EXOS 8031 and EXOS 8032

TCP/IP Protocol Package
For PDP/RSX Systems

Reference Manual

Publication No. 4200014-00
Revision A August 8, 1985

Excelan, Inc.
2180 Fortune Drive
San Jose, CA 95131

The TCP/IP code in the EXOS 8031 and EXOS 8032 software is derived from BBN code as modified by Berkeley. The initial implementation of the TCP/IP code on the EXOS Ethernet front-end processor was done by Bruce Borden, Kipp Hickman, and Bill Northlich.

EXOS 8031 and EXOS 8032
Reference Manual

REVISION HISTORY

REVISION	DATE	SUMMARY OF CHANGES
A	08-08-85	Release A. EXOS 8031 and EXOS 8032 TCP/IP Protocol Package For PDP/RSX Systems Reference Manual Publication No. 4200014-00

PREFACE

This document describes the EXOS 8031 and EXOS 8032 TCP/IP Protocol Packages. (For convenience in this manual, the EXOS 8031 and EXOS 8032 are referred to as the EXOS 8031 when there is no difference between the two. Differences between the products are explicitly indicated.) The EXOS 8031 TCP/IP software is a vertically integrated implementation of the DARPA protocol standards for computer systems running the RSX-11M operating system. This manual covers the information necessary to integrate the EXOS 8031 package with a RSX-based system that has an EXOS 203 or EXOS 204 Ethernet front-end processor installed in it. The manual also provides information on how to use the various programs supplied as part of the package.

On an Ethernet network, RSX-based host systems using the EXOS 8031 can communicate with any system on the network that supports the standard TCP/IP protocols.

The EXOS 8031 package includes two network application utilities and several network systems utilities. The application utilities allow a user to transfer files between compatible hosts and to emulate a virtual terminal connected to a remote host. The systems utilities allows a user (typically, a network/system administrator) to manipulate the network database.

The software in the EXOS 8031 Protocol Package is compatible with the current versions of the Department of Defense TCP, IP, ICMP, UDP, ARP, FTP, and TELNET protocols.

The following is a list of reference and study material related to EXOS 8031. The EXOS 8031 implements protocols defined by the following ARPA documents:

[1] Postel, J., ed., *Transmission Control Protocol – DARPA Internet Program Protocol Specification,* RFC 793, USC/Information Sciences Institute, September 1981.

[2] Postel, J. ed., *Internet Protocol – DARPA Internet Program Protocol Specification,* RFC 791, USC/Information Sciences Institute, September 1981.

[3] Postel, J., *Internet Control Message Protocol – DARPA Internet Program Protocol Specification,* RFC 792, USC/Information Sciences Institute, September 1981.

[4] Postel, J., *User Datagram Protocol,* RFC 768, USC/Information Sciences Institute, August 1980.

[5] Postel, J., *File Transfer Protocol,* RFC 765, IEN 149, USC/Information Sciences Institute, June 1980.

[6] Postel, J., *Telnet Protocol,* RFC 764, IEN 148, USC/Information Sciences Institute, June 1980.

[7] Plummer, David C., *An Ethernet Address Resolution Protocol* RFC 826, Network Working Group, November 1982.

The EXOS 8031 is designed for use with the EXOS Ethernet front-end processors, which are described in the following documents:

[8] Excelan, Inc., *EXOS 204 Ethernet Front-End Processor Reference Manual*, 1984.

[9] Excelan, Inc., *EXOS 203 Ethernet Front-End Processor Reference Manual*, 1984.

The following reference describes the C language, which is used for procedural and data structure specifications in this manual:

[10] Kernighan, B.W. and Ritchie, D.M, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

The following reference describes the ISO Open Systems Model:

[11] International Organization for Standardization (ISO), "Open Systems Interconnection – Basic Reference Model" Document no. ISO 7498-1984 (E).

Because the EXOS 8031 is used with Ethernet, the following reference is useful:

[12] DEC, Intel, and Xerox Corporations, "The Ethernet: A Local Area Network: Data Link Layer and Physical Layer Specifications," Document No. T588.B/1080/15K, Intel Corp., September 1980.

The following reference, which is a multivolume set, describes the RSX-11M operating system:

[13] DEC, *RSX-11M System Documentation*, 1984.

DOCUMENTATION CONVENTIONS

A number of typographic conventions are used in this manual to present various types of information. You should be familiar with these conventions before using this manual.

Numerical constants are given in decimal notation except in the following cases. Hexadecimal numbers are either prefixed with a 0x or 0X or postfixed with an H. Octal numbers are followed by the word (octal).

Italicized characters or words in the context of a command indicate a variable for which you must supply a value. For example, the command specification

lls *directory_name*

means that you must provide the name of a directory. Also, when a utility names is used in text, it is italicized. For example,

The *ftp* utility...

Bold characters or words indicate responses that you type to system prompts. These should be typed exactly as shown, unless the text indicates otherwise.

Brackets ([ ]) in command lines enclose optional arguments or parameters. If you include the option or parameter in the command line, do not type the brackets. In text brackets indicate RSX-11M directory names.

(blank page)

TABLE OF CONTENTS

Chapter **Page**

Chapter **Page**

Appendix **Page**

LIST OF TABLES

Chapter 1
INTRODUCTION

1.1. INTRODUCTION

The EXOS 8031 PDP TCP/IP Software Package is a set of protocol software modules and network utilities for connecting host systems to an Ethernet network. The EXOS 8031 connects Q-Bus-based PDP-11 or LSI-11 hosts to an Ethernet through an EXOS 203 Ethernet front-end processor board. The EXOS 8032 connects UNIBUS-based PDP-11 hosts to an Ethernet through an EXOS 204 Ethernet front-end processor board. (For convenience in this manual, the EXOS 8031 and EXOS 8032 are referred to as the EXOS 8031 when there is no difference between the two. Differences between the products are explicitly indicated.) Typically, the host systems that use the EXOS 8031 software are DEC's PDP-11/LSI-11 family of minicomputers.

The EXOS 8031 software provides the following benefits:

- Transfer files between the RSX-11M system and other compatible systems.
- Use a virtual terminal connection between the RSX-11M system and other compatible systems.
- Write additional applications for process-to-process communications between the RSX-11M system and other compatible systems.

The following systems are compatible with RSX-11M systems that use the EXOS 8031 software:

- Other RSX-11M systems running the EXOS 8031 software
- UNIX 4.2BSD systems with Berkeley networking
- Any VAX/VMS system running the EXOS 8043 software
- Any VAX System V or System V.1 systems running the EXOS 8015 software
- UNIX Version 7, Version 4.2 BSD, System III, System V, and XENIX running the EXOS 8010 software
- Unisoft's BNET and 3COM's UNET
- IBM PC family and compatible systems running PC-DOS and the EXOS 8051
- IBM PC AT and compatible systems running XENIX and the EXOS 8011 software
- All other systems that conform to ARPANET/DOD specifications for TCP/IP and application protocols

The EXOS 8031 software package components are organized into two parts:

- The TCP/IP Protocol Module
- Host Utilities and Integration Kit

The TCP/IP Protocol Module is an implementation of the Department of Defense standard ARPANET TCP/IP protocols. The protocol module (also referred to as protocol software in this manual) is downloaded to and executes on the EXOS front-end processor. The protocol module contains object code for

- IP – the Internet Protocol,
- ARP – the Address Resolution Protocol,
- ICMP – the Internet Control Message Protocol,
- TCP – the Transmission Control Protocol,
- UDP – the User Datagram Protocol,
- TELNET – the TELNET Protocol (the server portion), and
- FTP – the File Transfer Protocol.

The Host Utilities and Integration Kit consists of a set of network application programs, an I/O driver, and command files for the installation of the EXOS 8031 software. This software resides in and executes on the host system. The I/O driver (also known as the device driver) provides the linkage between the network utilities running on the host and the protocol software running on the EXOS front-end processor. The network utilities are supplied as executable files; the device driver is supplied as an object library together with assembly language source files suitable for assembly by the MACRO-11 assembler. The driver is assembled and linked with host operating system at the time of EXOS 8031 installation.

The EXOS 8031 and EXOS 8032 packages can be installed easily on a RSX-11M-based host system that is connected to an Ethernet system through an EXOS 203 or EXOS 204 front-end processor, respectively. Most of the installation steps are executed by the command files that are included as a part of the EXOS 8031 package.

The EXOS 8031 TCP/IP package and an EXOS 203 Ethernet front-end processor board or the EXOS 8032 and an EXOS 204 form, in effect, a node on the network. In this node, various levels defined in the Open Systems Interconnection (OSI) reference model of the International Standards Organization (ISO) are implemented as follows.

OSI Levels	Implemented in EXOS Subsystem in
Session, Presentation, Application	Modules that execute on the host system
Network, Transport	Modules of the EXOS 8031 package that are downloaded to and executed on the EXOS front-end processor board
Physical, Data Link	Hardware and firmware on the EXOS Ethernet front-end processor board

In the traditional architecture used for network integration of minicomputers and mainframes, the network and transport levels of the OSI model are implemented in the host system. However, as shown above, an EXOS node implements the network and transport levels on the EXOS 203 or EXOS 204 Ethernet front-end processor. This "front-end approach" provides the following advantages:

- Maximum Host CPU Availability.
  Network protocols, especially at the Transport level, can consume a large amount of CPU time when run on the host system. Running these protocols on a front-end processor considerably reduces the demand on the host CPU time.

- Economical Host Memory Usage.
  Network protocol code and buffer space requirements increase the amount of host memory used. With the front-end approach this contention for the host memory space is practically eliminated. The host system can use the memory that would otherwise be required for the protocol for other applications.

- High Network Throughput.
  Because more CPU time and memory are available to the host system and because the EXOS front-end processor's environment is optimized for high-level protocol execution, the network throughput for the host increases sharply.

- Rapid Integration.
  In a conventional system, the protocol software must be tightly coupled to the host operating system before it can run on the host system. This can take a significant amount of development and debugging effort, which is not required with the front-end approach.

- Flexibility.
  In the front-end approach, the protocol software is not coupled to the host operating system; the network applications and the operating system communicate with the protocol software through the I/O driver. Therefore, the protocol software can be easily integrated with different host operating systems simply by modifying the I/O driver.

1.1.1. The EXOS 8031 Protocol Module

The EXOS 8031 protocol module part of the EXOS 8031 TCP/IP protocol package runs on the EXOS front-end processor. The module provides high-level protocol services to a host system. It implements the following Department of Defense ARPA Internet protocols:

IP The Internet Protocol provides packet routing, fragmentation, and reassembly through the Data Link level. Fragmentation normally does not occur.

ARP The Address Resolution Protocol translates Internet addresses to physical Ethernet addresses.

ICMP The Internet Control Message Protocol is an adjunct to IP. ICMP conveys control and error information. The host system cannot access this protocol.

TCP The Transmission Control Protocol provides reliable, full-duplex, process-to-process data stream connections. It is based on IP, and adds reliability, flow control, multiplexing, and connections.

UDP The User Datagram Protocol provides simple but unreliable IP-based datagram services. It adds a checksum and additional process-to-process addressing information.

TELNET
The TELNET Protocol server provides a simple virtual terminal communication. It passes data from the remote TELNET host to the terminal driver of the local host. The TELNET server executes on the front-end processor for ease of integration and high performance.

FTP The File Transfer Protocol provides for the transfer of files between systems and for the remote management of a file system. (This server runs on the host, not on the front-end processor.)

These protocols are based on the current DARPA standards. They can be used to communicate with any implementation that adheres to the DARPA specifications. Internet Protocol specifications are described in *Internet Protocol Transition Workbook*, which is available from

Network Information Center SRI International Menlo Park, CA 94025

In a typical application, the host system initializes the EXOS front-end processor and configures it for operation in the front-end mode. Then the host downloads the protocol module part of the EXOS 8031 software package to the front-end processor.

Once the downloading is complete, the host system sends a start message to the EXOS board, which begins executing the protocol software. After this, the host sends protocol service request messages to the front-end processor through the host-to-EXOS message queue. The requests contain parameters such as the protocol type, a connection ID (for TCP), and pointers to data buffers. When the protocol module has processed the service request, it sends a reply message to the host through the EXOS-to-host message queue.

The protocol software is supplied as an executable file that executes on the EXOS 203 or EXOS 204 front-end processor board. The protocol software runs under the control of and uses services provided by the on-board, PROM-based NX kernel.

The protocol software download and execution procedures and the NX kernel are described in the *EXOS 203 Ethernet Front-End Processor Reference Manual* or *EXOS 204 Ethernet Front-End Processor Reference Manual.*

Because the EXOS 8031 is a layered communications product, compatibility with other TCP/IP implementations is important at several interface levels. The following table shows the correspondence of these levels with EXOS 8031 protocols:

Interface Level	Corresponding EXOS 8010 Protocol(s)
Application	FTP, TELNET
Transport	TCP, UDP
Network	IP, ICMP, ARP
Data link	Ethernet

Starting with the lowest interface level, the EXOS 8031 network hardware communicates with systems that support Ethernet Version 1.0 or 2.0. If the Ethernet has an Internet gateway to other data links, the EXOS 8031's IP protocol also accesses these links through this gateway.

At the network level, the EXOS 8031 communicates with systems that implement the current DARPA specification for the IP protocol. Currently, the type-of-service feature and option field are not supported.

At the transport interface level, the same considerations for IP apply to TCP and UDP. Given compatibility at the network level, TCP and UDP communicate with implementations that use the current specification. Interoperation has been tested and confirmed on the implementations explicitly listed on Page 1-1.

Protocols are more diverse at the application interface level. The EXOS 8031 implements the DARPA File Transfer Protocol (FTP) and the TELNET protocol. However, not all FTP or TELNET protocol features are implemented. Refer to Appendix A for a discussion of the features that are implemented on the 8031.

1.1.2. The EXOS 8031 Host Utilities and Integration Kit

The Host Utilities and Integration Kit software consists of the I/O driver and several network utilities.

The kit provides the software to integrate the EXOS 8031 TCP/IP protocol package with the host operating system. The low-level system software is written in assembly language; the utilities are coded in C and assembly languages. The software is distributed as a combination of source and executable image files for rapid integration into RSX-11M. Command procedures are included to automate compiling/assembling, linking the source files, and installing the software.

The I/O driver serves as a message-passing mechanism between user application programs and the front-end processor. The driver is delivered as an object library together with MACRO-11 source files. A command procedure customizes it for the local hardware configuration, assembles it, and loads it. The system administrator can load the driver automatically when the system is

booted. Application programmers can access the driver directly using QIO function calls.

The utilities provided as part of the EXOS 8031 are classified into two categories: network systems utilities and network application utilities. The network systems utilities manipulate the network database and, as such, are mainly used by network administrators. The network application utilities provide for communication between hosts on the network. The application utilities are used by anyone who needs to communicate with other systems on the network.

Two application utilities – FTP and TELNET – are included in the EXOS 8031 package.

The File Transfer Program (FTP) is an implementation of the DARPA Internet File Transfer Protocol. Using FTP, you can transfer files between hosts and obtain current status information about the file systems on other hosts.

The TELNET utility uses the DARPA TELNET protocol. Using TELNET, you can connect to a remote host, login, and use the services of the remote host as if you were using an ASCII terminal.

1.1.3. Manual Organization

This manual is organized as follows:

Chapter 1, Introduction, outlines the principal features and organization of the EXOS 8031 TCP/IP software package.

Chapter 2, Installation, describes the step-by-step installation procedure for the EXOS 8031 TCP/IP software package.

Chapter 3, Network Administration, provides a guide to network administration. It describes a network administrator's functions and responsibilities. It also describes the network systems utilities, which are supplied as part of the EXOS 8031 package. The network systems utilities are mainly used by a network administrator, though some of their basic operations can be performed by all users.

Chapter 4, Network Application Utilities, provides a detailed description of each of the network application utilities supplied as part of the EXOS 8031 package. These utilities are used by anyone who needs to communicate with other hosts on the network.

Chapter 5, Programming Interface, documents the available QIO function calls. These calls provide the interface, through the device driver, between the host-resident applications software and the protocol software executing on the EXOS front-end processor.

Appendices A through F provide formal descriptions of the utilities, EXOS 203/EXOS 204 installation guide, QIO function call summary, error status codes, error messages, and troubleshooting, respectively.

1.2. INSTALLATION

The EXOS 8031 TCP/IP software is supplied as a set of RX01 floppy disks or a nine-track, 1600-bpi magnetic tape. The minimum hardware configuration for using this software consists of a PDP-11/LSI-11 minicomputer that has been appropriately connected to an Ethernet network through an EXOS 203 (for the EXOS 8031) or EXOS 204 (for the EXOS 8032) Ethernet front-end processor board and an RX01-compatible floppy disk drive suitable for reading the distribution disks or a nine-track, 1600-bpi tape drive for reading magnetic tapes. The minimum software configuration requirements are that the PDP-11/LSI-11 be running under a compatible release of the RSX-11M operating system and that 1800 512-byte blocks of disk space be available for EXOS 8031 use.

The EXOS 8031 software installs easily using an interactive procedure. Most of the procedure steps are included in the command files supplied as part of the software package. When user input is required, the system prompts for it.

A typical EXOS 8031 TCP/IP software installation takes less than 60 minutes.

1.3. NETWORK ADMINISTRATION

Administration of an Ethernet network includes the following tasks:

- Plan the physical layout of the network.
- Install the EXOS front-end processor in individual systems.
- Physically connect the systems to the network via Ethernet cable, connecters, transceivers, and terminators.
- Install TCP/IP software on individual systems.
- Administer host names and their corresponding Internet and Ethernet addresses.
- Maintain network databases.
- Disconnect/relocate systems.

Once the network has been laid out, the host systems appropriately connected to the network, and the EXOS 8031 software installed, network administration mainly involves management of the following two tasks:

- Administer host names and the corresponding Internet and Ethernet addresses.
- Maintain network databases.

It is convenient for users at all levels to reference a host system by a name, such as *oregon*, *sanjose*, or *finance*. The EXOS 8031 protocol software, however, references hosts by the unique Internet address assigned to each of them. The EXOS 203 or EXOS 204 front-end processor hardware, on the other hand, uses the Ethernet addresses for communicating with other hosts.

The network administrator maintains the Hosts file which contains the host name and the associated Internet address for each host on the network. The protocol software automatically translates Internet addresses to Ethernet addresses, and vice versa.

The network database consists of several files and tables. The network systems utilities, provided as part of the EXOS 8031 software package, allow easy manipulation of the database. This allows a network administrator to perform several routine tasks: reconfigure various options in the protocol software; examine and reset network statistics; enter, delete, and modify Internet-to-Ethernet address translation information; and enter, delete, and modify internetwork routing information.

1.4. NETWORK APPLICATION UTILITIES

Two network application utilities – *ftp* and *telnet* – are provided as part of the EXOS 8031 TCP/IP software package. These utilities provide two important, often-used functions: transfer of files between hosts and use of a terminal on a local host as a virtual terminal connected to a remote host. Typically, these utilities are used by those who need to access remote systems.

The *ftp* utility transfers files between hosts using the DARPA standard FTP protocol. It can also examine and change working directories on the remote host as well as on the local host. The *ftp* utility is normally used in interactive mode. However, it can also be used in batch mode by redirecting the input from a file.

The *telnet* utility allows any terminal on a local host to emulate a virtual terminal, which can then be "connected" to any host on the network that supports the DARPA standard TELNET protocol. Once connected to a remote host, the virtual terminal can be used as a terminal hard-wired to that host.

1.5. PROGRAMMING INTERFACE

A set of QIO function calls is supplied as part of the EXOS 8031 software package. These calls provide a communication facility between an application program and the protocol software through the EXOS I/O driver. The application programs execute on the host system and the protocol software executes on the EXOS front-end processor; the EXOS I/O driver is integrated into the host operating system.

Using the QIO function calls, users can develop their own network application programs in C and other high-level languages.

Chapter 5 provides descriptions of all the QIOs that are supplied as part of the EXOS 8031 software package. This chapter also provides an example of a user-written network application program.

Chapter 2
INSTALLATION

2.1. INTRODUCTION

The EXOS 8031 and EXOS 8032 TCP/IP software is supplied in a set of RX01 floppy disks, numbered sequentially from one, or on a single nine-track, 1600-bpi magnetic tape. It is designed for use on DEC's PDP-11/LSI-11 family of computers that run under the RSX-11M operating system and that are connected to an Ethernet via Excelan's EXOS 203 (Q-Bus, for the EXOS 8031) or 204 (UNIBUS, for the EXOS 8032) Ethernet front-end processor board.

This chapter discusses the hardware and software considerations for installing the EXOS 8031 TCP/IP package and provides a step-by-step procedure for the installation. The procedure for the EXOS 8032 is identical.

2.2. MINIMUM HARDWARE/SOFTWARE CONFIGURATION

The following list describes the minimum hardware and software configurations required to install and use the EXOS 8031 TCP/IP software package:

1. The host computer must be one of the DEC's PDP-11/LSI-11 family of computers.

2. The host computer must be running under the applicable release(s) of DEC's RSX-11M operating system. These release numbers are specified in the Release Notes which are shipped with the EXOS 8031 TCP/IP Software Package.

3. The host must be connected to an Ethernet via Excelan's EXOS 203 (for the EXOS 8031) or EXOS 204 (for the EXOS 8032) Ethernet front-end processor board.

4. The peripherals of the host must include an RX01-compatible floppy disk drive if you are loading the software from floppy disks or a nine-track, 1600-bpi tape drive if you are loading the software from magnetic tape.

5. 1800 blocks (512 bytes each) of disk space should be available for EXOS 8031 package use.

6. The EXOS 203 or EXOS 204 board requires four contiguous bytes, starting at a longword-aligned address, in the host system's I/O memory to establish and maintain communication with the host system. It should be ensured that these locations do not overlap any other device address in the system. If this problem should occur, it can be easily fixed by jumper-selecting different addresses on the EXOS board. (Refer to the *EXOS 203 Ethernet Front-End Processor Reference Manual* or the *EXOS 204 Ethernet Front-End Processor Reference Manual* for further details.)

7. The tasks PIP, MOU, DMO, MAC, TKB, and LBR must be installed in the system. If you are loading the software from magnetic tape, the task BRU must also be installed.

8. The following files must be on your system in the specified directories:

   - LB:[1,1]EXEMC.MLB
   - LB:[11,10]RSXMC.MAC
   - LB:[1,54]RSX11M.STB
   - LB:[1,1]EXELIB.OLB

9. The directories LB:[1,1], LB:[1,2], and LB:[1,54] must exist.

10. The directory [1,100] must exist if you are loading the EXOS 8031 software from magnetic tape. It is used during the installation.

2.3. INSTALLATION PROCEDURE

IMPORTANT

The person responsible for the installation must be a privileged user with a group number of 1.

