ETHERMON

V2.4-1

29-Oct-1992

written by

Andrew C. Pavlin

General Electric Co.

Aerospace Information Technology

Syracuse, NY 13221

Abstract

ETHERMON is a software implementation of an Ethernet line activity monitor. It permits a suitably privileged user to observe the flow of data across the Ethernet, as seen from any DEC-standard Ethernet adapter connected to a VAX/VMS system. The data can be observed statistically in several forms, or can be recorded to a disk file for additional off-line analysis. The received data can be selectively filtered to remove messages not of interest. When possible, Ethernet addresses are displayed symbolically in addition to the physical hardware representation.

1. Requirements for Use

To operate, ETHERMON requires a VAX/VMS system at version 4.x or higher, and at least one Ethernet or FDDI interface that uses a driver compatible with the Ethernet driver specifications in the *VMS I/O User's Reference Manual: Part II* (DEUNA, DELUA, DEQNA, DELQA, DEBNT, etc.). The user account running ETHERMON must have PHY_IO, NETMBX, and TMPMBX (or SETPRV) privileges, and a buffered I/O byte limit quota (BYTLM) of at least 16000 (depending on the value of certain SYSGEN parameters; see section I.1 on page 62). For systems running VAX/VMS V5.4-3 and above, CMKRNL privilege is also required; it is preferred for systems at lower versions.[1]The system must not be running any application using promiscuous mode on the selected Ethernet interface (if running in fallback Ethernet-only mode, there must be no other protocol type FF-FF user on the Ethernet interface; see section I.2 for more details).

Note that only one copy of ETHERMON can be running on an Ethernet interface at a time (due to a driver limitation on the number of simultaneous promiscuous Ethernet users); however, any number of *other* applications (DECnet, TCP/IP, etc.) may be running on an Ethernet interface at the same time as ETHERMON (as long as those applications do not create a promiscuous user).

Note also that, if your Ethernet has any kind of serious load on it, ETHERMON will tend to be a CPU hog. The program has been tuned, but the vast majority of its CPU time is spent in the VMS I/O executive and the Ethernet interface I/O driver (per VAX SPM analyses), which can't be "tweaked" for better performance. It is therefore recommended that you use ETHERMON on the most powerful CPU you have available, in order to prevent data loss in ETHERMON's analysis. Work is being done to attempt to correct this problem, but the work has not been completed as of this release.

One final limitation is that ETHERMON cannot see any messages being transmitted by the local system through the selected Ethernet interface (the loopback buffer isn't big enough for ETHERMON to bother turning it on, and most interfaces can't run in loopback or echo without also turning off real transmissions to and receptions from the actual cable). Bear this in mind when you interpret the received data.

1.1. Statement of Liability

As usual, you use this program **at your own risk**; neither the author, the author's employer, DECUS, or anybody but yourself is liable for any problems caused by this program. The information used to construct this program was culled from many sources, not all of which could be considered to be 100% reliable; the author has tried for the best accuracy and reliability possible, but some of the data and logic in this program is based on conjecture. If you have any comments or suggestions as to improvements, corrections, etc., please contact me regarding them. As of 6 March 1992, I can be reached at:

[1]For more information regarding this new privilege requirement for ETHERMON V2.4 (needed to implement the new "fast" port interface), please see section 3.1 describing ETHERMON's *modus operandi* on page 11.

```
General Electric Company
Electronics Park, Bldg. 7, Room C67
Syracuse, NY 13221 U.S.A.
telephone: 1-315-456-6212
           8*256-6212 (GE/RCA DIALCOMM)
fax:       1-315-456-0956
GE DECnet: C5VR::PAVLIN
Internet: PAVLIN@syr.ge.com
          PAVLIN@c5vr.dnet.ge.com
```


1.2. Acknowledgements

DEC, DECnet, DECbridge, DECserver, LAT, UCX, Ultrix, VAX, VAXcluster, and VMS are trademarks of Digital Equipment Corporation.

ARPANET, Internet, and TCP/IP are trademarks of the U.S. Government Department of Defense.

MultiNet is a trademark of SRI International and TGV, Inc.

