT      000000       SYSTEM
```

```
T.GGF       000110      SYSTEM
T.HDLN      000110      SYSTEM
T.IOC       000003      SYSTEM
T.LBN       000041      SYSTEM
T.LDV       000044      SYSTEM
T.LGTH      000112      SYSTEM
T.LNK       000000      SYSTEM
T.MXSZ      000050      SYSTEM
T.NAM       000006      SYSTEM
T.OCBH      000076      SYSTEM
T.OFF       000066      SYSTEM
T.PCB       000046      SYSTEM
T.PRI       000002      SYSTEM
T.RCVL      000012      SYSTEM
T.RDCT      000102      SYSTEM
T.RRFL      000072      SYSTEM
T.SAST      000054      SYSTEM
T.SRCT      000071      SYSTEM
T.STAT      000032      SYSTEM
T.ST2       000034      SYSTEM
T.ST3       000036      SYSTEM
T.TCBL      000030      SYSTEM
T.TIO       000057      SYSTEM
T.TKSZ      000060      SYSTEM
T.UCB       000026      SYSTEM

POD>EXIT
```