The EXOS 8031 TCP/IP software is installed interactively. This section details the user interaction with the host system. In the description of the interactive procedure the following conventions are used:

- Regular, nonbold characters represent system responses.
- **Bold** characters represent commands, parameters, or values to be entered by the user.
- *Italicized* characters give descriptive names for parameters to be replaced by user-supplied values.
- Each command line is assumed to be terminated by a carriage return <cr>.

The following steps describe the installation procedure:

1. Note the bus address (Port A) used by the EXOS processor board. This is used in building/installing the driver in Step 10. This address is jumper-selectable; when shipped from the factory, the bus address is 764000 (octal). This is the default address used by the EXOS 8031 software.

2. Login to the system.

INSTALLATION FROM FLOPPY DISKS:

3. If you are loading the EXOS 8031 software from magnetic tape, skip to Step 7. Otherwise, create a directory for initial loading of the EXOS 8031 software. Set this directory to be the default directory.

```
> SET /UIC=[directory_name ]
> UFD [directory_name ]
```

4. Mount the first distribution floppy (labeled "EXOS1") into the floppy disk drive.

   > **ALL** *ddnn:*
   > **MOU** *ddnn:***EXOS1**

   where *dd* is the device mnemonic (for example DX) and *nn* is the unit number (for example, 1).

   If you want to store all the EXOS 8031 utilities in a different directory, create that directory before installing the software.

5. Copy the indirect command file INSTALL.CMD to the default directory. This command file loads the distribution package onto your system.

   > **PIP** =*ddnn:***INSTALL.CMD**

6. Execute the indirect command file INSTALL.CMD to load the EXOS 8031 software.

   > **@install**

   Skip to Step 10.

INSTALLATION FROM MAGNETIC TAPE:

7. Set the UIC to [1,100] and create the directory [1,100] on the system disk:

   > **SET /UIC=[1,100]**
   > **UFD** *xx:***[1,100]**

   where *xx* is the device name for the system disk.

8. Mount the EXOS 8031 distribution tape on the tape drive and restore the files. (If "mt0:" is not the correct device name for your system's tape drive, replace it with the correct name.)

   > **ALL MT0:**
   > **BRU /VER/REW/NOI/DIS/BAC:EXOS8030/NEW MT0:** *xx:***[1,100]**
   > **DEA MT0:**

9. Execute the indirect command file TAPEINS.CMD to load the EXOS 8031 software:

   > @**TAPEINS.CMD**

10. Whenever the command file pauses and asks you to mount another floppy disk, insert the requested disk in the disk drive. The command file also asks several questions. Each question, followed by an explanation of the responses to them, is shown below.

    Verbose? [Y/N]

    Answer YES to keep track of command lines executed by the installation procedure.

    Delete previous version of EXOS software? [Y/N]

    Answer YES to delete previous versions of EXOS 8031 software.

    Delete source file from current UFD in target disk? [Y/N]

    The installation procedure copies source files to the current UFD so it can rebuild the software. If you answer YES, the installation procedure will delete all the source files from the current UFD after it rebuilds the required software.

    Build driver and ACP only? [Y/N]

    Answer YES to rebuild just the EXOS 8031 drivers and the ACP (for instance, after you change the interrupt vector or the port address of the EXOS 203 or EXOS 204 board). In this case, the EXOS 8031 utilities will NOT be copied from the distribution disks.

    Maximum number of concurrent FTP server sessions? [D: 1]

    Enter the number of concurrent FTP server sessions allowed. This number depends on the amount of memory available in your system. Typically, each FTP server session requires 156500 (octal) bytes. Other tasks that requires memory are the following (all numbers are octal):

    - FTP master: 10300 bytes
    - EXOS driver: 1100 bytes
    - Telnet driver: 40000 bytes
    - EXOS ACP: 26300 bytes

    If you enter only a carriage return, 1 is used as the default (this is adequate for a 124 Kword system).

    Interrupt vector location? [D: 400]

    Enter the interrupt vector location. If you enter only a carriage return, 400 (octal) is used as the default.

    Offset address of port A? [D: 4000]

    Enter the port A address offset in octal (offset from 760000). For example, if you select the port A address 764200 (octal), enter 4200. If you enter only a carriage return, 4000 (port A address 764000 [octal]) is used as the default.

    Please enter the UFD for the EXOS utilities?

    Select a directory for the EXOS utilities and enter the UFD here.

11. Initialize the network address file. (See Section 3.5.3.2 for a description of the network address file.) If you are installing the software for the first time, provide the indicated responses to the following three prompts. Do this also in case of subsequent installations if you want to delete the previous network address file and create a new one. (The second and third response lines appear only if a **y** response is entered for the first prompt.)

    Do you want to initialize the network addresses file (HOSTS.NET)?
    Name of host: <*host_name*>
    Host internet address: <*Internet_address*>

    If you are installing the software for the second or any subsequent time and you want to save the previous network address file, enter a carriage return or any other response. In this case, the procedure continues with the next step.

12. The installation procedure then creates the network startup command file in LB:[1,1]EXOSLOAD.CMD.

13. The system displays the following message indicating that the software installation is complete.

    ```
    ;
    ;     Installation completed.  Now you can execute
    ;     @LB:[1,1]EXOSLOAD
    ;     to start up the network connection.
    ```

14. Study the *netload* program description in Chapter 3 and Appendix A. If you decide to include any of the optional features of the program for your system, edit the command file LB:[1,1]EXOSLOAD.CMD to insert the needed optional arguments in the command line that invokes the *netload* program.

15. Create/update the network address file. Edit the file LB:[1,1]HOSTS.NET to enter the Internet addresses and names of other systems on the Ethernet.

16. Edit the system startup command file LB:[1,2]STARTUP.CMD to execute the required EXOS startup procedures. Place the following command line at the appropriate location in the command file:

    @LB:[1,1]EXOSLOAD

17. Execute the following command to download the protocol software to the EXOS 203 front-end processor memory and startup the network:

    > **@LB:[1,1]EXOSLOAD**

Following Step 17, you can invoke and use any of the network systems utilities described in Chapter 3 and the network applications utilities described in Chapter 4.

Chapter 3
NETWORK ADMINISTRATION

3.1. INTRODUCTION

Network administration of an Ethernet includes management of the following tasks:

- Plan the physical layout of the network.
- Install the EXOS 203 or EXOS 204 front-end processor in individual systems.
- Physically connect the systems to the network via Ethernet cable, connecters, transceivers, and terminators.
- Install TCP/IP software on individual systems.
- Administer host names and their corresponding Internet and Ethernet addresses.
- Maintain network databases.
- Disconnect/relocate systems.

It should be mentioned here that it is not necessary for all hosts on the network to be running under the RSX-11M operating system. Nor is it necessary for all hosts to be using the EXOS 203 or EXOS 204 front-end processor boards and the EXOS 8031 software. It is necessary, however, that hosts which are to communicate with each other support the TCP/IP protocols and the application programs to be used for communication. If the Address Resolution Protocol (ARP) is not used, it may also be necessary to emulate a common Ethernet address block to accommodate front-end processors or link-level boards from different manufacturers.

Some of the above-mentioned tasks require use of network system utilities supplied as part of the EXOS 8031 TCP/IP software. These tasks can be grouped into the following topics:

- Network Planning
- EXOS 203 or EXOS 204 Front-End Processor Installation
- TCP/IP Software Installation
- Network Database Management
- Network System Utilities

3.2. NETWORK PLANNING

The physical layout of a network is, in general, dictated by the site where the network is to be installed. However, the layout must meet all the requirements detailed in the Ethernet specifications jointly published by Xerox, Intel, and Digital Equipment Corporation.

An accurate map of the Ethernet layout should be prepared for efficient maintenance, fault isolation, and repairs. This map should show the length of each segment, transceiver locations, terminator locations, and identification of the attached host. The map should be updated whenever a system is added to or disconnected from the network.

3.3. EXOS FRONT-END PROCESSOR INSTALLATION

Before a host system using the EXOS 8031 or EXOS 8032 software can be connected to the network, an EXOS 203 or EXOS 204 front-end processor must be installed in the host. The installation procedure is provided in Appendix B. Also provided in Appendix B is the procedure for connecting the host to the network.

3.4. TCP/IP 8031 SOFTWARE INSTALLATION

The EXOS 8031 TCP/IP software installation procedure is described in Chapter 2. The procedure first copies the device driver source module and the protocol object module (from the EXOS 8031 TCP/IP software floppy disks or magnetic tape) into the host disk storage, compiles and integrates the driver into the RSX-11M operating system. Then, the procedure downloads the protocol software to the EXOS front-end processor memory.

3.5. NETWORK DATABASE MANAGEMENT

As a network administrator, you will be required to manage and maintain the network database, which consists of several files and tables. However, before discussing the database management, it is important to understand the host naming conventions, and Internet and Ethernet addressing schemes. It is also important to understand the logical-to-physical host address translation concept and the function of the Address Resolution Protocol (ARP). In addition, understanding the routing of communications to nodes on other networks is also important.

This section first discusses the above-mentioned topics and then the database management.

3.5.1. Host Names, Internet Addresses and Classes, and Ethernet Addresses

On Ethernet a host can be referenced by a name (previously assigned to it), its Internet address, or its Ethernet address, depending on the context. A host name is a string that can be up to 32 characters long. The Internet address is a 32-bit quantity. It can be specified in a textual form as digit groups consisting of decimal, octal, or hexadecimal digits separated by period(s). The address class of an Internet address is determined by the value in the high-order bits of the Internet address. The address itself consists of a network number and the number of the host on that network. An Ethernet address is the "physical address" of an Ethernet front-end processor and is a 48-bit quantity.

3.5.1.1. Host Names

A host can be assigned any name up to 32 characters (letters, digits, and special characters) long; the first character must be a letter. (Uppercase and lowercase letters are considered equivalent.) This assignment is done by including the name (along with the corresponding Internet address and aliases, if any) in the network address file residing on disk (in the file

LB:[1,1]HOSTS.NET). At the user interface level the host name is used in command lines and in database files. At the system level the name is translated into its logical Internet address, which is eventually translated into the corresponding physical Ethernet address.

3.5.1.2. Internet Addresses and Address Classes

The Internet address for any host is a four-byte numeric value that can be specified in decimal, octal, or hexadecimal form. (Decimal numbers have no prefixes and do not begin with a zero; octal numbers are preceded by a 0 [zero]; hexadecimal numbers are preceded by the 0x or 0X character pair.) The Internet address can be specified in three different formats using the "." (pronounced "dot") notation.

The high-order bits of an Internet address determine its class, as follows:

0	in the high-order one bit indicates	Class A address.
10	in the high-order two bits indicate	Class B address.
110	in the high-order three bits indicate	Class C address.

This means that the first byte of a Class A address must be less than 128, the first byte of a Class B address must be from 128 to 191, and the first byte of a Class C address must be at least 192.

All hosts on a given network must use the same address class. Thus, it is possible to refer to a network as a Class A, Class B, or Class C network.

The Internet addresses of different classes are most conveniently specified as follows:

Class	Bits to Identify Network	Bits to Identify Host
A	8	24
B	16	16
C	24	8

The following example shows a Class A address:

89.0x81.0x01.0x82

The first eight bits specify the network address, which is 89 (decimal). The remaining 24 bits specify the host address, which is 0x81.0x01.0x82.

The following example shows a Class B address:

133.0x00.0x01.0x82

The first 16 bits specify the network address, which is 133 (decimal). The last 16 bits specify the host address, which is 0x01.0x82.

The following example shows a Class C address:

192.0x00.0x00.0x02

The first 24 bits specify the network address, which is 192 (decimal). The remaining 8 bits specify the host address, which is 0x02.

Note that an Internet address of any class can be specified in any dot notation; the high-order bits (not the dot notation format) determine the address class.

3.5.1.3. Ethernet Addresses

The Ethernet address is the unique, six-byte physical address for each Ethernet host on a network. This address is in a block assigned to Excelan by the Xerox Corporation. For EXOS front-end processors, it is permanently stored in PROM on the processor board. This means that the Ethernet address of any host is the physical address of the EXOS front-end processor used in that host. The high-order three bytes of this address represent the manufacturer's identification code (also known as the address block); the low-order three bytes represent the host identification.

Normally, you will not need to reference an Ethernet address. The only situations when you will need to specify an Ethernet address will be when you want to add an entry to the ARP table (see Section 3.6.1) and when you want to emulate a different Ethernet address (see Section 3.6.3.4).

3.5.2. Logical-to-Physical Host Address Translation

The TCP/IP protocol software uses logical Internet addresses, while the network hardware uses the physical Ethernet addresses. Thus, in order to send a message from one host to another, the protocol software must translate the Internet address of the second host to its Ethernet address before transmitting the message to the network.

There are two methods to accomplish the Internet-to-Ethernet address translation:

- The ARP method
- The constant mapping method

The method to be employed is selected prior to downloading the protocol software to the EXOS 203 or EXOS 204 front-end processor by modifying the command line for the *netload* utility in the network startup command file (LB:[1,1]EXOSLOAD.CMD). To change the method, the command line must be modified and the protocol software again downloaded to the EXOS 203 or EXOS 204 front-end processor.

3.5.2.1. The ARP Method

The ARP method uses the Address Resolution Protocol. When IP is requested to send a message to an Internet address, it consults ARP for the associated Ethernet address. If ARP does not have this information in its cache, it first broadcasts a packet containing the target host's Internet address. All hosts on the network receive this packet. However, only the target host responds with its Ethernet address. On receipt of a response, ARP on the sending host saves the address translation in its cache and uses it for current and future communication with the target host.

No correlation between the Internet and Ethernet addresses is necessary to use the ARP method. However, it is necessary that either all hosts which are to communicate with each other support the Address Resolution Protocol (ARP) or one of the hosts contain in its Internet-to-Ethernet translation table a *publ* option for the non-ARP-supporting host (see Section 3.6.1).

See Section 3.6.3.6 for details on how to enable and disable ARP.

3.5.2.2. The Constant Mapping Method

When this method is used, IP on a local host creates an Ethernet address by concatenating the uppermost three bytes of its physical address (manufacturer's ID part) with the lowermost three bytes (host ID part) of the target host's Internet address. The local host obtains the target host's Internet address from the network address file.

This method can be used only for Class A networks, where the low-order three bytes of the Internet address of any host must equal the low-order three bytes of the the Ethernet address for that host. In addition, the uppermost three bytes of Ethernet address for each host must either be same or must be emulated to be same. See the description of the *netload* utility in Section 3.6.3 for a description of Ethernet address emulation.

3.5.3. Routing Across Networks

Communication between a host on a local network and a host on a remote network is established through gateways that physically connect the local network and the remote network. At the user interface level, the packets intended for a host on the remote network are simply sent to the target host. Internally, however, the target host's Internet address is mapped to the Internet addresses of the gateway associated with it and eventually mapped to the Ethernet address of the gateway. This means that when you send a packet to a host on a remote network, the software sends the packet to the Internet address of an intermediate gateway.

The gateway can be a regular host that also performs the gateway function or it can be a dedicated gateway that performs only the gateway function.

The gateway mapping information is maintained in the routing table (see Section 3.5.4.5.).

3.5.4. Network Database

The network database consists of the following files and tables. Each of these is described in the sections that follow.

- Network Startup Command File
- Network Address File
- Board Statistics Table
- Internet-to-Ethernet Translation Table
- Routing Table

3.5.4.1. The Network Startup Command File

The file LB:[1,1]EXOSLOAD.CMD is the Network Startup command file. It contains a command line that invokes the *netload* program. The *netload* program in turn downloads the TCP/IP object code to the EXOS front-end processor. Editing this file allows you to deselect several default options and/or select several new options. These options include manipulation of Internet and Ethernet addresses and enabling/disabling of the Address Resolution Protocol (ARP).

You must be a privileged user in order to execute the LB:[1,1]EXOSLOAD.CMD command file.

If new options are to be selected, the command line(s) in this file should be edited and then the command file re-executed to download the protocol software to the EXOS front-end processor.

The command line and the various options for the *netload* utility are described in Section 3.6.3.

3.5.4.2. The Network Address File

The file LB:[1,1]HOSTS.NET is the network address file. It contains mappings between the Internet addresses and the names and aliases for various hosts on the network. This file exists on all the hosts on the network. When a user references a host by name, the underlying application uses this file to translate the host name into an Internet address.

The mapping for each host is specified on a single line in the following format:

Internet_address system_name [*alias*] ...

Internet_address is four-byte value specified in decimal, octal, or hexadecimal using dot notation. See Section 3.5.1.2 for Internet address specification formats.

system_name is the name associated with the *Internet_address*. It can be any string up to 32 characters long. The first character must be an alphabetic character. *system_name* must be unique across the network. Uppercase and lowercase letters are considered equivalent. Embedded blanks are not permitted in name specifications.

alias is an alternate name for the host. Typically, it would be a shorter name. More than one alias for a given host is permitted. An *alias* need not be unique across the network.

The entry in the network address file for the local host must have the alias *localhost*. The *netload* utility uses this to define the Internet address of the local host, and the *ftp* utility requires this.

For example, the network address file for a host named "california" could have the following entries:

```
185.0.5.2 california ca localhost
185.0.5.4 oregon or og
185.0.5.2 mexico mex
127.0.0.0 loopback lb
185.0.5.10 finance fin
185.0.3.22 manufacturing manu
```

The Internet address 127.0.0.0 is a special address for the local EXOS front-end processor. Any communication sent to the host associated with this address is returned to the sender without ever getting on the network. Thus any communication transmitted to the host loopback will be received by "california." This is a useful feature for testing/debugging applications.

For communication among all the hosts, the network address file on each host must contain mappings for all other hosts on the network.

3.5.4.3. The Board Statistics Table

The board statistics table resides in the EXOS 203 or EXOS 204 processor memory. The statistics in this table are compiled and stored by the on-board firmware, NX. The table contains the following counts:

- Packets received from the network
- Packets transmitted to the network
- Packets received with alignment error
- Packets received with CRC error
- Packets lost because no buffers were available

The table also contains the version numbers for EXOS 8031 software and EXOS 203 or EXOS 204 hardware.

Anyone can display the statistics in this table using the *bstat* utility described in Section 3.6.2. A privileged user can also reset the board statistics.

3.5.4.4. The Internet-to-Ethernet Translation Table

The Internet-to-Ethernet translation table resides in the EXOS front-end processor memory. It is created and dynamically maintained by the Address Resolution Protocol (ARP). This table contains the Internet addresses of those hosts on the network that responded to individual ARP broadcasts and their corresponding Ethernet addresses. The table also contains unresolved host entries.

The entries in this table contain the following information:

Internet_address *Ethernet_address*

The following display shows examples of resolved and unresolved entries. This display is composed by the *arp* utility, which uses both the Internet-to-Ethernet translation table and the network address file for the purpose.

```
Munich  (0x59.0x60.0x0.01)  at  0x8:0x0:0x14:0x60:0x0:0x1
Munich  (0x59.0x60.0x0.01)  ---  incomplete
Munich  (0x59.0x60.0x0.01)  ---  no  entry
```

Anyone can display the entries in the ARP table using the *arp* utility. A privileged user can also add and delete entries from this table. Additions can be permanent or temporary. See Section 3.6.1 for a description of the *arp* system utility.

3.5.4.5. The Routing Table

The routing table resides in the EXOS front-end processor memory. The initial entries are based on the gateways available as a consequence of hardware/software configuration in the system. The table contains single-line entries for available routes in one of the following two formats:

host_Internet_address gateway_Internet_address
network_Internet_address gateway_Internet_address

In the above formats, *host_Internet_address* is the Internet address for the host on a remote network. Similarly, *network_Internet_address* is the Internet address for the remote network. *gateway_Internet_address* is the Internet address for the gateway on the local network, which is used when routing communication to hosts on remote networks.

Anyone can display gateway entries in the routing table using the *route* utility. A privileged user can also add and delete gateway entries. See Section 3.6.4 for a description of the *route* utility.

3.6. NETWORK SYSTEM UTILITIES

The EXOS 8031 TCP/IP software package includes six utility programs in addition to the protocol software. These utilities provide very important, often-used functions such as file transfer between hosts on the network, terminal emulation, and manipulation of data tables internal to the TCP/IP and the Ethernet front-end processor. These utilities are grouped into two categories:

1. Network Application Utilities (FTP, TELNET)

2. Network System Utilities (ARP, BSTAT, NETLOAD, ROUTE)

The Network Application Utilities, which are typically used by end users, are described in Chapter 4.

The Network System Utilities, which are typically used by network/system administrators, are described in a user's guide style in the subsections indicated below. These descriptions are not meant to be comprehensive. A detailed, formal description for each of these utilities is provided in Appendix A.

Utility	Name	Section
ARP	The Address Resolution Control Utility	3.6.1
BSTAT	The Board Statistics Utility	3.6.2
NETLOAD	The Protocol Software Load Utility	3.6.3
ROUTE	The Routing Control Utility	3.6.4

The *arp* utility allows a user to display the Internet-to-Ethernet address mapping for those hosts that have responded to the Address Resolution Protocol (ARP) broadcasts and other entries made manually by a user. It also allows a privileged user to add and delete mappings from the ARP cache.

The *bstat* utility displays the board statistics (packet traffic) for an EXOS front-end processor, which in effect means a local host. It also allows resetting the statistics and displaying the version numbers for the current TCP/IP software and the EXOS board hardware and firmware.

The *netload* utility performs the initial load of the TCP/IP software. At load time, this utility allows the configuration of several options.

The *route* utility allows examination, insertion, and deletion of gateway addresses for internetwork communication.

These utilities can be used by anyone to view current values or states, but only privileged users can alter the values or states.

3.6.1. ARP – The Address Resolution Control Utility

The *arp* utility displays and manipulates the entries in the Internet-to-Ethernet translation table generated by the Address Resolution Protocol (ARP). This table contains the Internet and Ethernet address mappings entered by a user (usually for non-ARP-supporting hosts) and for those hosts on the network that responded to individual ARP broadcasts. The table also contains unresolved host/address entries. (See Appendix A for a detailed, formal description of the *arp* utility.)

NOTE

Only privileged users can manipulate the ARP translation table.

Using the ARP Utility

You can display the current entry for a specific host (say munich) in the Internet-to-Ethernet translation table by entering the command shown below. Depending on whether the entry is a resolved entry, an unresolved entry, or a nonexistent entry, one of the three responses shown will be displayed. (Note that in order to include the host name in the response lines, the *arp* utility uses both the network address file and the Internet-to-Ethernet translation table to compose the response lines.)

```
> arp munich
munich (0x59.0x60.0x0.0x1) at 0x8:0x0:0x14:0x60:0x0:0x1
munich (0x59.0x60.0x0.0x1) incomplete
munich (0x59.0x60.0x0.0x1) no entry
```

You can display all the current resolved and unresolved entries in the translation table with the following command:

```
> arp -a
munich (0x59.0x60.0x0.0x1) at 0x8:0x0:0x14:0x60:0x0:0x1
india (0x59.0x10.0x1.0x81) at 0x8:0x0:0x14:0x10:0x1:0x81
london (0x59.0x1.0x33.0x65) at 0x8:0x0:0x14:0x40:0x1:0x89
```

By adding the keyword *entire* to the above command you can display the entire translation table, which includes resolved entries, unresolved entries, entries for which no corresponding entries in the network address file exist, and "empty" entries.