MULTIBUS II is a trademark of Intel Corporation.

Scribe is a trademark of Unus, Inc.

SUN and NFS are trademarks of Sun Microsystems Inc.

U-B, NIU, and Access/One are registered trademarks of Ungermann-Bass, Inc.

AT&T and Unix are trademarks of American Telephone & Telegraph Co.

Xerox, XNS, and PUP are trademarks of Xerox Corporation.

HIU, CC, and Xyplex are trademarks of Xyplex Corporation.

Macintosh, AppleTalk, and EtherTalk are trademarks of Apple Computer, Inc.

PostScript is a registered trademark of Adobe Systems Incorporated.

HP is a trademark of Hewlett-Packard.

All other manufacturer names are those manufacturers’ trademarks.

2. Installing the Program

2.1. Manifest

ETHERMON consists of several FORTRAN subroutines, a set of MACRO routines and a syntax table, a file of error message definitions, a VAX/VMS help library, and documentation files. The complete set of files composing ETHERMON contains:

1. the FORTRAN source files and INCLUDE files defining COMMON blocks:
    - ETHERMON.FOR - initialization and support routines source file
    - ETHERMON_ETHREAD.FOR - Ethernet read routines source file
    - ETHERMON_CMD_AL.FOR - commands (from @ to L) source file
    - ETHERMON_CMD_MR.FOR - commands (from M to R) source file
    - ETHERMON_CMD_S.FOR - commands (S) source file
    - ETHERMON_CMD_TZ.FOR - commands (from T to Z) source file
    - ETHERMON_ADDR.INC - Ethernet address table
    - ETHERMON_BRIDGES.INC - bridge table and bridge interconnection map
    - ETHERMON_CMD.INC - control flags related to command execution (not parsing)
    - ETHERMON_DEV.INC - device and system names
    - ETHERMON_ETH.INC - raw Ethernet data buffers and port parameters
    - ETHERMON_FILTER.INC - data filtering table
    - ETHERMON_MANUF.INC - manufacturer prefix numbers and names table
    - ETHERMON_PING.INC - PING command flags and data buffer
    - ETHERMON_PROT.INC - protocol type numbers and names table
    - ETHERMON_PRS.INC - parameters from command line parsing
    - ETHERMON_SMG.INC - pointers and control information for use of the Screen Management Facility (SMG)
    - ETHERMON_STATS.INC - counters for message statistics
    - ETHERMON_TCPIP.INC - protocol numbers and names for the DoD Internet protocol suite
    - ETHERMON_TRACE.INC - control flags and RMS blocks for TRACE command
2. ETHERMON_PARSE.MAR - the MACRO source file containing the LIB$TPARSE state table and assorted support action routines.
3. ETHERMON_FASTCALL.MAR - the MACRO source file providing the run-time link access to the "fast" port interface.
4. ETHERMON_FASTSHR.MAR - the MACRO source file containing the kernel-mode access routines used to implement the "fast" port interface.
5. ETHERMON_MSG.MSG - the MESSAGE source file (containing the error message definitions).
6. ETHERMON.COM - a command prcedure which compiles and links the ETHERMON program, creates the ETHERMON help library, and attempts to build the unsupported

utilities (see section 4 and the list of unsupported off-line utility files below).