You must be a privileged user in order to exceute the following commands.

You can delete a host (say india) from the translation table by using the following command.

> **arp -d india**

The following command will insert an entry, consisting of a host name and the corresponding Ethernet address, in the ARP translation table. Normally, the entry will be permanent; if the option *temp* is used, the command makes the entry temporary. If the option *publ* is specified, the entry is "published," which enables a host to respond to ARP broadcast even if the specified Internet address is not its own address. (See Appendix A for an explanation of *temp* and *publ* options.)

> **arp -s** *host_name Ethernet_address* **[temp] [publ]**

You can insert multiple entries in the translation table by reading them from a text file using the following command:

> **arp -f** *filename*

The format for the file *filename* is as follows:

host_name Ethernet_address **[temp] [publ]**

3.6.2. BSTAT – The Board Statistics Utility

The *bstat* utility obtains packet-traffic statistics from the EXOS front-end processor. NX, the on-board firmware, compiles these statistics and stores them in the board memory. These statistics show the total number of packets received from and transmitted to the Ethernet. Also included in the statistics are counts for the following error conditions. (See Appendix A for a detailed, formal description of the *bstat* utility.)

- Packets received with an alignment error – the packets were not in 8-bit multiples
- Packets received with CRC errors
- Packets lost because no receive buffers were available

- Packets suffering from SQE (transceiver "heartbeat") test failures
- Transmissions that failed with DMA underrun

The utility displays the statistics since the last statistics reset or downloading to the EXOS front-end processor, whichever occurred most recently. *bstat* prints only those statistics that have nonzero values.

In addition to the above functions, the *bstat* utility can reset the statistics on the board and display the version numbers for the current TCP/IP software, NX firmware, and the EXOS board hardware.

NOTE

Only privileged users can reset the board statistics.

Using the BSTAT Utility

To display the statistics, enter the following command:

```
> bstat
```

The system responds with a display similar to the following one:

```
nnnn frames transmitted
nnnn SQE (transceiver heartbeat) test failures
nnnn transmissions failed with DMA underrun
nnnn frames received
nnnn frames received with alignment error
nnnn frames received with crc error
nnnn frames lost (no receive buffers)
```

To reset the statistics, enter the following command:

```
> bstat –r
```

The system then displays the current statistics. A subsequent *bstat* command does not display anything since the statistics have been reset (to zero) and *bstat* displays only nonzero statistics.

To display the version numbers for the current TCP/IP software running on the board, the NX firmware, and the EXOS board hardware, enter the following command. (System response is also shown.)

```
> bstat –v

EXOS:Firmware Release:4.4
EXOS:Hardware Release:0.0
EXOS:Software Release:3.2
```

3.6.3. NETLOAD – The Protocol Software Load Utility

The *netload* utility downloads the TCP/IP protocol object code portion of the EXOS 8031 software package to the local memory on the EXOS board. This utility is invoked from the command file LB:[1,1]EXOSLOAD.CMD.

The *netload* utility lets you configure several options in the TCP/IP code. These options, among other things, can display debug messages, assign a host's Internet address from the network address file, allow an Ethernet controller board to emulate a different board, and disable ARP.

These options are configured by specifying appropriate arguments in the command line that invokes the *netload* utility. This means that the command file which contains the *netload* invocation command must be edited. These options can be configured at the EXOS 8031 initial installation time or whenever the protocol software is downloaded to the EXOS front-end processor.

NOTE

Only privileged users can use this utility.

3.6.3.1. Using the NETLOAD Utility

As mentioned above, the *netload* utility is invoked from the command file LB:[1,1]EXOSLOAD.CMD. Therefore, in order to configure the options, the command line that invokes the utility must be edited. The *netload* command line format is as follows.

netload [**–d**] [**–h** *host*] [**–e** *enet_addr*] [**–t** *n*] [**–m**] [**–r** *n*] [**–x** *n*]
[**–p** *n*] [**–i**] [**–l**] [**–c***n*] [**–o**] [*net_file*]

If no options are specified, the *netload* command configures TCP/IP with the following defaults:

- Do not print debug messages.
- Use the Internet address for the host with the name or alias "localhost" in the hosts file.
- Use the Ethernet address supplied by the board's manufacturer.
- Enable the on-board TELNET-server program and support 16 connections.*
- Enable the Address Resolution Protocol (ARP).
- Load the TCP/IP protocol module from the file LB:[1,2]NET.

The following subsections describe the *netload* options.

3.6.3.2. Printing Debug Messages

The **–d** option prints useful debug information. It can facilitate tracing program flow, program debugging, and program maintenance.

3.6.3.3. Overriding the Default Host Address

The **–h** option allows you to override the default address for your own system and to specify a name or address for it.

*In the current release only eight connections can be supported.

The *host* parameter specifies the new host address or name. If this specification begins with a digit, *netload* attempts to convert it to an Internet address in the standard dot notation. Otherwise, it looks up the host by name in the network address file and uses the Internet address associated with it.

3.6.3.4. Emulating a Different Ethernet Address Block

The **–e** option allows for emulation of the address block of another manufacturer's Ethernet controller. In other words, your board can emulate a board from another manufacturer. (The *enet_addr* parameter represents the Ethernet address to be emulated.) This is useful when you are connecting to a network that mainly uses controller boards from a manufacturer other than Excelan and that does not employ ARP. For example, if the network you are connecting to mainly uses 3Com boards, your EXOS board can emulate the 3Com address block as follows:

–e 02-60-8C-E0-2A-56

The *enet_addr* must be specified, as shown above, in the Ethernet address format: six hexadecimal numbers (one or two digits, upper- or lowercase) separated by hyphens (-). The three high-order bytes identify the manufacture, the three low-order bytes identify the host.

3.6.3.5. Specifying Number of Connections to the TELNET Server

By using the **–t** option, you can specify the number of connections the on-board TELNET server program will support. This number is specified by the *n* parameter which can be any number in the range 0 to 8. A 0 specification disables the TELNET server program.

3.6.3.6. Enabling and Disabling the Address Resolution Protocol

By default the Address Resolution Protocol (ARP) is enabled. The **–m** option disables ARP.

When ARP is enabled (default), it allows communication among Ethernet controller boards manufactured by different vendors. This protocol dynamically maps Internet addresses to Ethernet addresses on a local area network. On receiving a request for a mapping of a previously unknown host address, ARP broadcasts a message across the Ethernet for that host to respond. When a response is received, ARP caches the physical address of the host for use in future communication.

If ARP is disabled, TCP/IP forms the Ethernet address of the target host by concatenating the three high-order bytes of the local system's Ethernet address with the three low-order bytes of the target system's IP address. This assumes that the network uses Class A Internet addresses.

Normally, the hosts employing the ARP protocol can establish communications only with those hosts that themselves support the ARP protocol. However, if the address cache of an ARP-supporting hosts contains an entry for a non-ARP-supporting host with the *publ* option, then all ARP-supporting hosts can communicate with the non-ARP-supporting host. Refer to Section 3.6.1 for a description of the *arp* utility and its *publ* option.

3.6.3.7. Specifying the Board Resources

The **–r** option allows you to specify the number of host requests that are to be supported simultaneously. This number should be at least as large as the maximum expected number of connections; it must not be larger than 127. Note that the larger this number, the smaller the space for data buffering.

The **–x** option allows you to specify the number of extended memory buffers that are allocated. A large number of memory buffers can improve bulk transfer performance but may affect operation when more than 32 connections are required. A minimum of 10 buffers are always allocated even if the number specified is less than 10.

The **–p** option sets the level of diagnostic messages from the board that the host displays. The level is determined by the value of *n*, as follows:

Level	Types of Messages
1	All
2	Errors that are benign (such as retransmissions) or more severe (default)
3	Errors that are nonfatal (such as bad host-requests) or more severe
4	Errors that are fatal, such as NX call failures

The **–i** option disables the timeout feature of *netload*. Normally, *netload* times out one minute after the board is reset and the diagnostics are unsuccessful or if an initialization message to the board is unsuccessful. The **–i** option is useful for debugging the board's processor or when using an emulator.

3.6.3.8. Specifying the Protocol Software File

By default, the *netload* utility expects to find the TCP/IP object module in the file LB:[1,2]NET. However, if you should prefer to load it from a different file, all you need to do is provide the name of that file as the *net_file* parameter in the command line.

3.6.4. ROUTE – THE ROUTING CONTROL UTILITY

The *route* utility manipulates the routing table in the EXOS board memory. The utility can add a new route, delete an existing route, and show an individual route or all available routes.

See Appendix A for a detailed, formal description of the *route* utility.

The initial entries in this table are based on the gateways available as a consequence of hardware/software configuration in the system. These table entries consist of Internet address for a host on a remote network (or for the remote network) and the associated gateway for communication with that host or network.

NOTE

Only privileged users can alter the routing table.

Using the ROUTE Utility

The *route* utility has three options: *add*, *delete*, and *show*. The format for using these optins is as follows:

route add *destination gateway*
route delete *destination gateway*
route show [*destination*]

In the formats shown above *destination* is the host on the remote network or the remote network itself and *gateway* is the gateway to which the packets should be addressed for eventual delivery to the host on the remote network. Both *destination* and *gateway* can be specified as names or as Internet addresses. Internet addresses must be specified in the same class as the network class.

Let us say you need to enter a new route – the destination network is "newyork" and the gateway is "ny." You can accomplish this with the following command. (System response to the command is also shown.) Note that the logical host, network, and gateway names are translated into Internet addresses, shown in four-part dot notation, in the system responses.

> **route add newyork ny**
add network 29.0.0.0 gateway 89.1.51.101

You can delete the above entry with the following command:

> **route delete newyork ny**
delete network 29.0.0.0 gateway 89.1.51.101

If "newyork" were a host on a remote network instead of a network with the Internet address (say) 29.4.5.6, the above input commands would be identical for the indicated tasks, but the system responses would be as follows:

add host 29.4.5.6 gateway 89.1.51.101
delete host 29.4.5.6 gateway 89.1.51.101

You can see the route for a specific remote host or network (say "ca") by entering the following command:

> **route show ca**
show network 27.0.0.0 gateway 89.1.51.101

You can see routes for all the remote hosts and networks supported by the local network by entering the following command:

> **route show**
show host 27.1.2.3 gateway 89.1.51.101
show network 29.0.0.0 gateway 89.1.65.201
show host 27.3.3.3 gateway 89.1.51.101
.
.
.

Chapter 4
NETWORK APPLICATION UTILITIES

4.1. INTRODUCTION

The EXOS 8031 TCP/IP software package includes two network application utilities, in addition to four network systems utilities and the protocol software. The application utilities provide two important, often-used functions: file transfer between hosts on the network and terminal emulation.

The two network application utilities are

- The File Transfer Utility (FTP)
- The Virtual Terminal Utility (TELNET)

The following sections describe these utilities in a user's guide style. Appendix A provides a detailed, formal description for these utilities.

4.2. FTP – THE FILE TRANSFER UTILITY

The *ftp* utility is a client program that uses the DARPA Internet standard File Transfer Protocol (FTP) to transfer files between a local host and a remote host.

In addition, *ftp* can access directories/files on a remote host and allow a user to perform usual operations, such as list and change working directories, list files at various levels, and rename directories and files. Naturally, to perform these operations, a user must be logged in to the remote system.

While the local system is (obviously) a RSX-11M-based system, the remote host can be any system – RSX-11M-based or otherwise – that supports the FTP protocol. When communicating with a remote host that is also running the EXOS 8031 software, the remote host's FTP-server program *ftpd* is the responding entity. Most implementations of TCP/IP support the FTP protocol.

Prerequisites for using *ftp* include the following:

- Both the local and remote host support ARPANET's standard File Transfer Protocol.
- The FTP server program is running on the remote host.
- File specification for the remote host is done according to the remote host's conventions.

All *ftp* operations are performed by first invoking the utility and then executing various *ftp* commands.

Section 4.2.1 describes the *ftp* invocation and remote login procedure. Section 4.2.2 gives examples of usage of some of the commands. Appendix A gives a comprehensive, formal description of the *ftp* utility, its options, and commands. Appendix A also provides a formal description of the FTP-server program *ftpd*, which is supplied as part of the EXOS 8031 TCP/IP software package. The FTP-server program description is included for completeness; normally, a user or even a system administrator will not need to use it explicitly except for ensuring that it is running on the systems that are to be accessed by other systems for file manipulation.

4.2.1. Invocation of FTP and Remote Login

To illustrate the invocation of *ftp* and login to a remote host, let us assume that a user "dave" at local system "finance" needs to login to remote host "warehouse." "dave" can login as himself (by entering his name) or he can login as any other user (let us say "mark"). In either case, the user must be a recognized user of the remote host. Furthermore, the user should provide the password if required by the remote host. (The password requirements are set by the remote host, which may or may not require typing in the password.)

"dave" can invoke the *ftp* and login to the remote host "warehouse" by using one of the two methods shown below as command sequences. The numbers indicate the response from the server.

Method 1:

> **ftp**
> ftp> **open warehouse**
> Connected to warehouse.
> 2xx Warehouse FTP Server (<version> <date>) ready.
> Remote user name: **mark**
> 331 Password required for mark.
> Password: *password*
> User mark logged in.
> ftp>

Method 2:

> **ftp warehouse**
> Connected to warehouse.
> 2xx Warehouse FTP Server (<version> <date>) ready.
> Remote user name: **mark**
> 331 Password required for mark.
> Password: *password*
> User mark logged in.
> ftp>

In both cases the local system enters the *ftp* command mode.

In Method 1, the command mode simply displays the "ftp>" prompt. Then, in response to the user input **open warehouse**, the command mode displays the "Name" prompt line.

In Method 2, the command mode implicitly interprets and executes the "**open warehouse**" command and then displays the "Name" prompt line.

In response to the "Name" prompt, "dave" could have entered his own name. In the present case, however, dave enters **mark**. The system then displays the "Password" prompt line. After dave enters the password, the system returns to the command mode, displays the prompt "ftp>," and awaits further user input.

4.2.2. Using FTP Commands

After you have logged in to a remote host as described above, you can use *ftp* commands for such operations as change working directories, list directories,

remove files, rename files, and copy/append files. This section illustrates the use of only a few of the *ftp* commands. Table 4-1 lists all *ftp* commands by function. A comprehensive, formal description of all the *ftp* commands is provided in Appendix A.

NOTE

All *ftp* commands are interactive. That is, if you enter a command without the required parameter(s), the system prompts you for the parameter(s).

The *help* command provides a brief description of each *ftp* command. It can also be used for listing all the available *ftp* commands. The format for using the *help* command is as follows:

ftp> **help** *command*
ftp> **help**

The first command displays information for the specified *command.* The second command lists all the available *ftp* commands.

Some commands simply set or reset various *ftp* options in a toggle fashion. That is, if an option is on, executing the relevant command sets the option to off. Executing the command a second time resets the option to on. For example, when you first login, the *verbose* option, which gives detailed messages is on by default. You can turn the *verbose* option off by typing the command

ftp> **verbose**

You can reset the verbose option to on by executing the above command one more time.

The state of all *ftp* options at any given time can be displayed by typing the command

ftp> **status**

The above command displays the options status in the following format:

Mode: Stream; Type: ascii; Form: non-print;
Structure: file; Verbose: on; Bell: off; Prompting: off

Table 4-1: FTP Commands Grouped by Function

Group/ Function	Command
Invoking/Quitting	
Establish connection to remote host's server	open
Terminate FTP session, but remain in FTP command mode	close
Terminate FTP session and return to operating system	bye, quit
Directory Operations	
Change current working directory	cd
Copy remote directory information to local file	dir
Change current working directory to local directory	lcd
Display contents of local directory	ldir
Display short form of directory information for local directory	lls
Display current local working directory	lpwd
Display short form of directory information for remote directory	ls
Write directory information for remote file(s) into local file	mdir
Create remote directory	mkdir
Write directory information for remote files to local file	mls
Display name of current remote directory	pwd
Delete remote directory	rmdir
File Operations	
Delete remote file	delete
Delete remote file(s)	mdelete
Rename remote file	rename
File Transfer	
Append local file to remote file	append
Copy remote file to local file	get, recv
Place remote file(s) into local working directory	mget
Copy local file(s) to remote working directory	mput
Copy local file to remote working directory	put, send

Table 4-1: FTP Commands Grouped by Function (Continued)

Group/ Function	Command
File Transfer Parameters	
Set file transfer form	form
Set file transfer mode	mode
Set file transfer structure	struct
File Transfer Data Format	
Set file transfer type to ASCII	ascii
Set file transfer type to binary	binary
Display or set file transfer type	type
Modifying FTP Environment	
Sound bell on completion of each command	bell
Toggle debugging mode	debug
Toggle interactive prompting	prompt
Toggle verbose mode	verbose
Miscellaneous Commands	
Toggle filename globbing	glob
Display on-line help documentation	help, ?
Send FTP command to server	quote
Display help documentation from remote host	remotehelp
Toggle use of port commands	sendport
Display current status of FTP options	status
Identify a user to remote host	user

4.2.2.1. Manipulating Files

The main function of the *ftp* utility is to transfer files between the local system and a remote host. However, it allows you to perform several additional operations such as list/change directories and/or list files of the remote host before actually transferring the files.

The following command lists all the directories and files in the current working directory of the remote host:

ftp> **dir**

You can change the current working directory on the remote host with the following command:

ftp> **cd** *directory*

You can change the working directory on the local system by using the following command. This command works whether or not you have established a connection with the remote server.

ftp> **lcd** *local_directory*

If *local_directory* is not specified, the current working directory changes to your home directory.

get and *put* are the two basic commands for transferring files between the local and the remote host. The *get* command copies files from the remote host to the local system. The *put* command copies files from the local system to the remote system. The format for using these commands is as follows:

ftp> **get** *remote_file local_file*
ftp> **put** *local_file remote_file*

The *append* command appends a local file to a remote file. The syntax for this command is as follows:

ftp> **append** *local_file remote_file*

If *remote_file* is omitted from the *put* and *append* commands, the local file is copied to the remote system under its original name.

4.2.2.2. Logging Out

Once you have finished the *ftp* session with a given remote system, you can either break the connection with (logout of) the remote system while still remaining in *ftp* or you can logout, exit *ftp*, and return to the operating system on the local system.

To logout of the remote system while staying in the *ftp* enter the following command. After this, you can login to another remote system without having to re-invoke the *ftp* utility.

ftp> **close**

To logout, exit the *ftp*, and return to the operating system, enter the following command:

ftp> **quit**

4.3. TELNET – THE VIRTUAL TERMINAL UTILITY

The *telnet* utility lets your system emulate a virtual terminal connected to a remote host. You can "connect" to any remote host on the network that supports the TELNET protocol and perform all operations as if you were using a terminal physically connected to the remote host.

4.3.1. Invocation of TELNET and Logging In

To initiate a virtual terminal connection, you first invoke the *telnet* utility and then establish a connection to the remote host. This requires execution of two simple commands. Alternatively, you can perform this operation by executing a single command that includes *remote_host* as an argument.

The two methods to "connect" your system to a remote host are given below as command sequences.

Method 1:

> **telnet**
(to) *remote_host*
Connected to <*remote_host*>
Escape character is "^]"

Method 2:

> **telnet** *remote_host*
Connected to <*remote_host*>
Escape character is "^]"

In Method 1, the first command invokes the *telnet* utility; the second command connects to *remote_system.*

In Method 2, the command invokes the utility and connects to *remote_host.*

Following connection to the remote host, you can login to the remote host in the normal manner: enter the user name and, when prompted, enter the password. After that you can proceed as though you were directly connected to *remote_host.* Of course, to perform these operations, you need to use the syntax conventions native to *remote_host.*

4.3.2. Using TELNET Commands

This section demonstrates the use of some of the *telnet* commands. comprehensive, formal description of the TELNET utility and its commands is provided in Appendix A.

telnet commands can be executed only when *telnet* is in command mode. (This mode is indicated by the presence of the "telnet>" prompt.) *telnet* can be forced into command mode by entering the current escape character on the command line. The default escape character is "^]" (CTRL-]), which means, while holding the "CTRL" key, press the "]" key. An example of forcing the *telnet* into command mode is shown below:

```
^ ]
telnet>
```

You can find out the name of the remote host you are connected to by using the *status* command, as follows:

```
^ ]
telnet> status
Connected to <remote_host>
Escape character is '^ ]'
```

You can change the escape character by using the *escape* command, as follows. Let us assume that you want to change the default escape character ^] to ^ A.

```
$ ^ ]
telnet> escape
new escape character: ^ A
Escape character is '^ A'
$
```

You can display on-line help documentation by using the "?" command as illustrated below. Let us assume you want information about the *status* command.

```
$ ^ ]
telnet> ? status
Print status information
telnet>
```

If the "?" command is entered without any argument, *telnet* lists all its available commands.

After *telnet* executes a command, you are returned to the host operating system, but the operating system prompt is not displayed. The exception is the "?" command: after executing this, the "telnet>" prompt is redisplayed.

4.3.3. Logging Out

Once you have finished the *telnet* session with a given remote host, you can break the connection with (logout of) the remote host and return to the operating system on the local host by entering the following command:

```
telnet> quit
```

Note that unlike the other TELNET servers, the 8031 TELNET server does not automatically break the connection after you logout from the RSX-11M system. You need to enter the *quit* command to break the connection.

Chapter 5
PROGRAMMING INTERFACE

5.1. INTRODUCTION

The QIO (queue input/output) programming interface is a set of system calls that allows an application program to access the network. Specifically, the QIO system calls allow communication between an application program running on the host system and the EXOS 8031 TCP/IP software running on the Ethernet front-end processor. QIO calls are passed to the EXOS device driver via the operating system. The driver, after interpreting and translating the calls, passes them to the EXOS board, which executes the requested I/O operation.

This chapter provides a general discussion of applications and sockets and discusses the procedures for invoking QIO function calls. At the end of the chapter is a detailed description of each call. Also given are several functions showing how to use the QIOs.

5.2. APPLICATIONS AND SOCKETS

A typical network application usage involves two distinct programs: a client program and a server program. A *client program* runs on the user's system (called the local host). It executes only when invoked by the user. A *server program* runs on a remote system (called the remote host); it is this program that the client program communicates with. The server program executes at all times, alert for service requests.

A *socket* is an end point for communication between client and server programs. Two sockets that are connected to each other (they normally reside on different hosts) form a single, full-duplex data stream. Sockets can be created and manipulated through the QIO programming interface. There are four types of sockets:

- Stream sockets. These provide sequenced, potentially reliable, two-way, connection-based streams with an out-of-band mechanism.
- Datagram sockets. These provide for connectionless, potentially unreliable messages of a fixed maximum length (typically a single packet) for user datagrams.
- Raw sockets. These provide access to internal network interfaces.
- Link-level sockets. These provide direct Ethernet I/O, bypassing TCP/IP.

A typical application that sends and/or receives data uses the following I/O functions in the listed sequence:

- Create a socket
- Connect to a socket or accept a connection request to a socket
- Read/write (and/or send/receive)
- Close the connection

When a user invokes the network application, the client program establishes a local socket. It then attempts to connect to a remote socket, already created by the server program on the target system, by issuing a connect request. The server program, which is waiting for a connection request, accepts it. At this

point a communications channel has been established. Using read and write function calls, data can be transferred between the local and remote hosts. Many concurrent conversations can be in progress. When communication on a channel concludes, the socket is closed.

Note that an application typically specifies the host name in symbolic form, but the QIOs require the Internet address. The program *rhost* performs this conversion. A call of the form

```
rhost (host_name)
char **host_name;
```

returns the 32-bit Internet address in network byte order.* If *host_name* is unknown, *rhost* returns -1. If the host name is known, **host_name* is replaced by the standard name of the host. Storage for the name is obtained by the *malloc* C library function call. This function operates by reading the network address file, which is described in Section 3.5.3.2. *rhost* is a user-level subroutine, and the C language source code for it is provided in the 8031 software package. This function can be used as provided, it can be converted or modified, or a different mechanism can be used to translate symbolic host names to their Internet addresses.

5.3. QIO SYSTEM CALLS

The QIO system calls provide programming access to the network and its sockets. The calls invoke system subroutines that direct the device driver to create and delete sockets, to establish connections between sockets, and to pass data between sockets.

5.3.1. The EXOS 8031 Device Driver

The EXOS 8031 driver supports a set of QIO functions that obtain specific services from the EXOS Ethernet front-end controller board. The EXOS device is a single sharable device refered to as "ZE0" that services all the processes. It is modeled as a multiple channel device where each channel is a path for communication with the front-end processor. User tasks must open one channel for each communication path and close it whenever it is no longer needed.

Before requesting any services from the driver, the application program must create a channel either with an open administrative channel call or with a socket open call. The driver returns a channel number (if a channel was successfully opened), which is specified in all subsequent calls to that channel. I/O requests without a proper channel number lead to an error condition. It is the responsibility of the program to close all channels it opened, before exiting. However, the system closes all channels remaining open when the task exits.

There are two basic sets of operations that are performed on the EXOS front-end processor:

*The most significant byte of 16-bit values is stored at the lowest memory address.

- Administrative operations – These are related to control operations on the EXOS front-end processor. They include the following:
  - Open an administrative channel
  - Initialize the EXOS front-end processor
  - Download (write to EXOS memory) protocol software to the front-end processor
  - Start execution of the protocol software
  - Close the channel
- Network operations – These are related to communicating with a peer program across the network and include the following:
  - Open a channel and associate a socket (an end point for network communication) with it
  - Accept a connection from or connect to a remote socket
  - Read/Write data on a connection
  - Close the connection and release the channel

The QIO request to the EXOS driver must specify the above services by providing a function code for the individual services or by providing a major function code and a function-modifier pair.

5.3.2. Issuing QIO Requests

Application programs (also called tasks or user tasks) issue I/O requests to logical units that have been previously associated with a physical device. Each program or task can establish its correspondence between a physical device and a logical unit number (LUN) either when the task is built (using the ASG option) or dynamically at run time (using the ALUN$ [Assign Logical Unit Number] system directive).

The user task performs I/O by submitting a request for I/O service using the QIO or QIOW system directive. (For access to the EXOS front-end processor the user task must assign a logical unit number to the EXOS front-end device "ZE0"). For each directive, the executive determines the appropriate device driver to service the request based on the physical device specified in terms of LUN.

In the case of the QIO directive, the executive then queues the request for the driver; it does not wait for the I/O to complete before continuing.

In the case of the QIOW directive, the task waits for completion of the I/O before continuing. I/O completion is indicated when the event flag (specified in the QIOW request) is set. The QIOW directive thus combines the functions of the QIO and WTSE (Waitfor) system directives. The QIOW directive or the QIO and WTSE directives together should be used for synchronization.

The format of the QIO system directive is as follows:

QIO$ fnc,lun,[efn],[pri],[isb],[ast],[<P1,P2,P3,P4,P5,P6>]

The QIOW system directive has the same arguments as the QIO directive. The options and parameters are described in the next section.

5.3.2.1. QIO Options and Parameters

This section briefly describes the options and parameters of the QIO and QIOW executive directives. For more details, refer to the *RSX-11M Executive Reference Manual*.

The first six arguments in these system directives – *fnc, lun, efn, pri, isb,* and *ast* – are function-independent. The last six arguments – parameters P1 through P6 – are function-dependent. Items in brackets ([ ]) are optional. *lun* (channel number) and *fnc* (function name) must be specified in all requests.

The directive status word *$dsw* is a return status code that indicates whether the I/O operation has been successfully queued to the device driver.

fnc is the I/O function to be performed. It consists of a function code and a function modifier, which together are 16 bits. These are summarized in Table 5-1 and detailed at the end of this chapter.

NOTE

The I/O functions IO_ATT (attach device) and IO_DET (detach device) should not be used. If they are used, no other program, including the TCP/IP utilities, can access the EXOS hardware.

lun identifies the device on which the I/O is to be done. For the EXOS 8031 it should be assigned to ZE0.

efn is the number of the event flag to be set when the I/O request completes. *efn* is cleared when the QIO request is issued. The event flag is set asynchronously when the I/O operation requested by the QIO completes; it can thus be used for synchronization.

pri is the priority. This field is ignored, but a zero must be present in its place.

isb is the address of the I/O status block (IOSB). The system posts the completion status of the operation at this location. This is the only place where an I/O completion status is returned. The IOSB is cleared when a new QIO request is issued and is set asynchronously with the final I/O status when the I/O completes.

The IOSB consists of two two-byte words:

- The low-order byte of the first word indicates whether the I/O operation completed successfully. The codes returned in this byte are mostly standard RSX, device-independent error codes.
- The high-order byte of the first word contains a one-byte error code returned by the EXOS front-end processor.
- The second word contains either a count of the number of bytes transferred (for read/write operations) or the channel number (for an open call that results in the creation of a channel).

ast is the entry point address of an AST (asynchronous system trap) procedure that is to be executed asynchronously when the I/O completes. This allows interruption of the normal application execution to execute special code when the I/O completes. Even if the process is blocked for a QIOW executive directive, the process is interrupted.

Table 5-1: QIO Function Calls

Group/ Request Description	Function Call	Function Modifier	Section
Kill all outstanding I/O	IO_KIL	none	5.4.9
Administrative Operations			
EXOS board I/O control operations	IO_EXC		
Reset and configure EXOS	IO_EXC	EX_INI	5.4.16
Open administrative channel	IO_EXC	EX_OPN	5.4.18
Close administrative channel	IO_EXC	EX_CLS	5.4.11
Position EXOS memory locator	IO_EXC	EX_POS	5.4.19
Start execution of EXOS process	IO_EXC	EX_STR	5.4.23
Read EXOS statistics	IO_EXC	EX_STS	5.4.24
Read and reset EXOS statistics	IO_EXC	EX_RST	5.4.20
Read configuration message	IO_EXC	EX_CNF	5.4.12
Add routing table entry	IO_EXC	EX_ART	5.4.10
Delete routing table entry	IO_EXC	EX_DRT	5.4.14
Fetch routing table entry	IO_EXC	EX_SRT	5.4.22
Fetch next routing table entry	IO_EXC	EX_NRT	5.4.17
Set ARP table entry	IO_EXC	EX_SAR	5.4.21
Retrieve ARP table entry	IO_EXC	EX_GAR	5.4.15
Delete ARP table entry	IO_EXC	EX_DAR	5.4.13
Read from EXOS memory	IO_RLB	none	5.4.25
Write to EXOS memory	IO_WLB	none	5.4.27
Data transfer operations	IO_XFR		
Send datagram to remote socket	IO_XFR	IX_SND	5.4.30
Write data to TCP stream	IO_XFR	IX_WRS	5.4.31
Receive message from socket	IO_XFR	IX_RCV	5.4.28
Read data from TCP stream	IO_XFR	IX_RDS	5.4.29

Table 5-1: QIO Function Calls (Continued)

Group/ Request Description	Function Call	Function Modifier	Section
Network Operations			
Socket access operations	IO_ACS		
Open a socket	IO_ACS	SA_OPN	5.4.4
Accept connection from remote socket	IO_ACS	SA_ACC	5.4.1
Connect to remote socket	IO_ACS	SA_CON	5.4.3
Obtain socket address	IO_ACS	SA_SAD	5.4.5
Close a socket	IO_ACS	SA_CLS	5.4.2
Check for I/O readiness	IO_ACS	SA_SEL	5.4.6
Receive urgent signal	IO_ACS	SA_URG	5.4.7
Unselect a socket	IO_ACS	SA_USL	5.4.8
Socket control operations	IO_SOC		5.4.26
Show keep-alive status (SIOCGKEEP)	IO_SOC	SO_GKP	5.4.26
Enable/disable keep-alives (SIOCSKEEP)	IO_SOC	SO_SKP	5.4.26
Enable/disable lingering (SIOCSLINGER)	IO_SOC	SO_SLG	5.4.26
Show lingering status (SIOCGLINGER)	IO_SOC	SO_GLG	5.4.26
Set process group (SIOCSPGRP)	IO_SOC	SO_SPG	5.4.26
Get process group (SIOCGPGRP)	IO_SOC	SO_GPG	5.4.26
Receive out-of-band data (SIOCRCVOOB)	IO_SOC	SO_ROB	5.4.26
Send out-of-band data (SIOCSNDOOB)	IO_SOC	SO_SOB	5.4.26
Distinguish urgent data (SIOCATMARK)	IO_SOC	SO_AMK	5.4.26
Halt socket I/O (SIOCDONE)	IO_SOC	SO_DON	5.4.26
Enable/disable nonblocking I/O (FIONBIO)	IO_SOC	SO_NBO	5.4.26
Return count of data in socket's receive buffer (FIONREAD)	IO_SOC	SO_NRD	5.4.26

The parameters P1 through P6 are function-dependent. In general they contain the following information:

- P1 – address of data buffer
- P2 – length of data buffer, in bytes*
- P3
  - address of socket I/O control (SOioctl) data structure, which contains socket-related information for connection setup and data transfer operations, or
  - address of parameters related to the control operations that modify socket characteristics
- P4 – used for administrative operations
- P5 – used for administrative operations
- P6 – channel number (created by the IO_ACS | SA_OPN or IO_EXC | OPN call)

For parameter P3, several of the I/O function calls require the address of the SOioctl (socket I/O control) data structure. This structure contains socket-related information (both input and output values) for setting up connections and transferring data. This information is dependent on specific function calls, and can convey input parameters and store results of a socket control operation. The SOioctl structure and related data structures are discussed in the next section.

For several other I/O function calls, P3 contains the address of a parameter block related to control operations that alter the characteristics of a socket. The parameter block conveys both the input parameters as well the results of a socket control operation. For example, it is used to specify and/or examine the keep-alive time on a network connection.

5.3.2.2. Data Structures

This section discusses the SOioctl data structure and other data structures that are used with the EXOS 8031. The fields of all data structures must be contiguous.

The SOioctl data structure is shown in Table 5-2.

* For the EXOS 8032 the maximum length of the data buffer can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.

Table 5-2: SOioctl Data Structure

Field	Length	Comments
Address flag	2 bytes	Nonzero if socket address is specified
Socket address	16 bytes	Address of the socket
Protocol flag	2 bytes	Nonzero if socket protocol is specified
Socket protocol	4 bytes	Underlying protocol
Socket type	2 bytes	Type of socket
Options	2 bytes	Socket options

The SOioctl fields are discussed below. In the discussion, the fields are grouped by related function: address flag and socket address are discussed under "Address"; protocol flag, socket protocol, and socket type are discussed under "Protocol"; and the options are discussed under "Options."

Depending on the function call, an SOioctl field may contain an input or output value. Input values are those specified by the application program. Output values are those returned to the application when the QIO request completes. The usage of each value is listed in the "Parameter" section of each call's detailed description, given at the end of this chapter.

Address

The *address flag* field is set to a nonzero value if the socket address field contains a socket address to be used or if the application wants an address to be returned.

The *socket address* field can contain data only if the address flag is nonzero. If the address flag field is zero, no socket address is expected. The socket address data structure varies depending on the protocol being used. Its three possible variations are shown in Table 5-3.

Table 5-4 lists some common TCP and UDP port values that can be specified with the Internet port number.

Table 5-3: Socket Address Data Structure

Variant/ Field	Length	Comments
First variant – Internet protocols		
Internet family	2 bytes	Address family is 2
Port number	2 bytes	See Table 5-4
Internet address	4 bytes	In network data order*
Not used	8 bytes	Must be zero
Second variant – Link-level protocol		
Link-level family	2 bytes	Address family is 13
Ethernet type	12 bytes	Up to six nonzero Ethernet type values in network data order*, starting at the beginning of the array; high-order unused bytes contain zeros
Not used	2 bytes	Must be zero

*For TCP/IP, network data order means that the most significant byte of 16-bit integer values are stored at the lowest memory address. The EXOS board sends addresses in network data order and expects to receive them the same way.

Table 5-4: TCP and UDP Port Numbers

Symbolic Name	Value	TCP/UDP	Meaning
IPPORT_FTP	21	TCP	File Transfer Protocol
IPPORT_TELNET	23	TCP	TELNET protocol
IPPORT_SMTP	25	TCP	Simple Mail Transfer Protocol
IPPORT_TFTP	69	TCP	Trivial File Transfer Protocol
IPPORT_LOGINSERVER	513	TCP	UNIX rlogin protocol
IPPORT_CMDSERVER	514	TCP	UNIX rsh, rcp protocols
IPPORT_WHOSERVER	513	UDP	UNIX rwho protocol
IPPORT_RESERVED	1024	TCP/UDP	Lowest unprivileged port

Protocol

The *socket type* and *socket protocol* (in Table 5-2) together specify the protocol to be used. The socket types that are currently defined are listed in Table 5-5.

Table 5-5: Socket Type Values

Type	Value	Meaning
Stream	1	For a reliable, sequenced, two-way connection-oriented virtual circuit
Datagram	2	For connectionless, potentially unreliable messages of a fixed maximum length (typically small)
Raw	3	For access to internal network interfaces (not discussed further in this manual)
Link	5	For complete control of Ethernet packet content, permitting host-based support of non-TCP/IP protocols (see Section 5.3.2.3)

The *protocol flag* and *socket protocol* fields should be set to zero.

Options

The SOioctl *options* (see Table 5-2) are available with the socket function call (EX_SOCKET). The option values (or their symbolic names) listed in Table 5-6 can be specified. Options can be combined by adding their values.

Table 5-6: Socket Request Options

Symbolic Name	Value	Meaning
SO_DEBUG	0x01	Enable recording of debugging information by the host system.
SO_ACCEPTCONN	0x02	Enable a stream-type socket to accept a connection. If this option is specified, the socket can only accept a connection; it cannot initiate one. If it is not specified, only initiation is permitted.
SO_DONTLINGER	0x04	Disable "lingering" on a stream-type socket. When a socket close is requested, the connection is reset and operations on a socket complete immediately. If lingering is enabled, a close request blocks until all buffered data are transmitted.
SO_KEEPALIVE	0x08	Keep connections alive. This option enables the periodic transmission of messages on a connected but idle stream-type socket to check that the remote host is still functional. Keep-alive packets are transmitted on an established connection that has been idle for longer than one minute. If there is no response from the remote system within four minutes, the connection is aborted. Subsequent service requests on the socket return an error code. If a connection is not established and is idle for one minute, it is aborted. If this option is not selected, timing out dead connections is the responsibility of the user process.
SO_DONTROUTE	0x10	Indicates that outgoing Internet messages should bypass the standard routing facility. Messages are instead directed to the appropriate network interface based on the network portion of the destination address.
SO_SMALL	0x20	Cause TCP to use 1/2K windows. This is typically used for low-throughput utilities.

5.3.2.3. Link-Level Access

Concurrent link-level access gives the host low-level Ethernet access while TCP/IP is running. To do this, link access must be enabled by specifying the *-l* option of the *netload* command when the TCP/IP protocol is downloaded to the EXOS board. (See Appendix A for a more detailed discussion of this option.) An application program can access the link level by opening a "link" socket with the socket function call (Task and Terminal Privilege is required to use this call in this way). The link-level socket address structure, the second variant shown in Table 5-3, specifies the input packet filtering to be applied to the socket.

An application program can perform two operations: send (IO_XFR | IX_SND) and receive (IO_XFR | IX_RCV). Instead of pointing to a datagram message, the buffer pointed to by parameter P1 contains the entire Ethernet packet, beginning with the six-byte destination and ending just before the frame check sequence. The allocated buffer space must be large enough to accommodate the largest Ethernet packet (1514 octets [bytes] – 1500 bytes of data and 14 bytes of header). The address flag and socket address in the SOioctl structures of IO_XFR | IX_SND and IO_XFR | IX_RCV are unused and should be set to zero.

5.3.3. Error Handling

The EXOS 8031 device driver reports error status information in the first word of the IOSB associated with the QIO function call.

The low-order byte of the IOSB's first word contains error codes from the executive and/or driver. These errors concern the validity of the QIO function call and indicate whether the queuing request was successful.

The high-order byte contains error codes from the EXOS board. These concern the actual execution of the I/O request.

RSX returns queuing request error codes in the Directive Status Word ($DSW).

Refer to Appendix D for more details on QIO error status codes.

5.4. QIO FUNCTION CALLS

This section provides a detailed description of each QIO function call. Each description consists of the following parts:

- Description – explains the purpose of the function call.
- Parameters – defines the optional parameters (P1 through P6) that are specific to the function call.
- Privilege restrictions – notes any user privileges required to request the function.
- Notes – contains additional information about the function call.

Table 5-7 illustrates the typical order of function calls in a normal network application that uses QIOs.

Table 5-7: Typical QIO Function Call Order

I/O Request	Function Call	
	Client Program	Server Program
Create end point for communication	IO_ACS | SA_OPN	IO_ACS | SA_OPN
Initiate conversation	IO_ACS | SA_CON	IO_ACS | SA_ACC
Transfer data	IO_XFR | IX_SND IO_XFR | IX_RCV .	IO_XFR | IX_RCV IO_XFR | IX_SND .
	.	.
	.	.
Conclude	IO_ACS | SA_CLS	IO_ACS | SA_CLS

5.4.1. ACCEPT CONNECTION FROM REMOTE SOCKET — IO_ACS | SA_ACC

Accept a connection to the socket identified by channel number.

Parameters

P1 Not used

P2 Not used

P3 Address of SOioctl structure. The following fields of the SOioctl structure contain significant values on input or output:

Address flag – Input (nonzero if socket address is desired)
Socket address – Output (address of initiating host); optional

The fields of the SOioctl data structure are explained in Table 5-2.

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. This call is used with connection-based (stream) socket types.

5.4.2. CLOSE A SOCKET — IO_ACS | SA_CLS

Close a socket.

Parameters

P1 Not used

P2 Not used

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. The socket will not be closed until all pending I/O on that socket complete.

5.4.3. CONNECT TO REMOTE SOCKET — IO_ACS | SA_CON

Initiate connection request using the underlying protocol of the socket identified by the channel number.

Parameters

P1 Not used

P2 Not used

P3 Address of SOioctl structure. The following fields of the SOioctl structure contain significant values on input or output:

Address flag – Input (nonzero)
Socket address – Input (address of host to connect to)

The fields of the SOioctl data structure are explained in Table 5-2.

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. EXOS status information is returned in the second byte of the IOSB.

5.4.4. OPEN A SOCKET — IO_ACS | SA_OPN

Create a communication end point and return a descriptor, known as socket.

Parameters

P1 Not used

P2 Not used

P3 Address of SOioctl structure. The following fields of the SOioctl structure contain significant values on input or output:

Address flag – Input (nonzero if socket address is provided)
Socket address – Input (address); optional
Protocol flag – Input (nonzero if socket protocol if provided)
Socket protocol – Input (protocol family and protocol within family)
Socket type – Input (stream, datagram, raw, link)
Options – Input (a word of one-bit flags)

The fields of the SOioctl data structure are explained in Table 5-2. Note that if a socket address is not specified, either one is assigned by the software or it is to be filled in later (for example, when later accepting a connection).

P4 Not used

P5 Not used

P6 Not used

Privilege Restrictions

None

Notes

1. The type of socket created defines the semantics for communication. These types are listed on page 5-1.
2. The EXOS processor returns a socket ID to the driver if a socket is created. This is saved in the newly created channel descriptor.
3. The driver returns the channel number in the second word of the IOSB. The channel number is used in future I/O calls to the socket.

5.4.5. OBTAIN SOCKET ADDRESS — IO_ACS | SA_SAD

Get socket address.

Parameters

P1 Not used

P2 Not used

P3 Address of SOioctl structure. The following fields of the SOioctl structure contain significant values on input or output:

Address flag – Output (nonzero)
Socket address – Output (socket address structure for remote host)

The fields of the SOioctl data structure are explained in Table 5-2.

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. Any EXOS error status is returned in the second byte of the IOSB.

5.4.6. CHECK FOR I/O READINESS — IO_ACS | SA_SEL

Places the address of the AST routine (specified in the QIO call) into a list of AST routines for this socket. When the socket is selected, control passes to this AST routine when the socket becomes ready for the requested operation.

Parameters

P1 Not used

P2 Not used

P3 Not used

P4 Flag (read = 0, write = 1)

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. A single AST routine only gives information about the socket corresponding to the channel number given in the QIO call. To get information about other sockets, one QIO call must be issued for each socket.
2. The channel number is returned in the second word of the IOSB.
3. The IOSB address is returned on the top of stack when control is passed to the AST routine, but currently there is no return status.
4. If there are multiple select calls on the same socket and the socket becomes ready, all the ASTs specified in the select calls will be executed. However, the order of execution is not guaranteed.

5.4.7. RECEIVE URGENT SIGNAL — IO_ACS | SA_URG

Specify an AST routine that receives control when an urgent signal arrives.