7. On-line documentation:
    - ETHERMON.HLP - the HELP command source text (used to build the VMS help library file).
    - ETHERMON.HLB - the VMS help library file for ETHERMON.
8. Off-line documentation:
    - ETHERMON.MSS - the user’s guide (this document) in Scribe manuscript source format.
    - ETHERMON.LN0 - the user’s guide in DEC LN03 laser printer output format.
    - ETHERMON.LPT - the user’s guide in "dumb" line printer output format (note that this document version will not contain illustrations).
    - ETHERMON.PS - the user’s guide in PostScript output format.
    - ETHERMON_MSS_KEYDEF.PS - illustration of default state keypad definitions for PostScript version of user’s guide (automatically included into ETHERMON.MSS by Scribe).
    - ETHERMON_MSS_KEYGOLD.PS - illustration of GOLD state keypad definitions for PostScript version of user’s guide (automatically included into ETHERMON.MSS by Scribe).
9. Distribution tools:
    - ETHERMON_SEND.COM - a command procedure to help in distributing ETHERMON by electronic mail (e-mail); currently, it only supports DECnet mail and Internet mail using the TGV MultiNet product’s foreign mail interface.
    - AAAREADME.TXT - the introductory file for the ETHERMON package.
10. Off-line unsupported utilities:
    - ETHTRSTRUC.INC - FORTRAN include file describing the structure of a binary TRACE file’s records
    - ETHERPARSE.FOR - FORTRAN source file of basic ASCII formatter for binary TRACE files
    - ETHERPARSE2.FOR - FORTRAN source file of enhanced ASCII formatter for binary TRACE files
    - ETHERPARSE2_CRF.MAR - MACRO source file of support code and data for ETHERPARSE2
    - ETHERPARSE_TCP.FOR - FORTRAN source file of alternate ASCII formatter
    - ETHERPARSE_XREF.FOR - FORTRAN source file of alternate ASCII formatter
    - ETHFILT_ETHADDR.FOR - FORTRAN source file for post-filter
    - ETHFILT_IPADDR_TGV.FOR - FORTRAN source file for post-filter
    - ETHFILT_IPADDR_UCX.FOR - FORTRAN source file for post-filter
    - ETHFILT_NFS.FOR - FORTRAN source file for post-filter
    - ETH_CHECK_SUBPROC.FOR - FORTRAN source file for support subroutine: verifying if a process is a TRACE/ANALYZER subprocess
    - ETH_MAP_ADDRTBL.FOR - FORTRAN source file for support subroutine: connecting a TRACE/ANALYZER subprocess to ETHERMON’s address table global

section

- ETHFILT_IPADDR_TGV.OPT - linker options file for post-filter
- ETHFILT_IPADDR_UCX.OPT - linker options file for post-filter
- ETHERMON_ALL.COM - sample command procedure for batch-mode data collection

2.2. Installation Procedure

To prepare ETHERMON for use, the program must be compiled and linked, and its associated help library must be built; optionally, the program can be defined as a DCL foreign command.

Building the program means, first of all, all the source files must be in the same directory, and that directory must be the current default directory. Then issue the following commands:

```
$  FORTRAN/OPTIMIZE/CONTINUATIONS=99 ETHERMON
$  FORTRAN/OPTIMIZE/CONTINUATIONS=99 ETHERMON_ETHREAD
$  FORTRAN/OPTIMIZE/CONTINUATIONS=99 ETHERMON_CMD_AL
$  FORTRAN/OPTIMIZE/CONTINUATIONS=99 ETHERMON_CMD_MR
$  FORTRAN/OPTIMIZE/CONTINUATIONS=99 ETHERMON_CMD_S
$  FORTRAN/OPTIMIZE/CONTINUATIONS=99 ETHERMON_CMD_TZ
$  MACRO ETHERMON_PARSE
$  MACRO ETHERMON_FASTCALL
$  MESSAGE ETHERMON_MSG
$  LINK ETHERMON,-
$_        ETHERMON_ETHREAD,-
$_        ETHERMON_CMD_AL,-
$_        ETHERMON_CMD_MR,-
$_        ETHERMON_CMD_S,-
$_        ETHERMON_CMD_TZ,-
$_        ETHERMON_PARSE,-
$_        ETHERMON_FASTCALL,-
$_        ETHERMON_MSG,-
$_        SYS$INPUT:/OPTIONS
COLLECT=ADDRESS_NAMES,ADDRESS_NAMES
$  MACRO ETHERMON_FASTSHR
$  LINK/SHARE/PROTECT ETHERMON_FASTSHR,-
$_                            SYS$SYSTEM:SYS.STB/SELECTIVE_SEARCH,-
$_                            SYS$INPUT:/OPTIONS
UNIVERSAL=FASTSHR_VERSION
UNIVERSAL=P_INIT_FAST_$QIO
UNIVERSAL=P_FAST_BUFIO_READ_$QIO
UNIVERSAL=P_FAST_$QIO_D_LINE
```

which produces several .OBJ files, ETHERMON.EXE, and ETHERMON_FASTSHR.EXE. Alternatively, do

```
$  @ETHERMON
```

which does the same thing (and additional compiles for other programs) with less typing.