Parameters

P1 Not used

P2 Not used

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. This request remembers the address of the AST routine specified in the AST parameter.
2. When an IO_KIL call is issued, the AST specified in the IO_ACS | SA_URG call is canceled.

5.4.8. UNSELECT A SOCKET — IO_ACS | SA_USL

This function code cancels the select function. The requested AST in the IO_ACS | SA_SEL call is not executed.

Parameters

P1 Not used

P2 Not used

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

5.4.9. KILL ALL OUTSTANDING I/O — IO_KIL

Terminate all the task's pending I/O requests on the EXOS device and return abnormal I/O termination status to the requester.

Parameters

P1 Not used

P2 Not used

P3 Not used

P4 Not used

P5 Not used

P6 Not used

Privilege Restrictions

Excelan advises that IO_KIL not be used.

Notes

1. When this call is issued, all channels and their corresponding sockets opened by the task, if any, are closed. This forces the EXOS device to reply to all pending I/O requests on it and to terminate all pending I/O. Then all outstanding I/O requests to the EXOS device are aborted one by one, returning an aborted status (IE.ABO) to the requester.
2. When a task that has open sockets with I/O pending on them is terminated by either the MCR ABO command or the executive directive ABRT$, all pending I/O requests are terminated. However, because there is no requester, no termination status is returned.

5.4.10. ADD ROUTING TABLE ENTRY — IO_EXC | EX_ART

Add an entry in the routing table.

Parameters

P1 Address of buffer that contains the routing table entry

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

The channel must be opened in write access mode. Both task and terminal user privileges are required.

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.

2. The EXOS 8031 has two types of routes, routes to hosts and routes to networks. Each type is stored in a separate hashing table. The host routes are more specific, so they take precedence when packets are transmitted.

   All routing requests use the following route data structure:

Field	Length	Comments
Destination	16 bytes	Address of host or network to which route goes; first variant of socket address structure (see Table 5-3).
Gateway	16 bytes	Address of gateway that route uses; first variant of socket address structure (see Table 5-3).
Next pointer	2 bytes	Pointer to next entry in routing table (for internal use). If a value is sent and a return value requested, return the value that was sent. Otherwise, when requesting a route, specify zero.
Dummy	2 bytes	For alignment purposes.
Raw packets	4 bytes	Number of raw packets forwarded.
Interface	2 bytes	For internal use. If a value is sent and a return value requested, return the value that was sent. Otherwise, when requesting a route, specify zero.
Dummy	2 bytes	For alignment purposes.
Routing flags	1 byte	Indicate whether the route is functional and whether it is to a host or a network (defined below).
Reference count	1 byte	Number of held references to that route (for internal use). If a value is sent and a return value requested, return the value that was sent. Otherwise, when requesting a route, specify zero.
Hash	2 bytes	Speed up route lookup time (for internal use). If a value is sent and a return value requested, return the value that was sent. Otherwise, when requesting a route, specify zero.

The routing flags have the following values and symbolic names:

Symbolic Name	Value	Meaning
RTF_UP	0x1	Route usable
RTF_GATEWAY	0x2	Destination is a gateway. If 1, the gateway field is meaningful; if 0, *destination* is directly connected and no routing is necessary
RTF_HOST	0x4	If 1, route applies to specific host; otherwise, the route applies to the entire destination network

5.4.11. CLOSE ADMINISTRATIVE CHANNEL — IO_EXC | EX_CLS

Close an administrative channel on the EXOS driver. This call clears the channel descriptor associated with the channel number and makes it available. All further I/O on this channel is not honored unless the channel is reopened.

Parameters

P1 Not used

P2 Not used

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

5.4.12. READ CONFIGURATION MESSAGE — IO_EXC | EX_CNF

Read configuration message from the EXOS board.

Parameters

P1 Address of buffer to receive configuration message

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. The format of the configuration message is given in the *EXOS 203 Ethernet Front-End Processor Reference Manual* and the *EXOS 204 Ethernet Front-End Processor Reference Manual.*
2. The buffer length should be at least 80 bytes. For the EXOS 8032 the maximum length of the data buffer can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.

5.4.13. DELETE ARP TABLE ENTRY — IO_EXC | EX_DAR

Delete an entry from the ARP table.

Parameters

P1 Address of buffer specifying the entry to be deleted.

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

The channel must be opened in write access mode. Both task and terminal user privileges are required.

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. To delete an entry from the ARP cache, the host should fill in the Internet address field in an EXarp structure (see EX_SAR in Section 5.4.21) and pass it to the EXOS board with the EX_DAR function call. The entry containing this Internet address is deleted.

5.4.14. DELETE ROUTING TABLE ENTRY — IO_EXC | EX_DRT

Delete an entry from the routing table.

Parameters

P1 Address of buffer specifying the routing table entry

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

The channel must be opened in write access mode. Both task and terminal user privileges are required.

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. All routing requests use the route data structure, which is shown in Section 5.4.10.

5.4.15. RETRIEVE ARP TABLE ENTRY — IO_EXC | EX_GAR

Read an ARP table entry.

Parameters

P1 Address of buffer to receive ARP table entry

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. To read all active entries in the ARP cache, the host can send a series of BRDGARP requests in which the host buffer address points to a buffer in shared memory containing the nextaddr structure:

Field	Length	Comments
Next family	2 bytes	Must be 11
Next count	4 bytes	ARP cache entry number; should contain 0 in the first request and should be incremented in each subsequent request
Unused	10 bytes	Must be zero

The reply to the read comes back in an EXarp structure, which is shown under EX_SAR (see Section 5.4.21).

5.4.16. RESET AND CONFIGURE EXOS — IO_EXC | EX_INI

Initialize the EXOS processor. All outstanding I/O to the board is terminated. The EXOS board is reset and then then configuration message is downloaded to it.

Parameters

P1 Not used

P2 Not used

P3 Not used

P4 Mode (link level = 0, host download = 1, net download = 2)

P5 Not used

P6 Not used

Privilege Restrictions

The administrative channel should be opened in write mode, and the task should be a privileged task run from a privileged terminal.

Notes

1. The status code returned by the board is returned in the first byte of the IOSB.
2. Only host download mode is currently implemented.

5.4.17. FETCH NEXT ROUTING TABLE ENTRY — IO_EXC | EX_NRT

Fetch next routing table entry.

Parameters

P1 Address of the buffer to receive the routing table entry

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. The difference between EX_SRT and EX_NRT is that EX_SRT fetches the specified routing table entry, while EX_NRT fetches the next entry in the routing table.
3. All routing requests use the route data structure, which is shown in Section 5.4.10.

5.4.18. OPEN ADMINISTRATIVE CHANNEL — IO_EXC | EX_OPN

Open an administrative channel to perform administrative operations on the EXOS board.

Parameters

P1 Not used

P2 Not used

P3 Not used

P4 Mode (read = 0, write = 1)

P5 Not used

P6 Not used

Privilege Restrictions

To open a channel in write mode, the QIO must be issued from a privileged task running on a privileged terminal.

Notes

1. This call creates a channel descriptor in which it stores the task ID, mode, and tasks privilege information. The EXOS memory locator is set to zero. The channel ID is returned in the second word of the IOSB.

5.4.19. POSITION EXOS MEMORY LOCATOR — IO_EXC | EX_POS

Read or write EXOS memory starting at specified address.

Parameters

P1 Not used

P2 Not used

P3 Not used

P4 Segment address

P5 Offset address

P6 Channel number

Privilege Restrictions

None

Notes

1. The segment and offset address formats are detailed in Chapter 3 of the *EXOS 203 Ethernet Front-End Processor Reference Manual* or the *EXOS 204 Ethernet Front-End Processor Reference Manual.*
2. The input memory address is remembered in the channel descriptor.

5.4.20. READ AND RESET EXOS STATISTICS — IO_EXC | EX_RST

Read and then reset EXOS board statistics.

Parameters

P1 Address of buffer to receive statistics information

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

The channel must be opened in write access mode. Both task and terminal user privileges are required.

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. All board requests use the following data structure:

Field	Length	Comments
Frames transmitted	4 bytes	Number of frames successfully transmitted
Heartbeat	4 bytes	Number of transmitted frames experiencing "heartbeat" failure
DMA underrun	4 bytes	Number of transmissions that failed because of DMA underrun
Frames received	4 bytes	Number of error-free frames received
Alignment error	4 bytes	Number of frames received with alignment errors
CRC error	4 bytes	Number of frames received with CRC errors
Lost frames	4 bytes	Number of frames lost due to lack of buffer space
NX firmware release	2 bytes	Version number of NX firmware release
8031 software release	2 bytes	Version number of EXOS 8031 software release
EXOS board release	2 bytes	Version number of EXOS front-end processor hardware release

For further explanation of the first eight fields in this structure, refer to the *EXOS 204 Ethernet Front-End Processor Reference Manual*. The last three fields contain the release level numbers in ASCII format. For example, for version 3.2, *sw_release* is defined as follows:

```
char  sw_release[2] = {'3', '2'};
```

5.4.21. SET ARP TABLE ENTRY — IO_EXC | EX_SAR

Set a new ARP table entry.

Parameters

P1 Address of buffer containing ARP table entry.

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.

2. To add a new entry to the ARP cache, the host should complete the following EXarp structure and pass it to the EXOS board with the EX_SAR function call:

Field	Length	Comments
Internet address	16 bytes	First variant of socket address data structure (see Table 5-3). If the first entry of this variant specifies the Internet address family, the next two bytes are interpreted as the port number (though in reality, this number is ignored) and the next four as the Internet address. If the first entry of this variant specifies a count, the next four bytes are interpreted as a number *n* and the *n*th entry in the ARP table is retrieved. If *n* exceeds the length of the table, the ENXIO error code is returned.
Ethernet address	16 bytes	Third variant of socket address data structure (see Table 5-3).
Flags	4 bytes	There are two bit positions: ATF_COM (0x2) – completed entry. If set, the entry is complete and establishes a valid mapping. Otherwise, the EXOS board has sent an ARP request but has not received a reply that specifies the Ethernet address. Therefore the Ethernet address is not valid. ATF_PUBL (0x8) – published entry. If set, the entry is "published," meaning that the EXOS board responds to ARP requests from other systems that mention the Internet address.

An error return indicates when the ARP list has been exhausted.

5.4.22. FETCH ROUTING TABLE ENTRY — IO_EXC | EX_SRT

Fetch an entry from the routing table.

Parameters

P1 Address of buffer to receive the routing table entry

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. All routing requests use the route data structure, which is shown in Section 5.4.10.

5.4.23. START EXECUTION OF EXOS PROCEDURE — IO_EXC | EX_STR

Start execution of a downloaded procedure.

Parameters

P1 Not used

P2 Not used

P3 Not used

P4 Base address

P5 Offset address

P6 Channel number

Privilege Restrictions

The channel must be opened in write access mode. Task and terminal user privileges are both required.

5.4.24. READ EXOS STATISTICS — IO_EXC | EX_STS

Read EXOS board statistics.

Parameters

P1 Address of buffer to receive statistics information

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. This function call does not reset board statistics.
3. All board requests use the board data structure, which is shown in Section 5.4.20.

5.4.25. READ FROM EXOS MEMORY — IO_RLB

Read EXOS memory beginning at the location previously set by position memory locator request (IO_EXC | EX_POS).

Parameters

P1 Address of buffer into which read information is placed

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. After reading is complete, the memory locator is updated by the number of bytes read.
3. The number of bytes read is passed to the second word of the IOSB.

5.4.26. PERFORM SOCKET CONTROL OPERATION — IO_SOC

Perform some I/O control operation on a socket.

Parameters

P1 Not used

P2 Not used

P3 Address of the buffer containing any control related information (see Note 3 below) to be passed to the EXOS board or copied to user.

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. The buffer associated with the operation should not be longer than 44 bytes.
2. The different types of socket I/O control operations are passed by the application program as a modifier of the IO_SOC function call. Depending on the operation, either information is copied from the program (from the buffer specified in parameter P3) and passed to the EXOS board together with the specific request or it is returned by the EXOS board and copied into the buffer specified in parameter P3.
3. The following are the socket control operations. They are specified as a function modifier of the IO_SOC function call.

Function Modifier	Symbolic Name	Explanation
SO_SKP	SIOCSKEEP	Enable/disable keep-alives (input parameter; 2 bytes). Keep-alives are explained in Table 5-6. A 1 enables keep-alives on the socket; a 0 disables them.
SO_GKP	SIOCGKEEP	Show keep-alive status (output parameter; 2 bytes). Returns 1 if keep-alives are enabled; returns 0 otherwise.

Function Modifier	Symbolic Name	Explanation
SO_DON	SIOCDONE	Halt socket I/O (input parameter; 2 bytes). Shuts down input and/or output on the designated socket, depending on the value of the parameter: 0 shuts down input, 1 shuts down output, and 2 shuts down input and output. On input, SIOCDONE throws away any buffered receive data and causes subsequent read requests to return an EOF. On output, SIOCDONE causes subsequent write requests to return an EPIPE error code.
SO_SLG	SIOCSLINGER	Enable/disable lingering (input parameter; 2 bytes). Enables/disables lingering on stream sockets and establishes how many seconds a connection lingers during closing. If the parameter is 0xFFFF, a close request blocks indefinitely until all sent data are acknowledged. If 0, lingering is disabled and a close request breaks the connection immediately. A value between 0 and 32767 establishes a limit, in seconds, on the close request. If this time expires before the connection closes normally, the connection is reset. Do not use values outside this range.
SO_GLG	SIOCGLINGER	Show lingering status (output parameter; 2 bytes). Returns the effective linger value, which is interpreted as explained under SIOCSLINGER.
SO_SPG	SIOCSPGRP	Set process group (input parameter; 2 bytes). Associates an arbitrary short value with a socket. After this request the EXOS board can send messages to the host that contain the same value, thus identifying the socket that received out-of-band data.
SO_GPG	SIOCGPGRP	Get process group (output parameter; 2 bytes). Returns the short value set by SIOCSPCRP.

Function Modifier	Symbolic Name	Explanation
SO_ROB	SIOCRCVOOB	Receive out-of-band data (output parameter; one character of urgent data).
SO_SOB	SIOCSNDOOB	Send out-of-band data (input parameter; one character of urgent data).
SO_AMK	SIOCATMARK	Distinguish urgent data (output parameter; 2 bytes). The IO_ACS|SA_URG function call prepares for receipt of urgent data. The application program is then alerted whenever urgent data have been detected. Once alerted, the host can distinguish urgent data from normal data by preceding each receive request with an SIOCATMARK request. When the SIOCATMARK request returns a value of 1, the next receive request will return normal data. At this point, the host may make an SIORCVOOB request, which returns the single byte that was sent out of band. This byte was extracted from the normal TCP data stream.
SO_NBO	FIONBIO	Nonblocking I/O (input parameter; 2 bytes). Enables/disables nonblocking I/O. If 1, nonblocking I/O is enabled on the socket. If 0, it is disabled. When nonblocking I/O is in effect, front-end operations that cannot complete immediately return an EWOULDBLOCK error. This operation governs the host/front-end interaction and is independent of the QIO/QIOW distinction of application program/operating system interaction.
SO_NRD	FIONREAD	Bytes in receive buffer (output parameter; 4 bytes). Returns byte count of data in socket's receive buffer. The host can use this to determine whether it can read data from the socket without blocking.

5.4.27. WRITE TO EXOS MEMORY — IO_WLB

Download software to the EXOS board from the host.

Parameters

P1 Address of buffer containing software to be downloaded

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

The channel must be opened in write access mode. Both task and terminal user privileges are required.

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. If overflow/underflow occurs, it is not reported and no action is taken. Thus it is the program's responsibility to keep track of the addresses and sizes.
3. The contents of the buffer are copied into EXOS memory starting at the address previously set by an IO_EXC|EX_POS command. The memory locator is incremented by the amount of bytes transferred.

5.4.28. RECEIVE MESSAGE FROM SOCKET — IO_XFR | IX_RCV

Receive message that is associated with a channel from a datagram or raw socket.

Parameters

P1 Address of buffer to receive message

P2 Buffer length, in bytes

P3 Address of SOioctl structure

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. The socket address field of the SOioctl structure contains the message source address.
3. The message is truncated if it is longer than the receive buffer specified by parameter P2.
4. A one-byte status code from the EXOS processor is returned in the second byte of IOSB.
5. The length of message received is returned in the second word of the IOSB.

5.4.29. READ DATA FROM TCP STREAM — IO_XFR | IX_RDS

Read information from a TCP stream.

Parameters

P1 Address of buffer to receive data

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. A count of the data read is returned in the second word of the IOSB.
3. Any status code returned by the EXOS processor is returned in the second byte of the IOSB.

5.4.30. SEND DATAGRAM TO REMOTE SOCKET — IO_XFR | IX_SND

Send datagram to a remote socket using UDP.

Parameters

P1 Address of buffer containing data to be sent

P2 Buffer length, in bytes

P3 Address of SOioctl structure

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. The driver returns the status code in the second byte of the IOSB.
3. The number of bytes sent is returned in the second word of the IOSB.

5.4.31. WRITE DATA TO TCP STREAM — IO_XFR | IX_WRS

Write data to a TCP stream.

Parameters

P1 Address of buffer containing data to be written

P2 Buffer length, in bytes

P3 Not used

P4 Not used

P5 Not used

P6 Channel number

Privilege Restrictions

None

Notes

1. For the EXOS 8032 the maximum length of the data buffer (parameter P2) can be 1024 bytes. A buffer size larger than this causes the error condition IE_SPC.
2. The EXOS processor returns a one-byte reply code in the second byte of the IOSB.
3. The length of the data sent is returned in the second word of the IOSB.

5.5. SAMPLE PROGRAM AND PROGRAM FUNCTIONS

This section presents one program and several examples of C functions written using the QIO function calls discussed in this chapter.

The program function shows how to display entries in the routing table.

The C functions perform the following operations:

- Create a socket
- Get socket address information
- Connect to a remote socket
- Accept a connection
- Read data from a stream
- Write data to a stream
- Enumerate routing table entries

The header file *ttcp.h*, as well a copy of all these functions, are included in the standard 8031 distribution.

The I/O status block used in the functions has the following structure:

```
status {
        TINY cc;                    /* condition code */
        TEXT lc;                    /* reply code from EXOS board */
        COUNT nread;                /* I/O count */
}
```

The following include and define statements should be used with the first six functions:

```
#include <std.h>
#include "ttcp.h"

#define   SOLUN 20                  /* logical unit used in EXOS I/O */
#define   SOEFN 1                   /* event flag used in EXOS I/O */
```

The function *cpybuf* used with some of these functions has the form

```
cpybuf(dest, src, len)
```

It copies *len* bytes from *src* to *dest*.

5.5.1. Create a Socket

The following function creates a socket for communication. It takes four input parameters:

- type – socket type (see Table 5-5)
- proto – protocol
- addr – socket address
- options – options to be used

The value returned by the function is the channel number of the opened socket.

Function

```
int socket(type, proto, addr, options)
int         type;       /* socket type */
struct      sockproto  *proto;                  /* protocol */
struct      sockaddr   *addr;                   /* socket address */
int         options;                            /* options for socket */
{
            int         ret;                    /* return value */
            struct      SOictl  soctl;          /* socket control structure */
            struct      iosb  status;           /* I/O status block */

            if (addr)   /* address is specified; fill in socket control structure */
            {
                        soctl.hassa = 1;
                        cpybuf (&soctl.sa, addr, sizeof (struct sockaddr) );
            }
            else
                        soctl.hassa = 0;        /* no socket address */

            if (proto)  /* protocol specified; fill in socket control structure */
            {
                        soctl.hassp = 1;
                        cpybuf (&soctl.sp, proto, sizeof (struct sockproto) );
            }
            else
                        soctl.hassp = 0;

            soctl.type = type;
            soctl.options = options;
```

```
    /* open socket; channel number is returned in the IOSB */

    ret = emt (QIOW, IO_ACS | SA_OPN, SOLUN, SOEFN,
                &status, 0, 0, 0, &soctl, 0, 0, 0);
    if ( (ret >= 0) && (status.cc >= 0) && (status.lc == 0) )
                return(status.nread);
    else if (ret < 0)
                return(ret - 512);                  /* directive error */
    else if (status.cc < 0)
                return(status.cc - 512);            /* generic I/O error */
    else
                return(-(status.lc & 0xff) );       /* device-specific error */
}
```

5.5.2. Get Socket Address Information

The following function gets the socket address from the remote host. It takes two input parameters:

- chan – socket channel number
- addr – socket address structure

There is no return value. The socket address is returned in the structure *addr*.

Function

```
sktaddr(chan, addr)
int         chan;
struct      sockaddr  *addr;
{
            struct      SOictl  soioctl;            /* socket I/O control structure */
            struct      iosb  status;               /* I/O status block */
            int         ret;                        /* QIO status */

            if (addr) {  /* address specified; fill in socket control structure */
                        soioctl.hassa = 1;
                        cpybuf(&soioctl.sa, addr, sizeof (struct sockaddr) );
            }
            else
                        soioctl.hassa = 0;

            ret = emt(QIOW, IO_ACS|SA_SAD, SOLUN, SOEFN,
                        &status, 0, 0, 0, &soioctl, 0, 0, chan);
            if ( (ret >= 0) && (status.cc >= 0) && (status.lc == 0) && addr)
                        cpybuf (addr, &soioctl.sa, sizeof (struct sockaddr) );
}
```

5.5.3. Connect to Remote Socket

The following function establishes a connection to a remote socket. It takes two input parameters:

- chan – socket channel number
- addr – socket address structure

The function's return value contains status information regarding the connection.

Function

```
int connect(chan,addr)
int         chan;
struct      sockaddr *addr;
{
            int ret;                                    /* QIO return status */
            struct SOictl soioctl;                      /* socket control structure */
            struct iosb status;                         /* I/O status */

            if (addr) {  /* address specified; fill in socket control structure */
                        soioctl.hassa = 1;
                        cpybuf (&soioctl.sa, addr, sizeof (struct sockaddr) );
            }
            else
                        soioctl.hassa = 0;

            ret = emt(QIOW, IO_ACS | SA_CON, SOLUN, SOEFN,
                        &status, 0, 0, 0, &soioctl, 0, 0, chan);
            if ( (ret >= 0) && (status.cc >= 0) && (status.lc == 0) )
                        return(status.nread);
            else if (ret < 0)
                        return(ret – 512);              /* directive error */
            else if (status.cc < 0)
                        return(status.cc – 512);        /* generic I/O error */
            else
                        return(–(status.lc & 0xff) );   /* device-specific error */
}
```

5.5.4. Accept Connection from Remote Socket

The following function accepts a connection from a remote socket. It takes one input parameter – chan, the socket channel number. The function returns one parameter – from, the socket address of the remote socket

The function's return value contains status information regarding the accept.

Function

```
int accept(chan, from)
int         chan;
struct      sockaddr *from;
{
            int ret;
            struct SOictl soioctl;
            struct iosb status;

            soioctl.hassa = from ? 1 : 0;
            ret = emt(QIOW, IO_ACS | SA_ACC, SOLUN, SOEFN,
                        &status, 0, 0, 0, &soioctl, 0, 0, chan);
            cpybuf(from, &soioctl.sa, sizeof (struct sockaddr) );

            if ( (ret >= 0) && (status.cc >= 0) && (status.lc == 0) )
                        return(status.nread);
            else if (ret < 0)
                        return(ret – 512);              /* directive error */
            else if (status.cc < 0)
                        return(status.cc – 512);        /* generic I/O error */
            else
                        return( – (status.lc & 0xff) );  /* device-specific error */
}
```

5.5.5. Read Data from Stream

The following function reads data from a stream. It takes three input parameters:

- chan – socket channel number
- buf – address of the buffer to receive the data
- len – the buffer size, in bytes

The function's return value indicates the numbers of bytes read.

Function

```
xread(chan, buf, len)
int         chan;
char        *buf;
int         len;
{
            int         ret;
            struct      iosb status;

            ret = emt(QIOW, IO_XFR | IX_RDS, SOLUN, SOEFN,
                        &status, 0, buf, len, 0, 0, 0, chan);

            if ( (ret >= 0) && (status.cc >= 0) && (status.lc == 0) )
                        return(status.nread);
            else if (ret < 0)
                        return(ret - 512);              /* directive error */
            else if (status.cc < 0)
                        return(status.cc - 512);        /* generic I/O error */
            else
                        return( - (status.lc & 0xff) );  /* device-specific error */
}
```

5.5.6. Write to Data Stream

The following function writes to a data stream. It takes three input parameters:

- chan – socket channel number
- buf – address of the buffer to receive the data
- len – the buffer size, in bytes

The function's return value indicates the number of bytes written to the stream.

Function

```
xwrite(chan, buf, len)
int         chan;
char        *buf;
int         len;
{
            int         ret;
            struct      iosb status;

            ret = emt(QIOW, IO_XFR | IX_WRS, SOLUN, SOEFN,
                        &status, 0, buf, len, 0, 0, 0, chan);

            if ( (ret >= 0) && (status.cc >= 0) && (status.lc == 0) )
                        return(status.nread);
            else if (ret < 0)
                        return(ret – 512);              /* directive error */
            else if (status.cc < 0)
                        return(status.cc – 512);        /* generic I/O error */
            else
                        return( – (status.lc & 0xff) );  /* device-specific error */
}
```

5.5.7. Enumerate Routing Table Entries

The program given in this section shows how to enumerate entries in the routing table using the IO_EXC|EX_SRT and IO_EXC|EX_NRT function calls. The comments in the code explain what is being done.

Program

```
1   #include <std.h>
2   #include "ttcp.h"                          /* provided in the 8031 distribution */

3   #define u_long long
4   #define u_short unsigned short
5   #define RTHASHSIZ       7

6   #define RTF_UP          0x1                /* route usable; see Section 5.4.10 */
7   #define RTF_GATEWAY     0x2                /* destination is a gatewa; see Section 5.4.10 '
8   #define RTF_HOST        0x4                /* host entry; net otherwise; see Section 5.4.1(

9   #define RTFREE(rt) \
10      if ( (rt)->rt_refcnt == 1) \
11          rtfree(rt); \
12      else \
13          (rt)->rt_refcnt--;

14  #define SOLUN           20
15  #define SOEFN           1

16  struct rtentry {                           /* route entry structure; see Section 5.4.10 */
17      struct sockaddr rt_dst;                /* key */
18      struct sockaddr rt_gateway;            /* value */
19      struct rtentry *rt_next;               /* next pointer */
20      int dummy;                             /* host pointer=4; board pointer =2 */
21      u_long rt_use;                         /* number of raw packets forwarded */
22      short rt_ifp;                          /* interface to use */
23      short dummyx;                          /* host pointer=4; board pointer =2 */
24      char rt_flags;                         /* up/down?, host/net */
25      char rt_refcnt;                        /* # held references */
26      u_short rt_hash;                       /* to speed up lookups */
27  };

28  struct rtentry route = { 0 };

29  static long htonl();

30  main()
31  {
32      struct sckadr_in *sin;
33      short n;
34      char *cmd;
35      int channel;                           /* channel number */
```

```
36      channel = brdopen(0);                          /* open administrative channel for read */
37      if (channel < 0) {
38          error("Error in opening administrative channel");
39          exit(1);
40      }

41      /* display all routes (host and network) */

42      route.rt_next = (struct rtentry * )0;

43      for (n = 0; n < RTHASHSIZ; n++) {
44          route.rt_hash = n;
45          route.rt_flags = RTF_HOST|RTF_GATEWAY;
46          if (showroute(channel, &route) )
47              continue;
48          show(&route);
49          while (route.rt_next) {                    /* more entries in the table ... */
50              route.rt_hash = n;
51              route.rt_flags = RTF_HOST|RTF_GATEWAY;
52              if (showroute(channel, &route) )
53                  continue;
54              show(&route);
55          }
56      }
57      route.rt_next = (struct rtentry * )0;

58      /* display network routes */

59      for (n = 0; n < RTHASHSIZ; n++) {
60          route.rt_hash = n;
61          route.rt_flags = RTF_GATEWAY;
62          if (showroute(channel, &route) )
63              continue;
64          show(&route);
65          while (route.rt_next) { /* more entries in the table ... */
66              route.rt_hash = n;
67              route.rt_flags = RTF_GATEWAY;
68              if (showroute(channel, &route) )
69                  continue;
70              show(&route);
71          }
72      }

73      /* if default route exists display it */

74      sin = (struct sckadr_in *)&route.rt_dst;
75      sin->sin_family = AF_INET;
76      sin->sin_addr.s_addr = 0l;
77      route.rt_flags = RTF_GATEWAY;
78      if (getroute(&route) == 0)
79          show(&route);
80  }
```

```
81  show(route)                                    /* get host route information */
82  struct rtentry *route;
83  {
84      u_long destination, gateway;
85      struct sckadr_in *sin;

86      sin = (struct sckadr_in *)&route->rt_dst;
87      destination = ( (struct sckadr_in *)&route->rt_dst)->sin_addr.s_addr;
88      destination = htonl( destination );
89      gateway = ( (struct sckadr_in *)&route->rt_gateway)->sin_addr.s_addr;
90      gateway = htonl( gateway );
91      printf("%s %d.%d.%d.%d, gateway %d.%d.%d.%d\n",
92          (IN_LNAOF(sin->sin_addr) )? "host": "network",
93          (int)( (destination>>24)&0xff),
94          (int)( (destination>>16)&0xff),
95          (int)( (destination>>8)&0xff),
96          (int)( (destination>>0)&0xff),
97          (int)( (gateway>>24)&0xff),
98          (int)( (gateway>>16)&0xff),
99          (int)( (gateway>>8)&0xff),
100         (int)( (gateway>>0)&0xff) );
101 }

102 static unsigned short htons(hostshort)         /* convert short entity from */
103 unsigned short hostshort;                      /* host order to network order */
104 {
105     return ( (unsigned short) ( (hostshort << 8) | ( (hostshort>>8) & 0xff) ) );
106 }

107 static long htonl(hostlong)                    /* convert long entity from */
108 long hostlong;                                 /* host order to network order */
109 {
110     union {
111         long l;
112         struct {
113             unsigned short s_high, s_low;
114         } sl;
115     } h;

116     h.l = hostlong;
117     h.sl.s_high = htons(h.sl.s_high);
118     h.sl.s_low = htons(h.sl.s_low);
119     return (h.l);
120 }

121 int brdopen(mode)                              /* open an administrative channel */
122 int mode;                                      /* mode for the board: 0 = read, 1 = write */
123 {
124     int ret;
125     struct iosb status;

126     ret = emt (QIOW, IO_EXC | EX_OPN, SOLUN, SOEFN, &status, 0, 0, 0, 0, mode, 0, 0);
127     return(setstatus(ret, &status) );
128 }
```

```
129 int brdclose(channel)                        /* close an administrative channel */
130 int channel;
131 {
132     int ret;
133     struct iosb status;

134     ret = emt(QIOW,IO_EXC|EX_CLS,SOLUN,SOEFN,&status,0,0,0,0,0,0,channel);
135     return(setstatus(ret, &status) );
136 }

137 int getroute(channel, route)                    /* fetch next routing table entry */
138 int channel;
139 struct rtentry *route;
140 {
141     int ret;
142     struct iosb status;

143     ret = emt(QIOW, IO_EXC|EX_NRT, SOLUN, SOEFN, &status, 0, route,
144         sizeof (struct rtentry), 0, 0, 0, channel);
145     return(setstatus(ret, &status) );
146 }

147 int showroute(channel, route)                   /* get network route information */
148 int channel;
149 struct rtentry *route;
150 {
151     int ret;
152     struct iosb status;

153     ret = emt(QIOW, IO_EXC|EX_SRT, SOLUN, SOEFN, &status, 0, route,
154         sizeof (struct rtentry), 0, 0, 0, channel);
155     return(setstatus(ret, &status) );
156 }

157 setstatus(value, status)                        /* determine status and error conditions */
158 int value;
159 struct iosb *status;
160 {
161     if ( (value >= 0) && (status->cc >= 0) && (status->lc == 0) )
162         return(status->nread);
163     else if (value < 0)
164         return(value-512);                      /* directive error */
165     else if (status->cc < 0)
166         return(status->cc - 512);               /* generic I/O error */
167     else
168         return(-(status->lc & 0xff) );          /* device specific error */
169 }
```

Appendix A
UTILITIES

A.1. INTRODUCTION

This appendix provides comprehensive descriptions for the network systems and application utilities that are included as part of the the EXOS 8031 TCP/IP software package. These utilities are described in a user's guide format in Chapters 3 and 4.

All TCP/IP utilities are installed by the network startup command file (LB:[1,1]EXOSLOAD.CMD).

A.2. ADDRESS RESOLUTION CONTROL UTILITY — ARP

The *arp* utility displays and manipulates the entries in the Internet-to-Ethernet translation table generated by the Address Resolution Protocol (ARP). This table contains the Internet and Ethernet address mappings entered by a user (usually for non-ARP-supporting hosts) and for those hosts on the network that responded to individual ARP broadcasts. The table also contains unresolved host/address entries.

Format

arp *host_name*
arp –a [entire]
arp –d *host_name*
arp –f *filename*
arp –s *host_name ethernet_address* [temp] [publ]

Options

arp *host_name*
Displays the ARP entry for the specified host.

–a [entire]
Displays all the current ARP entries in the translation table. Normally, only the host entries that have been resolved are displayed. However, by specifying *entire*, the entire contents of the ARP table, which includes incomplete and empty entries, are displayed.

–d *host_name*
Deletes the entry for *host_name* from the translation table.

–f *filename*
Reads the file *filename* and copies entries from the file into the ARP translation table. The file can contain multiple entries in the format

host_name ethernet_address [temp] [publ]

The *temp* and *publ* options are explained below.

–s *host_name ethernet_address* [temp] [publ]
Creates an ARP entry for the host *host_name* with the Ethernet address *ethernet_address*. The Ethernet address is given as six hexadecimal bytes separated by colons. Alternatively, each of the six bytes can be specified in octal (numbers start with the digit 0), decimal (numbers start with the digits 1 to 9), or hexadecimal (numbers start with 0X or 0x). In this alternate addressing scheme, entries must be separated by periods. In both cases, all hexadecimal digits should be lowercase. Entries made into the table are permanent unless the *temp* option is specified.* Permanent entries do not age out of the cache and are given preference to remain in the cache when the table overflows. The publ option "publishes" the entry. That is, a host will respond to ARP requests for *host_name* even though the host name or Internet address is not its own.

*In the current release, all entries into the ARP table are transient.

Privilege Restrictions

You must be a privileged user in order to modify translation table and alter the contents of the ARP cache. If a nonprivileged user uses the command *arp –f*, an error message is displayed for each line in the file.

Example

The following example shows how to use the *arp* utility. It also illustrates how Ethernet addresses must be specified.

```
arp –s munich 0x8:0x0:0x14:0x60:0x0:0x1
```

A.3. BOARD STATISTICS UTILITY — BSTAT

This utility obtains packet-traffic statistics from the EXOS front-end controller board and prints all statistics that have nonzero values. NX, the on-board firmware, compiles these statistics and stores them in the board's memory. These statistics show

- Total number of packets received from and transmitted to the Ethernet
- Number of packets received with an alignment error (packets must be in 8-bit multiples)
- Number of packets received with CRC errors
- Number of packets lost because no receive buffers were available
- Number of packets suffering from SQE (transceiver "heartbeat") test failures
- Number of transmissions that failed with DMA underrun

bstat displays the statistics since the last reset or download of the protocol module, whichever occurred most recently.

Format

bstat [–r] [–v]

Options

–r Resets all board statistics to zero (0).

–v Lists the current version numbers for the TCP/IP software, the on-board NX firmware, and the EXOS board hardware.

Privilege Restrictions

You must be a privileged user to use the –r option.

A.4. FILE TRANSFER PROTOCOL UTILITY — FTP

The *ftp* utility transfers files between a local and a remote host using the standard ARPANET File Transfer Protocol. A remote host can be specified in the command line. If specified, *ftp* immediately attempts to connect to an FTP server on that host. Otherwise, *ftp* enters its command interpreter and awaits instructions from the user. When awaiting commands, *ftp* displays the prompt "ftp>." *ftp* recognizes the commands listed in the "Commands" section below.

Format

ftp [–d] [–g] [–i] [–n] [–v] [*remote_host*]

Options

–d Enables debugging.

–g Disables filename globbing. (Globbing is explained below under the *glob* command.)

–i Disables interactive prompting during multiple file transfers.

–n Disables autologin.

–v Enables displaying of all responses from the remote server and reports data transfer statistics. This is the default if *ftp* is used interactively.

Commands

append *local_file* [*remote_file*]
Appends a local file to a remote file. If *remote_file* is not specified, *local_file* copies under its original name. The current settings for *type*, *format*, *mode*, and *structure* are used during file transfers.

ascii Sets the file transfer *type* to network ASCII (the default).

bell Sounds a bell after each file transfer command completes (toggle).

binary Sets the file transfer *type* to support binary image transfer.

bye Terminates the *ftp* session with the remote server, exits *ftp* command mode, and returns to the operating system.

cd *remote_directory*
Changes the current working directory on the remote host to *remote_directory*.

close Terminates the *ftp* session with the remote server but remains in the *ftp* command interpreter.

delete *remote_file*
Removes *remote_file* on the remote host.

debug [*debug_value*]
Toggles debugging mode. When *debug_value* is nonzero, debugging is on and *ftp* displays each command sent to the remote host.

dir [*remote_directory* [*local_file*]]
: Lists the contents of *remote_directory* on the remote host in *local_file* on the local host. If *remote_directory* is not specified, the current working directory on the remote host is listed. If *local_file* is not specified, the *remote_directory* listing is shown on the terminal.

form *format*
: Sets the file transfer form to *format*. The default format is "nonprint." This is the only format currently supported.

get *remote_file* [*local_file*]
: Copies *remote_file* from the remote host to the local system. If *local_file* is not specified, *remote_file* retains its original name when copied to the local machine. If a - (hyphen) is specified for *local_file*, the file is displayed on the terminal. If *remote_file* is specified as [*directory_name*].*file_name*, the file is copied to the local host under the same directory and file names. The current settings for *type*, *format*, *mode*, and *structure* are used during file transfers.

glob
: Toggles filename wildcard processing. When on, each local file or pathname is processed for wildcard characters. These characters are * and %. Local filenames are expanded according to the wildcard processing of MCR. Remote files specified in multiple-item commands (such as *mget* and *mput*) are globbed by the remote server. When globbing is off, all files and pathnames are treated literally.

  Note that since RSX-11M does not allow the use of % or * in file specifications, you should always keep glob on.

hash
: Toggles printing a hash sign (#) for each data block (1024 bytes) transferred.

help [*command*]
: Lists all *ftp* commands. If *command* is specified, on-line help documentation for that command is provided.

lcd [*local_directory*]
: Changes the current working directory on the local system to *local_directory*. If *local_directory* is not specified, the user's home directory becomes the current working directory.

ldir [*local_directory*]
: Gives a long listing of *local_directory* on the local host. If *local_directory* is not specified, the contents of the current working directory are listed.

lls [*local_directory*]
: Lists the abbreviated contents of *local_directory* on the local host. If *local_directory* is not specified, the contents of the current working directory are listed.

lpwd
: Prints the current working directory of the local host.

ls [*remote_directory* [*local_file*]]
: Lists the abbreviated contents of a directory on the remote host in *local_file*. If *remote_directory* is not specified, the contents of the current working directory are listed. If *local_file* is not specified, the *remote_directory* listing is shown on the terminal.

mdelete *remote_file1* [*remote_file2* ...]
: Deletes one or more files on the remote host. If globbing is enabled, *remote_files* is first expanded via *ls*.

mdir [*remote_file1* [[*remote_file2* ...] *local_file*]]
: Places directory information for a listing of one or more files from the remote host in *local_file*. The last file in the file list is treated as *local_file*. If *remote_file* is not specified, information for the current remote directory is displayed on the screen.

mget *remote_file1* [*remote_file2* ...]
: Places one or more files from the remote host in the current working directory on the local host. If globbing is enabled, remote filenames are expanded via *ls*.

mkdir *remote_directory*
: Creates *remote_directory* on the remote host. The directory must not already exist. Note: The 8031 and 8032 FTP servers do not support this function.

mls [*remote_file1* [[*remote_file2* ...] local_file]]
: Places abbreviated directory information for one or more files on the remote host into *local_file*. If *remote_file* is not specified, information for the current remote directory is displayed on the screen. The last file in the file list is treated as *local_file*. If a - (hyphen) is specified for *local_file*, the file is displayed on the terminal.

mode [*mode_name*]
: Sets the file transfer mode to *mode_name*. The default mode is "stream." This is the only mode currently supported.

mput *local_file1* [*local_file2* ...]
: Copies one or more files from the local host to the current working directory on the remote host.

open *host_name* [*port*]
: Establishes a connection to the specified *host_name* FTP server. If *port* is specified, *ftp* attempts to contact an FTP server at that port. The default *port* for *ftp* is 21. If the *auto_login* option is on (default), *ftp* also attempts to automatically log the user into the FTP server (see FTPD).

prompt
: Toggles interactive prompting. This allows the user to selectively retrieve or store files during multiple file transfers. When off, any *mget* or *mput* transfers all files. Default is on.

put *local_file* [*remote_file*]
: Copies a file from the local system to the remote system. If *remote_file* is not specified, *local_file* is copied under its original name. If a - (hyphen) is specified for *local_file*, *ftp* reads input from the terminal. Type CTRL-Z to terminate input. The current settings for *type*, *format*, *mode*, and *structure* are used during file transfers.

pwd
: Displays the name of the current working directory of the remote host.

quit
: Terminates the *ftp* session with the remote server, exits *ftp* command mode, and returns to the operating system.

quote
: Sends an arbitrary *ftp* command to the server, bypassing normal command parsing. The user is prompted for a command line. (For use by experts only.)

recv *remote_file* [*local_file*]
: Same as the *get* command.

remotehelp [*command_name*]
: Displays help documentation from the remote FTP server. If *command_name* is specified, help for that command is provided.

rename *remote_file1 remote_file2*
: Renames *remote_file1* to *remote_file2*.

rmdir *remote_directory*
: Deletes the directory *remote_directory*. The remote directory must be empty before it can be deleted.

send *local_file* [*remote_file*]
: Copies a file from the local system to the remote system. If *remote_file* is not specified, *local_file* retains its original name when copied to the remote system. The current settings for *type, format, mode*, and *structure* are used during file transfers.

sendport
: Toggles the use of port commands. When *sendport* is on (default), *ftp* attempts to use the port specified by the client when establishing a connection for each data transfer. If this fails, *ftp* uses the default data port (port 23). When *sendport* is off, *ftp* uses the default port. This is useful for certain FTP implementations that ignore port commands but incorrectly indicate that they have been accepted.

status Displays the current state of all *ftp* options.

struct [*struct_name*]
: Sets the file transfer structure to *struct_name*. The default structure is "stream." This is the only structure currently supported.

type [*type_name*]
: Sets the file transfer type to *type_name*. If *type_name* is not specified, the current type is printed. Network ASCII is the default. The other type is image (binary).

user *user_name* [*password*] [*account*]
: Identifies the user to the remote FTP server. If *password* is not specified and one is required by the server, *ftp* prompts the user for one (after disabling local echo). If *account* is not specified and one is required by the server, *ftp* prompts the user. Unless *ftp* is invoked with *auto_login* disabled, this process occurs when first connecting to the FTP server.

verbose
: Toggles verbose mode. When on (default when *ftp* is used interactively), all responses from the FTP server are displayed, and statistics on the efficiency of the file transfer are reported.

? [*command*]
: Lists all *ftp* commands. If *command* is specified, on-line documentation for that command is provided.

Privilege Restrictions

No special privileges are required to use this utility.

Notes

1. The *ftp* commands that require parameters are interactive. That is, if a command is entered without the required parameters, the system prompts for the parameters. The following commands are interactive: *append*, *delete*, *get*, *mdelete*, *mget*, *mkdir*, *mput*, *open*, *put*, *quote*, *recv*, *rename*, *rmdir*, and *send*.
2. The remote host does not have to be running RSX-11M, but it must support the File Transfer Protocol (FTP). At the time of the transfer it must also be running the FTP server program *ftpd* or the equivalent.
3. File specification for the remote host is done according to the conventions of the remote host.
4. The user must be a recognized user on the remote host and must know any required passwords.
5. *ftp* supports only the defaults for the file transfer parameters *mode*, *form*, and *struct*.
6. No line editing is performed on passwords.

A.5. DARPA INTERNET FILE TRANSFER PROTOCOL SERVER FTPD

ftpd is the DARPA Internet file transfer prototocol server process. It uses the FTP protocol and listens at the port specified in the *ftp* service specification.

Format

ftpd [–d] [–l] [–t*timeout*]

Options

–d Displays socket debug messages.

–l Displays server debug messages.

–t*timeout*
Defines the inactivity timeout period to be *timeout* seconds. If it is not specified, the FTP server times out of an inactive session after 60 seconds.

Privilege Restrictions

Only a privileged user can start this utility.

Notes

1. The FTP server supports the following *ftp* requests (Internet RFC 765). They can be entered in uppercase or lowercase letters.

Request	Description
ACCT	Specify account (ignored)
ALLO	Allocate storage (vacuously)
APPE	Append to a file
CWD	Change working directory
DELE	Delete a file
HELP	Get help
LIST	List directory files
MODE	Specify data transfer mode
NLST	List file names
NOOP	Do nothing
PASS	Specify password
PORT	Specify data connection port
QUIT	End session
RETR	Retrieve a file
RNFR	Specify rename-from file name
RNTO	Specify rename-to file name
STOR	Store a file
STRU	Specify data transfer structure
TYPE	Specify data transfer type
USER	Specify user name
XCUP	Change to parent directory
XCWD	Change working directory
XMKD	Make a directory
XPWD	Print the current working directory
XRMD	Remove a directory

The remaining FTP Internet RFC 765 requests are recognized but are currently not implemented.

2. *ftpd* authenticates users according to the normal login procedures for the host system.

3. Commands cannot be aborted.

4. *ftpd* interprets filenames according to the globbing conventions of the normal command interpreter for the host system. The metacharacters * and % can be used.

5. The files are written with the system default protection.

6. The FTP server uses global event flags 50 and 51.

7. The FTP server does not support mkdir.

A.6. PROTOCOL SOFTWARE LOAD UTILITY — NETLOAD

The *netload* utility downloads *net_file* onto the EXOS Ethernet front-end processor board. *net_file* is the EXOS 8031 TCP/IP protocol module, which is supplied by Excelan in executable object form. This utility is invoked from the LB:[1,1]EXOSLOAD.CMD command file. This latter file can be edited to change the *netload* options. Options can be reconfigured any time before downloading the protocol software to the EXOS board; this is automatically done when the host system is rebooted. If no options are specified, *netload* configures TCP/IP with the following defaults:

- Debug messages are not printed.
- The Internet address for the host with the name or alias "localhost" in the hosts file is used.
- The Ethernet address supplied by the board's manufacturer is used.
- The on-board TELNET server is enabled and supports eight connections.
- ARP is enabled.
- The TCP/IP protocol module found in the file LB:[1,2]NET is loaded.

Format

net [**–d**] [**–h** *host*] [**–e** *enet_addr*] [**–t** *n*] [**–l**] [**–m**] [**–r** *n*] [**–x** *n*] [**–p** *n*] [**–i**] [**–c**] [*net_file*]

Options

–c Specifies at what segment address to load code. The code location is specified by a 16-bit number in any format. A zero is appended to the number and the code is loaded at that absolute address. This option is only used for testing *netload*.

–d Displays debugging information. By default, debugging messages are not printed.

–e *enet_addr*
Emulates a board from another manufacturer. *enet_addr* is the Ethernet address to be used instead of the board's default address (which is supplied by Excelan). It must be specified in the standard Ethernet address format – six two-digit hexadecimal numbers separated by hyphens (-).

–h *host_name*
Supplies an explicit hostname or Internet address for the local host, overriding the default name address. *host_name* is the new host address or name. If it begins with a digit, *netload* attempts to convert it to an Internet address in standard dot notation. Otherwise, *netload* looks up the name in the Hosts file and uses the Internet address associated with it.

–i Disables timing out. *netload* times out one minute after the board is reset and the diagnostics are unsuccessful or if an initialization message to the board is unsuccessful. This option is useful for debugging the board's processor or when using an emulator.

–l Enables link access. Incoming packets are then held by the EXOS board in an in box until they are read by the host; this temporarily ties up board resources. It is thus more efficient to run without link-level access.

–m Disables the address resolution protocol (ARP). (By default ARP is enabled.) TCP/IP forms the Ethernet address of the target host by concatenating the three high-order bytes of the local host's Ethernet address with the three low-order bytes of the target host's IP address. The network must use Class A Internet addresses if ARP is disabled. The *–e* option can be used to emulate another Ethernet vendor's address block, and the *–h* option can be used to specify a network number other than the default (89 decimal).

–r *n* Specifies the number of simultaneously blocking host requests, *n*, that are supported. The default is 66, which is large enough for most applications. This number cannot exceed 127 and should be at least as large as the maximum expected number of connections.

–t *n* Specifies *n* TELNET connections that can be supported by the on-board *telnet* server program. *n* can be a number from 0 to 16. (Note that the current 8031 release supports a maximum of 8 connections.) If 0, the server is disabled; then the protocol server can be implemented on the host system.

–x *n* Specifies the number of extended memory buffers, *n*, that are allocated. A minimum of 10 are always allocated, regardless of the requested amount. For 128-Kbyte EXOS boards, specifying a larger number can improve bulk transfer performance but may affect operation when a great many (more than 32) connections are required.

–p Sets the level of diagnostic messages from the board that the host displays. The level is determined by the value of *n*, as follows:

Level	Types of Messages
1	All
2	Errors that are benign (such as retransmissions) or more severe (default)
3	Errors that are nonfatal (such as bad host-requests) or more severe
4	Errors that are fatal, such as NX call failures

net_file Specifies the TCP/IP object module from this file instead of from the default file LB:[1,2]NET.

Privilege Restrictions

You must be a privileged user in order to use this utility.

A.7. ROUTING CONTROL UTILITY — ROUTE

The *route* utility manipulates the network routing table in the memory of the EXOS board. A routing table is created by the *netload* utility when the system is booted. Table entries consist of a host on a remote network or a remote network address and the associated gateway for communication with that host or network.

route accepts three options: add, delete, and show, for adding, deleting, and displaying entries in the routing table. Entries to the routing table can be added in two ways: the system administrator can modify it using the add option, or a gateway can update it.

Format

route add *destination gateway*
route delete *destination gateway*
route show [*destination*]

Options

destination

Name or Internet address of the host on the remote network or of the remote network itself. Internet addresses should be specified in the class used by the remote network. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. If *destination* has a local address part of zero, the route is assumed to be a network; otherwise, it is assumed to be a route to a host. All symbolic names specified for *destination* are first looked up in the host name database (Hosts file). If the address of the host is not found in the routing table, the packets are sent to the network, which then attempts to deliver them to the specified host.

gateway

Name or Internet address of gateway to which the packets should be addressed for delivery to the host on the remote network. Internet addresses should be specified in the class used by the local network. All symbolic names specified for *gateway* are first looked up in the host name database (Hosts file).

route add *destination gateway*

Enters a new route into the routing table. If *destination* is 0 (zero), this route becomes the default route; it is the route used if no other route to a destination is available. Typically, the default route is the address of a gateway.

route delete *destination gateway*

Deletes a route from the routing table.

route show [*destination*]

Displays an individual route or all the routes currently configured. All routes are displayed by omitting *destination*. An individual route is displayed by specifying *destination*.

Privilege Restrictions

You must be a privileged user to change the routing table.

A.8. VIRTUAL TERMINAL EMULATION UTILITY — TELNET

The *telnet* utility allows a host system to emulate a virtual terminal connected to another host that supports the TELNET protocol. If *telnet* is invoked without arguments, it prompts for a host to which a connection is desired. If it is invoked with arguments, *telnet* attempts to establish a connection to the host specified on the command line.

When a connection is established, *telnet* enters input mode. In this mode, typed text is sent to the remote host. This is done in one of two ways at the remote host's discretion. If the remote host volunteers to perform character echo, each character typed is sent to the remote host uninspected (except that end-of-lines are converted to carriage return/line feed pairs). Otherwise, local line editing can be performed, and data are not sent until a complete line is typed. In either case, the line is terminated with the standard *telnet* end-of-line sequence (<cr> <lf>). Similarly, incoming data are assumed to be in chunks of lines and the <cr> <lf> sequence is translated into the local end-of-line marker.

telnet enters command mode when the *telnet* escape character is typed. The default escape character is CTRL-] . In command mode, *telnet* accepts and executes the commands listed below. These commands are processed locally and are not seen by the remote host. Command names can be typed with just enough characters to uniquely identify them.

Format

telnet [–v]
telnet [–v] *remote_host* [*port*]

Options

–v All *telnet* option negotiations are displayed. Options sent by *telnet* are shown as "SENT," while options received from the TELNET server are shown as "RCVD."

remote_host
The host *telnet* attempts to establish a connection with. The host specification can be either a host name or an Internet address (in dot notation). If this argument is not specified, *telnet* prompts for the host to which a connection is desired.

port The remote host's port number. If no port number is specified, *telnet* attempts to contact a TELNET server at the default TELNET port (port 23).

Commands

escape *escape_char*
Sets the *telnet* escape character, the character used to enter command mode. Single-character control characters can be used.

quit Terminates the *telent* session with the remote server, exits *telnet* command mode, and returns to the operating system.

status Displays the current *telnet* status, including name of remote host, the default escape character, and peer and debugging state.

? *command_name*

Displays on-line help about *command_name*. If *command_name* is not specified, all *telent* commands are listed.

CTRL-] Default escape character. An escape character forces *telnet* into command mode.

Privilege Restrictions

No special privileges are required to use this utility.

Notes

1. After *telnet* executes a command, the user is returned to the host operating system, but the operating system prompt is not displayed. The exception is the "?" command: after exectuting this, the "telnet>" prompt is redisplayed.

Appendix B
EXOS 203 AND EXOS 204 INSTALLATION

B.1. INTRODUCTION

The EXOS 203 and EXOS 204 Ethernet front-end processor boards are designed for used in PDP-11 computers. The EXOS 203 is designed for use with the EXOS 8031 software on Q-Bus systems. The EXOS 204 is designed for use with the EXOS 8032 softwar eon UNIBUS systems. This appendix describes the procedures for installing these boards in PDP-11s. The last section of this appendix provides general information about connecting the host system to the network.

The installation and connection procedures described in this appendix assume that you have acquired the required hardware components, including the EXOS connecting cable, transceivers, transceiver cables, and connectors.

B.2. EXOS 203 INSTALLATION

The EXOS 203 is a six-layer DEC quad-sized printed circuit board that interfaces to computers using the Q-Bus, such as DEC's PDP-11/23 and PDP-11/23-PLUS. The installation and connection procedures described in this appendix assume that you have acquired the required hardware components, including the EXOS connecting cable, the transceivers, and the transceiver cables.

This section discusses the system address for the EXOS 203 and the procedure for installing the board in PDP-11 systems.

B.2.1. System Address for EXOS 203

The EXOS 203 uses two two-byte words of address space in the system. The factory set default for this space address is 764000 (octal). Therefore, addresses 764000 and 764002 should not be used by any other device in the system. However, if there should be any contention for this address space, different address space can be jumper-selected for the EXOS 203. The jumpers are described in the *EXOS 203 Ethernet Front-End Processor Reference Manual.* (If the EXOS 203 is assigned a new address, it must be supplied to the EXOS I/O driver at the time the the driver is built during the EXOS 8031 software installation.)

The EXOS 203 also uses one interrupt vector. The EXOS 8031 software expects this vector to be at the address 770 (octal). If there should be a contention for this address a different address can be specified at the EXOS I/O driver build time. No jumper change is required.

B.2.2. Installation Procedure for PDP-11/23-PLUS

A step-by-step description of the procedure for installing the EXOS 203 in PDP-11/23-PLUS minicomputer models is given below.

NOTE

The backplane of the PDP-11/23-PLUS has nine slots, each of which can accept either two dual-width cards or one quad-width card. The EXOS 203 is a quad-width card. The CPU card must be in the top slot of the backplane, and there cannot be any empty slots between the CPU card and the last card in the backplane. If the last slot is occupied by a single dual-width card, this card must be moved down a slot so that the EXOS 203 can be inserted and no empty slots are left between the CPU card and the last card. Note that the bus grant signals move in a serpentine fashion. This means that if you have to move a single dual-width card, you must move it to the dual slot diagonally opposite from it, not to the slot directly below it.

PROCEDURE:

1. Switch off system power and remove the CPU cover panels.
2. Insert the EXOS 203 board in the backplate.
3. Plug the male end of the EXOS cable in to the Ethernet connector on the EXOS 203 board. Assure that the latches on the board have seated properly; this prevents the cable from being pulled out accidentally.
4. Route the EXOS cable to the back of the computer cabinet.
5. Bolt the bracket that holds the Ethernet connector to the I/O panel located at the back of the computer cabinet. If no space is available on the I/O panel, bolt the bracket to one of the vertical bars. (The Ethernet cable from the transceiver on the network will be plugged into this connector.)
6. Ensure that enough +15-volt power is available for the transceiver. Normally, the transceiver gets the power through the EXOS 203 board. Insufficient power for the transceiver can cause your computer to crash.
7. Plug the Ethernet cable from the transceiver in to the Ethernet connector bolted to the I/O panel (or the vertical bar) in Step 5.

Assuming that the network hardware has already been installed, following Step 7 you are ready to install the EXOS 8031 software on your system.

B.2.3. Installation Procedure for PDP-11/23

A step-by-step description of the procedure for installing the EXOS 203 in PDP-11/23 minicomputer models is given below.

NOTE

The backplate of the PDP-11/23 has nine slots, each of which can accept either two dual-width cards or one quad-width card. The EXOS 203 is a quad-width card. The CPU card must be in the top slot of the backplate, and there cannot be any empty slots between the CPU card and the last card in the backplate. If the last slot is occupied by a single dual-width card, this card must be moved down a slot so that the EXOS 203 can be inserted and no empty slots are left between the CPU card and the last card.

PROCEDURE:

1. Switch off system power and remove the CPU cover panels.
2. Insert the EXOS 203 board in the backplate.
3. Plug the male end of the EXOS cable in to the Ethernet connector on the EXOS 203 board. Assure that the latches on the board have seated properly; this prevents the cable from being pulled out accidentally.
4. Route the EXOS cable to the back of the computer cabinet.
5. Bolt the bracket that holds the Ethernet connector to the I/O panel located at the back of the computer cabinet. If no space is available on the I/O panel, bolt the bracket to one of the vertical bars. (The Ethernet cable from the transceiver on the network will be plugged into this connector.)
6. Ensure that enough +15-volt power is available for the transceiver. Normally, the transceiver gets the power through the EXOS 203 board. Insufficient power for the transceiver can cause your computer to crash. In such cases, power the transceiver(s) from a transceiver multiplexer such as a DEC DELNI.
7. Plug the Ethernet cable from the transceiver in to the Ethernet connector bolted to the I/O panel (or the vertical bar) in Step 5.

Assuming that the network hardware has already been installed, following Step 7 you are ready to install the EXOS 8031 software on your system.

B.3. EXOS 204 INSTALLATION

The EXOS 204 is designed for use in PDP-11 minicomputers. It is a six-layer DEC quad-sized printed circuit board that interfaces only to the Small Peripheral Controller (SPC) slots on a UNIBUS back panel using connector rows C, D, E, and F.

This section discusses the system address for the EXOS 204 and the procedure for installing the board in PDP-11 systems.

B.3.1. System Address for EXOS 204

The EXOS 204 uses two two-byte words of address space in the system. The factory set default for this space address is 764000 (octal). Therefore, addresses 764000 and 764002 should not be used by any other device in the system. However, if there should be any contention for this address space, different address space can be jumper-selected for the EXOS 204. The jumpers are described in the *EXOS 204 Ethernet Front-End Processor Reference Manual*. (If the EXOS 204 is assigned a new address, it must be supplied to the EXOS I/O driver at the time the the driver is built during the EXOS 8032 software installation.)

The EXOS 204 also uses one interrupt vector. The EXOS 8032 software expects this vector to be at the address 770 (octal). If there should be a contention for this address a different address can be specified at the EXOS I/O driver build time. No jumper change is required.

B.3.2. Installation Procedure

A step-by-step description of the procedure for installing the EXOS 204 in PDP-11 minicomputer models is given below.

NOTE

In many PDP-11 models, each unused card slot originally comes with small continuity card installed in it. In addition, pins CA1 and CB1 in each unused card slot are wire-linked. However, you can replace any small continuity card with a regular double-width bus grant card and remove the wire link permanently. Whenever a continuity card is removed and an active board is installed, the wire link must also be removed with wire-unwrap tool. Conversely, whenever an active board is removed and a small continuity card re-installed, the wire-link between pins CA1 and CB1 must also be re-installed. Wire link is not needed if a regular double-width bus grant card is re-installed.

PROCEDURE:

1. Switch off system power and remove the CPU cover panels.
2. Remove the small continuity card or the regular bus grant card, as applicable, from the slot in which you wish to install the EXOS 204 board.
3. If present, remove the wire-link between pins CA1 and CB1 in the same slot with a wire-unwrap tool.

4. Insert the EXOS 204 board in the C-D-E-F portion of the card slot emptied in Step 2.
5. Plug the male end of the EXOS cable in to the Ethernet connector on the EXOS 204 board. Assure that the latches on the board have seated properly; this prevents the cable from being pulled out accidentally.
6. Route the EXOS cable to the back of the computer cabinet.
7. Bolt the bracket that holds the Ethernet connector to the I/O panel located at the back of the computer cabinet. If no space is available on the I/O panel, bolt the bracket to one of the vertical bars. (The Ethernet cable from the transceiver on the network will be plugged into this connector.)
8. Ensure that enough +15-volt power is available for the transceiver. Normally, the transceiver gets the power through the EXOS 204 board. Insufficient power for the transceiver can cause your computer to crash. This can happen if a DMF32 multiport board is present or if large power demands are placed on the system power supply by other devices. In such cases, power the transceiver(s) from a transceiver multiplexer such as a DEC DELNI.
9. Plug the Ethernet cable from the transceiver in to the Ethernet connector bolted to the I/O panel (or the vertical bar) in Step 7.

Assuming that the network hardware has already been installed, following Step 9 you are ready to install the EXOS 8032 software on your system.

NOTE

If the EXOS 204 board is removed from its slot, either a double-width bus grant or a small continuity card must be installed in that slot. Furthermore, if a small card is installed, the wire link between pins CA1 and CB1 must also be installed. The wire link is not needed if a regular double-width bus grant card is used.

B.4. CONNECTING TO THE NETWORK

The network hardware should be installed following directions supplied with individual components and in accordance with the Ethernet specifications jointly published by Xerox, Intel, and Digital Equipment Corporation.

Appendix C
QIO FUNCTION CODE SUMMARY

C.1. INTRODUCTION

This appendix summarizes the EXOS QIO function calls. Table C-1 lists the QIO function calls, their modifier symbols and associated values, and a descripton of the calls.

Table C-1: Summary of QIO Function Calls

Function Code/ Modifier Symbol	Value	Description
IO_KIL	000012	Kill all outstanding I/O
IO_WLB	000400	Write to EXOS memory
IO_RLB	001000	Read from EXOS memory
IO_EXC	002400	EXOS board I/O control operation
EX_INI	0	Reset and configure EXOS
EX_STR	1	Start execution of EXOS process
EX_STS	5	Read EXOS statistics
EX_RST	6	Read and reset EXOS statistics
EX_CNF	7	Read configuration message
EX_POS	10	Position EXOS memory locator
EX_SAR	20	Set ARP table entry
EX_GAR	21	Retrive ARP table entry
EX_DAR	22	Delete ARP table entry
EX_ART	23	Add routing table entry
EX_DRT	24	Delete routing table entry
EX_SRT	25	Fetch routing table entry
EX_NRT	26	Fetch next routing table entry
EX_OPN	0020	Open administrative channel
EX_CLS	0021	Close administrative channel
IO_ACS	003000	Socket access operations
SA_OPN	50	Open a socket
SA_ACC	51	Accept connection from remote socket
SA_CON	52	Connect to remote socket
SA_SAD	55	Obtain socket address
SA_CLS	56	Close a socket
SA_SEL	59	Check for I/O readiness
SA_URG	0200	Receive urgent signal
SA_USL	0210	Unselect a socket

Table C-1: Summary of QIO Function Calls (Continued)

Function Code/ Modifier Symbol	Value	Description
IO_XFR	003400	Data transfer operation
IX_SND	53	Send datagram to remote socket
IX_RCV	54	Receive message from socket
IX_RDS	0000	Read data from TCP stream
IX_WRS	0001	Write data to TCP stream
IO_SOC	004000	Socket control operations
SO_DON	0	Halt socket I/O (SIOCDONE)
SO_SKP	1	Enable/disable keep-alives (SIOCSKEEP)
SO_GKP	2	Show keep-alive status (SIOCGKEEP)
SO_SLG	3	Enable/disable lingering (SIOCSLINGER)
SO_GLG	4	Show lingering status (SIOCGLINGER)
SO_SOB	5	Send out-of-band data (SIOCSNDOOB)
SO_ROB	6	Receive out-of-band data (SIOCRCVOOB)
SO_AMK	7	Distinguish urgent data (SIOCATMARK)
SO_SPG	8	Set process group (SIOCSPGRP)
SO_GPG	9	Get process group (SIOCGPGRP)
SO_NRD	127	Return count of data in socket's receive buffer (FIONREAD)
SO_NBO	126	Enable/disable nonblocking I/O (FIONBIO)

Appendix D
QIO ERROR STATUS CODES

D.1. INTRODUCTION

Two broad categories of error messages are returned as the result of a QIO request: messages that indicate whether the queuing request was successful, and messages that indicate whether the I/O request itself was successful.

The success or failure status of the queuing request is indicated by the Directive Status Word ($DSW) that is returned following a QIO directive.

The success or failure of the I/O operation itself is indicated in the IOSB. If unsuccessful, the low-order byte of the IOSB's first word contains error codes reported by the Executive/Driver and/or the high-order byte contains error codes reported by the EXOS board, depending on the nature of the error.

This appendix lists the error messages associated with these two categories of errors and explains the causes of the errors.

D.2. QUEUING REQUEST ERROR CODES

Queuing request error codes are returned in the Directive Status Word ($DSW). The EXOS 8031 software subtracts 512 from the standard RSX system error codes and displays the error as a numerical code, if possible, and the message "UNSPECIFIED ERROR." To determine the exact error code, add 512 to the displayed error code and refer to the *RSX-11M/M-PLUS Executive Reference Manual.*

D.3. I/O OPERATION ERROR CODES

The first word of the IOSB contains error codes related to the unsuccessful completion of the I/O request.

The low-order byte of the IOSB's first word contains error codes reported by the Executive/Driver. The EXOS 8031 software subtracts 512 from the standard RSX driver (ACP) error codes. It displays the error as a numerical code, if possible, and the message "UNSPECIFIED ERROR." To determine the exact error code, add 512 to the displayed error code and refer to the *IAS/RSX-11 I/O Operations Reference Manual.*

The high-order byte of the IOSB contains error codes reported by the EXOS board. These error codes are explained below.

Error Code and Symbolic Name	**Explanation**
0	The I/O operation was successful.
1 – EPERM	Not owner. The EXOS board does not own the buffer to which the host message queue header points.
2 – ENOENT	No such file or directory.

Error Code and Symbolic Name	**Explanation**
3 – ESRCH	No such process. This may be returned by *route* if no route can be found in response to a BRDSHOWRT (show route) or BRDDISPRT (display route) request.
4 – EINTR	Interrupted system call. This may indicate that a request is pending on a socket and out-of-band data have arrived.
5 – EIO	I/O error.
6 – ENXIO	No such device or address. This may be returned by *arp* if the entry specified by a BRDGARP (get ARP entry) or BRDDARP (delete ARP entry) is not found in the cache.
7 – E2BIG	Arg list too long. Too many arguments were specified with the function.
8 – ENOEXEC	Exec format error.
9 – EBADF	Bad file number. A close request was issued for a nonexistent socket.
10 – ECHILD	No children.
11 – EAGAIN	No more processes.
12 – ENOMEM	Not enough core.
13 – EACCES	Permission denied. Insufficient privilege exists to perform the operation. Specifically, an attempt was made to open a socket without sufficient privileges, or a raw, link-level, or UDP/TCP socket was specified with a port number less than 1024 and without sufficient privileges.
14 – EFAULT	Bad address
15 – ENOTBLK	Block device required.
16 – EBUSY	Mount device busy.
17 – EEXIST	File exists. (1) A bad request code was passed to the routing code (on the EXOS board). (2) An attempt was made to delete the entry for the default gateway.

Error Code and Symbolic Name	**Explanation**
18 – EXDEV	Cross-device link.
19 – ENODEV	No such device.
20 – ENOTDIR	Not a directory.
21 – EISDIR	Is a directory.
22 – EINVAL	Invalid argument. The I/O request contained an invalid argument. This occurs when an active socket created without the SO_ACCEPTCONN option accepts a request for connection.
23 – ENFILE	File table overflow.
24 – EMFILE	Too many open files.
25 – ENOTTY	Not a typewriter.
26 – ETXTBSY	Text file busy.
27 – EFBIG	File too large.
28 – ENOSPC	No space left on device.
29 – ESPIPE	Illegal seek.
30 – EROFS	Read-only file system.
31 – EMLINK	Too many links.
32 – EPIPE	Broken connection. I/O to a closed connection was attempted.
33 – EDOM	Argument too large.
34 – ERANGE	Result too large.
35 – EWOULDBLOCK	Operation would block. An I/O operation that would normally have blocked (such as receive, send, or accept) cannot block because FIONBIO is set, and thus cannot complete immediately because nonblocking I/O is in effect. No data are transferred, and the I/O operation may be attempted again until the I/O actually occurs.

Error Code and Symbolic Name	**Explanation**
36 – EINPROGRESS	Operation now in progress. This is returned by connect and close requests when FIONBIO (blocking I/O) has been set. With a connect request, try the request again. With a close request, the socket will eventually be closed so no further action is required.
37 – EALREADY	Operation already in progress. An attempt was made to close the same connection twice or to close a connection that was in the process of being closed (TCP).
38 – ENOTSOCK	Socket operation on non-socket.
39 – EDESTADDRREQ	Destination address required. No destination address was specified with a send or write operation.
40 – EMSGSIZE	Message too long. The datagram sent is larger than allowed by the EXOS board (for UDP and raw sockets).
41 – EPROTOTYPE	Protocol wrong type for socket.
42 – ENOPROTOOPT	Protocol not available.
43 – EPROTONOSUPPORT	Protocol not supported. (1) The socket type and socket protocol fields in the SOioctl structure do not agree (see Table 5-2). (2) For a SOSOCKET request, neither the socket type nor the socket protocol match a supported protocol.
44 – ESOCKTNOSUPPORT	Socket type not supported.
45 – EOPNOTSUPP	Operation not supported on socket. (1) The Internet address field of the EXarp structure is not recognized. An invalid address was specified in a socket call. (2) An attempt was made to send out-of-band data on a datagram socket. (3) The socket control code was unrecognized.
46 – EPFNOSUPPORT	Protocol family not supported.

Error Code and Symbolic Name	**Explanation**
47 – EAFNOSUPPORT	Address family not supported by protocol family. An invalid family was specified in the socket protocol field of the SOioctl structure (see Table 5-2).
48 – EADDRINUSE	Address already in use. This is returned by a socket call when an already existing socket is using the same (UDP/TCP) port number. This condition is often transient, because once a connection is established the local address can be reused. Either wait for the socket to become available or retry, specifying a different port.
49 – EADDRNOTAVAIL	Cannot assign requested address. An invalid network number may have been specified in a socket request.
50 – ENETDOWN	Network is down.
51 – ENETUNREACH	Network is unreachable. A connect or send request has been given, but there is no known route to the remote host.
52 – ENETRESET	Network dropped connection on reset.
53 – ECONNABORTED	Software caused connection abort. Something probably went wrong at the remote end of the connection and the local system detected it (TCP only).
54 – ECONNRESET	Connection reset by peer. Something probably went wrong at the remote end of the connection and the remote system detected it (TCP only).
55 – ENOBUFS	No buffer space available. No buffers are available on the EXOS board.
56 – EISCONN	Socket is already connected. This is a response to a connect request that attempts to connect to an already connected socket.
57 – ENOTCONN	Socket is not connected. A write or read request was attempted on a socket that had not been connected.
58 – ESHUTDOWN	Can't send after socket shutdown.

Error Code and Symbolic Name	**Explanation**
59 – ETOOMANYREFS	Too many references: can't splice
60 – ETIMEDOUT	Connection timed out. No response was received from the remote system. Either the remote system is not functioning or there is an addressing problem.
61 – ECONNREFUSED	Connection refused. No accept was issued by the remote system for a connect request.
62 – ELOOP	Too many levels of symbolic links.
63 – ENAMETOOLONG	File name too long.
64 – EHOSTDOWN	Host is down.
65 – EHOSTUNREACH	No route to host. There is no route to the specified host, and there may be no default gateway either.
66 – SYSERR	Unspecified operating-system-specific error.

Appendix E
UTILITY ERROR MESSAGES

E.1. INTRODUCTION

The network application utilities provided with the EXOS 8031 TCP/IP software – *ftp* and *telnet* – return two types of error messages. Those preceded by three uppercase letters and two hyphens are RSX errors. For an explanation of these messages, refer to the appropriate RSX manual. Error messages not sent by the operating system are due to a malfunctioning of either the network software or the utility itself. Most of the error messages result from problems with the network software and should never appear when you are using the utilities. If they do appear, however, they indicate a serious problem that should be resolved by the system administrator.

This appendix lists and explains the error messages that may be displayed when you are using *ftp* and *telnet*.

E.2. FTP ERROR MESSAGES

Error messages that appear while you are using *ftp* may originate either from the remote host or from the local host.

Error messages that are preceded by three digits originate from the remote host. They indicate that an attempt was made to perform an invalid operation on the remote host. These messages should be self-explanatory; for further explanation, consult the documentation for the remote system.

The error messages that originate from the local host are listed in alphabetical order and explained below. If any other messages appear that are not self-explanatory and that interrupt your work, contact your system administrator.

Error Message	Explanation
Already connected to ..., use close first.	You must close the existing connection before you can open another.
Ambiguous command	The command you typed can be interpreted in more than one way. Use the *help* command for a list of valid commands.
Connection refused	The remote server is too busy or it does not have the proper server software. In the first case, try establishing the connection later. In the second case, contact your system administrator or the administrator of the remote system.
Connection reset by peer	Something probably went wrong at the remote end of the connection and the remote system detected it. Contact your system administrator or the administrator of the remote system.

Error Message	Explanation
Connection timed out	The attempt to communicate with the remote host did not succeed. Most likely, the remote host is down or the path to it has been severed. Try again later or contact your system administrator.
Invalid command	The command you typed was not valid. Use the *help* command for a list of valid commands.
Login failed	The name and password you supplied to the remote system were not acceptable.
Network dropped connection on reset or Lost connection	Communication to the remote host was interrupted and lost, and the state of the operation that was in progress, if any, cannot be determined. Try the operation again.
Network is down	The local host cannot communicate with the network. Try again later.
Network is unreachable	No route to the remote host and/or the remote network is known. Contact your system administrator.
No route to host	There is no route to the remote host. Contact your system administrator or the administrator of the remote host.
No such file or directory	You attempted to access a nonexistent file or directory.
Not connected	File operations cannot be performed until you have established a connection to the remote host. Use the *open* command to do this.
Not sufficient privilege for operation or Permission denied	You attempted an operation for which you do not have the proper permission(s).
... : Unknown host	The host you specified is not in the hosts database, the file that lists the remote systems that you can connect to from your system. Contact your system administrator.
usage: ...	You specified options incorrectly on the command line. This error message shows the correct command line format. For more information, refer to the section on *ftp* in Appendix A.

E.3. TELNET ERROR MESSAGES

All error messages that occur when you are using the *telnet* command originate from the local host. They are listed in alphabetical order and explained below. If any other messages appear that are not self-explanatory and that interrupt your work, contact your system administrator.

Error Message	Explanation
Connection refused	The remote server is too busy or it does not have the proper server software. In the first case, try establishing the connection later. In the second case, contact your system administrator or the administrator of the remote system.
Connection reset by peer	Something probably went wrong at the remote end of the connection and the remote system detected it. Contact your system administrator or the administrator of the remote system.
Connection timed out	The attempt to communicate with the remote host did not succeed. Most likely, the remote host is down or the path to it has been severed. Try again later or contact your system administrator.
Host is down	The remote host is not functioning. Contact your system administrator or the administrator of the remote host.
Invalid command	The command you typed was not valid. Use the *help* command for a list of valid commands.
Network dropped connection on reset	Communication to the remote host was interrupted and lost, and the state of the operation that was in progress, if any, cannot be determined. Try the operation again.
Network is down	The local host cannot communicate with the network. Try again later.
Network is unreachable	No route to the remote host and/or the remote network is known. Contact your system administrator.
No route to host	There is no route to the remote host. Contact your system administrator or the administrator of the remote host.
Not sufficient privilege for operation or Permission denied	You attempted an operation for which you do not have the proper permission(s).

Error Message	Explanation
Software caused connection abort	Something probably went wrong at the remote end of the connection and the local system detected it. Contact your system administrator.
... : Unknown host	The host you specified is not in the hosts database, the file that lists the remote systems that you can connect to from your system. Contact your system administrator.
usage: ...	You specified options incorrectly on the command line. This error message shows the correct command line format. For more information, refer to the section on *telnet* in Appendix A.

Appendix F
TROUBLESHOOTING

F.1. INTRODUCTION

Most problems reported by end users can be resolved by reference to Appendix E, where the error messages displayed by the utilities are explained. More severe problems are generally reported to the system console and must resolved by the system administrator. This appendix provides guidance for the system administrator in resolving common faults in the operation of networking software.

The system administrator has access to more diagnostic facilities than other users do. These facilities include the following:

- System console. The EXOS 8031 logs general error messages on the system console. The conditions reported on the console include suspicious situations detected by the front-end processor. Refer to the section on replies in the RSX *System Management Guide* for information on how to redirect and/or disable the logging of error messages.
- The *bstat* utility. This utility, described in Section 3.6.2 and in Appendix A, reports statistics on low-level communication activity. This information contains clues about problem conditions.
- *route* and *arp* utilities. These utilities allow the system administrator to inspect and alter tables in the front-end that affect the transmission of packets over the network. *route* is described in Section 3.6.4 and *arp* in Section 3.6.1; both are detailed in Appendix A.

F.2. INSTALLATION PROBLEMS

If problems arise during the installation and startup of EXOS 8031, the most likely point is when "netstart" is invoked. Symptoms can vary. If the problem can be diagnosed by RSX, a message appears on the console; an example is an "initialization failed" message. These error messages are explained in the RSX documentation. More serious situations can have consequences as severe as crashing the system.

Typical causes to investigate include the following:

- The front-end board was not properly installed. Read Appendix B carefully, especially the notes which indicate common trouble spots.
- The values typed during installation for UNIBUS adapter number, CSR address or interrupt address did not match the hardware configuration. The consequences could be that the host attempted to manipulate the wrong device or a nonexistent device.

To ascertain the front-end processor's status during initialization, examine the LEDs mounted on the board. They flash status codes, which are explained in the EXOS 203 and EXOS 204 *Ethernet Front-End Processor Reference Manuals.*

F.3. USER MESSAGES

Most user messages can be resolved without extensive investigation. However, three messages require some explanation.

No route to ...

This message from the EXOS board indicates that no route to the remote host can be found. This can happen if two hosts on the *same* network were assigned Internet addresses containing *different* network numbers. This can also happen if the remote host is on a different network but you have not added the proper route entry using the *route* command. Check the routing tables and Hosts file to resolve the incompatibility. Also make sure that the systems use the same network number.

If the remote system is from Sun Microsystems, be aware that its default network number is 193, while Excelan's is 89. Change one or both to make them compatible.

Connection refused

This message typically indicates that even though a path was established to the remote host, the remote host is not prepared for the type of connection attempted. Therefore, the connection was refused. The most likely cause is that the server (daemon) has not been enabled on the remote host. Servers are normally enabled automatically when a system is booted. Check with the system administrator of the remote host.

Connection timed out

This is a more serious problem than a refused connection. It means that low-level packet transmission could not be accomplished in one or both directions. Either the connection request or its reply failed to get through. To isolate the problem, first determine if it occurs for all communicating partners or for just a specific one. Bearing this in mind, examine the history of console messages from EXOS. The next section details the console messages.

F.4. CONSOLE MESSAGES

Two types of information appear on the system console. When the EXOS front-end board is initialized, it reports a variety of configuration and version information, including the software/firmware/hardware versions and the Internet and Ethernet addresses that the networking software thinks are its own. If problems are suspected, addresses should be checked for consistency with the Hosts file and with the other hosts on the network.

The second category of console messages consists of warnings that occur during system operation. The following messages can appear; they provide clues to various trouble situations.

EXOS CODE 0001

This occurs when the TCP checksum calculated by the EXOS board for an incoming packet does not agree with the checksum in the packet header. Most likely, checksums were disabled or incorrectly calculated on the system that sent the packet. Otherwise, the packet may have suffered a transmission error (very unlikely on Ethernet) or the EXOS TCP checksum calculation was in error. Check the intercommunicability

of various hosts with each other to determine which is malfunctioning.

EXOS CODE 0003 rxmt time = nnn

This indicates that the front-end was forced to retransmit packets due to a lack of response. It is normal for this to happen with *nnn* = 2 the first time you connect with a host. Trouble is indicated if it happens repeatedly with increasing values of *nnn*. The most likely causes are the following:

- The host with which communication is being attempted is down or is unable to communicate. Check if that host can communicate with another system.
- The IP address of the remote host is not properly entered in the Hosts file. You should ensure that both hosts agree on what each others' Internet addresses are. Check the following places for consistency: the Hosts files of the two machines and the addresses that the networking software believes to be their own. In the case of EXOS software systems, the address that the networking software believes to be its own is shown on the operator's console when the network module is started on the front-end board.
- There is an ARP mismatch. All hosts on a network must agree on whether ARP is in use. EXOS supports both alternatives, but other systems may not. Consult Section 3.6.1 and Appendix A for a description of ARP usage. Check the documentation of the other host's networking software to determine if the host supports ARP and, if so, how to enable/disable it. If the remote host does not understand ARP, you can use the ARP command to set up the address of the remote host in the ARP cache manually. Alternatively, you can operate the entire network without ARP; see Section 3.6.3.4 for information on mixing Ethernet boards from different vendors on the same network.
- If the remote computer is running the UNIX 4.2BSD operating system, beware of the following problem. As with EXOS, some 4.2BSD systems let you use "old style" mapping, in which Internet addresses are converted to Ethernet addresses by taking the low-order three bytes of the Internet address and appending them to the high-order three bytes of the Ethernet address. Unlike EXOS, UNIX 4.2BSD makes the decision based on the local host number (which is extracted from the Internet address). If this host number is greater than or equal to *oldmap*, old style mapping is used; otherwise ARP is used. *oldmap* is a variable inside the 4.2BSD operating system. Current 4.2BSD systems set *oldmap* to 1024. To circumvent this problem, you can either force your Internet address to meet the above constraint (that is, choose your local host number to be less than 1024) or modify the *oldmap* variable in the UNIX 4.2 kernel to be the largest positive number on the 4.2 system. The *adb* utility can be used to change *oldmap*.
- A bad routing table entry may be causing packets for the intended host to be launched to the wrong host or to a gateway that is either nonexistent or out of service. Use the *route* utility to examine the current state of the routing table. Check gateways for proper operation. Be aware that routing table updates can originate not only from the operator (using the *route* utility) but also as a result of

"redirect" packets sent from a gateway.

- Be aware that for two host to communicate, *both* their routing tables need to contain paths to the other.

EXOS CODE 0102 xmit err 10

This message appears when the EXOS board is having trouble transmitting. This typically happens because the transceiver is not properly connected to the Ethernet cable. This causes excessive collisions. Ensure that the transceiver is properly connected.

Another possible cause is a general problem with the network, such as a short circuit.

EXOS CODE 0102 xmit err 04

This message indicates DMA underrun for the Ethernet chip. It can be reported by NX revisions greater than 4.7. If this problem persists, report it to Excelan.

EXOS CODE 0102 xmit err 20

This message indicates a problem in the attachment of the local node to the Ethernet cable. The symptom is lack of carrier sense curing transmission. Review the material in Section B.3 that describes how to connect to the network.

EXOS CODE 0102 xmit err CB

This message results from one of two causes. Either the attachment to the network is faulty (as in the preceding paragraph) or there is a hardware fault on the front-end processor. The symptom is a timeout during transmission.

EXOS CODE 0105 recv err04

This indicates a DMA overrun within the front-end configuration. If it happens persistently and interferes with system operation, report the problem to Excelan.

EXOS CODE 0105 recv err 10

A packet longer than the specification permits was received.

EXOS CODE 0106
EXOS CODE 0107

An excessively long "trailer" packet was received. UNIX 4.2BSD networking produces trailer packets. These codes indicate that the packets are being produced incorrectly. Pursue the problem with the system(s) that might be transmitting the packets.

EXOS CODE 0115

No route exists to a correspondent. See Section F.3.

EXOS CODE 0207
EXOS CODE 0208
EXOS CODE 0616

These messages indicate a shortage of data buffers on the front-end processor. They may occur if many connections are actively transferring data concurrently, especially if the host stops reading data for numerous connections, forcing data to accumulate on the front end. If this situation persists in normal operation, add memory or reduce the number of active connections. In any event, arrange for host

applications to read input data from the board most of the time.

EXOS CODE 0301 *parameters*

A host socket request contained illegal parameters such that it failed to specify a protocol that EXOS supports. *parameters* are displayed in hexadecimal in the following order:

- Type of protocol within a protocol family
- Protocol family
- Protocol within family
- Socket options
- Internet protocol family
- Internet TCP/UDP port
- Internet socket address

To find the source of the problem, investigate application programs attempting to open sockets.

EXOS CODE 0408 nnn

The EXOS software does not recognize a request code received from the host. There is probably a bug in the host's driver software. The number *nnn* is the improper request code. To find the source of the problem, investigate application programs trying new things. If you are using only Excelan-provided software, report this error to Excelan as a driver or utility problem.

EXOS CODE 0705
EXOS CODE 0706

These are checksum calculation mismatches similar to EXOS CODE 0001. Code 0705 pertains to the ICMP protocol, code 0706 to IP.

EXOS CODE xxxx

If codes that are not listed above are displayed and the EXOS is not working correctly, report the problem to Excelan.

CIRCUIT WAS RESET

This is actually a report from the host-resident device driver. It indicates that a connection was closed because an administrator caused the front-end processor to be reinitialized.

F.5. ABSENCE OF CONSOLE MESSAGES

If there are no messages on the system console to indicate the problem, investigate by looking at the traffic statistics of both noncommunicating partners. On any current EXOS system, use the *bstat* command (see Section 3.6.2 and Appendix A). If the remote system is running UNIX 4.2BSD, use its *netstat –i* command. Bracket the communication attempt with calls to these commands. For each host, you will then be able to tell if it is successfully sending and/or receiving. You will also be able to note if there are excessive numbers of abnormal conditions occurring, such as alignment errors, parity errors, and excessive collisions. All these situations occur occasionally; if one dominates the statistics, it points to the problem.

F.5.1. Alignment Errors

This error condition occurs because not all Ethernet controller boards are compatible with Version 2.0 of the Ethernet specification. In particular, the specification states that "alignment errors" are acceptable as long as the

(Ethernet) checksum is correct. (In the previous version of the Ethernet specification, indication of an alignment error was sufficient grounds to discard the packet.) This can be a problem since certain (transmitting) boards that use Seeq chips can generate an alignment error on the receiving side. If the specification is not totally met, this can cause packets to be discarded.

If the remote host is using an Interlan board (in a link-level mode), the board may be incompatible with early Excelan boards. Some older EXOS versions can cause alignment errors to be reported by the Interlan board. As a result, packets sent by the Excelan board are discarded by the Interlan board, but packets sent by the Interlan board are accepted by the Excelan board.

If this problem persists after investigating the possibility of a hardware failure or installation error, upgrade the Ethernet board(s) to a newer revision.

F.5.2. Excessive CRC Errors

This is most likely a hardware malfunction in one of the systems. The checksum on packets being transmitted is failing the test on receipt. Try communicating among a variety of hosts to see if one is unable to communicate with any others. This host is the likely culprit.

F.5.3. Excessive Collisions

Most likely, one station on the network is suffering a hardware malfunction causing it to continually assert carrier, in violation of Ethernet specifications. If this is really the problem, you will notice that no systems will be able to communicate successfully. Try selectively disconnecting hosts from the network to isolate the culprit.

F.5.4. SQE Test Failures

If you are using an Ethernet Version 1.0 transceiver, you have probably jumpered the EXOS board for responding to SQE tests; this can only be done by Ethernet Version 2.0. If you are using an Ethernet Version 2.0, an SQE test failure indicates that there is no heartbeat. Other possible causes are a cabling fault, a transceiver fault, or an EXOS hardware problem.

F.5.5. UDP and UNIX 4.2BSD

If you are using a UDP datagram protocol (not FTP or TELNET – these are TCP-based) and the remote machine is running UNIX 4.2BSD, there is a possible checksum problem. In early versions of 4.2, there was a bug in the UDP checksum calculation algorithm. Obtain a new version of software from your 4.2BSD supplier.

F.5.6. DMA Underrun

If this occurs frequently, report the problem to Excelan.

F.5.7. No Receive Buffers

This means there is insufficient data buffer space on the board for the level of activity. Add memory or reduce the number of active connections.

F.5.8. Duplicate IP Address

If you are communicating with a 4.2BSD system, this message occurs on the 4.2BSD system when the Hosts file from one host is copied to another system and the *localhost* entry in the file is not modified to reflect the new host. As a result, *netload* uses the same IP address (indexed by *localhost*) as the system from which the file was copied.

If you copy the Hosts file from another system, be sure to update it so that the local IP address has *localhost* on the proper line.