The shareable image ETHERMON_FASTSHR.EXE must be located in the same directory as the copy of the ETHERMON.EXE file that will be run, or the shareable image's location must identified by an executive-mode system-wide logical name. The shareable image must also be installed as a shared, protected image with the INSTALL utility, using commands like the following:

```
$ ! begin optional logical name definition
$ ASSIGN/SYSTEM/EXECUTIVE_MODE -
$_    dev:[dir]ETHERMON_FASTSHR.EXE -
$_    ETHERMON_FASTSHR
$ ! end optional logical name definition
$ INSTALL :== $INSTALL/COMMAND_MODE
$ INSTALL
INSTALL> ADD dev:[dir]ETHERMON_FASTSHR.EXE/SHARE/OPEN/HEAD/PROT
INSTALL> EXIT
```

It is suggested that these commands be placed in your system startup command procedure (SYS$MANAGER:SYSTARTUP_V5.COM) if you plan on using ETHERMON regularly.

Building the help library means:

$ LIBRARY/CREATE/HELP ETHERMON ETHERMON

which produces ETHERMON.HLB. The program expects to find the .HLB file in the same directory as the .EXE files; it cannot be redirected by use of logical names. If you wish to change this behavior, the subroutine HELP in ETHERMON_CMD_AL.FOR must be edited appropriately.

ETHERMON can be invoked either with the RUN command or as a foreign command. To define ETHERMON as a foreign command, the following line of DCL must be executed before ETHERMON is used:

$ ETHER*MON :== $*disk***:[***dir***]ETHERMON**

where *disk* and *dir* specify the location of the ETHERMON*.EXE and ETHERMON.HLB files (a good place to put this line is in your LOGIN.COM file; although it can be done in SYLOGIN.COM, it is not recommended to encourage general users to try to use ETHERMON). Once that is done, ETHERMON can be invoked with the command line:

$ ETHERMON *interface*

where *interface* is the name of the Ethernet interface to use. Only the device and controller letters need be specified; the unit number will always be discarded and replaced by a zero (to ensure that the program gets a unique device unit for its work). Optionally, you can specify an initial command to ETHERMON as well by using the command form:

$ ETHERMON *interface***,***command*

where *command* is any legal ETHERMON command as defined later in this manual (including an indirect file using the @*filename* command form). This command will be executed immediately upon completing initialization of the Ethernet interface. You must specify an interface name if you also wish to specify a command. Remember also, if you are going to use a command with embedded quotes ("), that DCL preparses the command line before passing it to the application, possibly removing your quote marks before ETHERMON gets to look at the command.

When invoking ETHERMON with a RUN command, the start-up command looks like this:

$ RUN ETHERMON

If you do not specify your Ethernet port name, or if you invoke ETHERMON with a RUN command, ETHERMON will search for all recognized interface devices on your system, and will display a list of acceptable device names for your system on your terminal and query you for a device name; if only one standard Ethernet interface exists on your system, ETHERMON will not query but will automatically select that one device and inform you of that action. For example, using the RUN command on a system with two Ethernet adapters (underlined text is user-entered):

```
$ run ethermon
%ETHERMON-I-PORT, enter VMS device name of the Ethernet interface to use
_ETA0:   DEBNA      1: 60-07 DEC: LAVC
                    2: 60-03 DEC: DECnet Phase 4
                    3: 60-01 DEC: MOP Dump/Load
                    4: 90-00 Loopback
                    7: 08-00 Internet IP (DoD)
                    8: 08-06 Internet ARP
_ETB0:   DEBNA      1: 06-00 XEROX: NS IDP
Ethernet_port_: eta0
ETHERMON V2.4-1, 29-Oct-1992
ETHERMON>
```

A single Ethernet adapter system's session (using the foreign command method without specifying an interface name):

```
$ ethermon
%ETHERMON-I-ONEPORT, implicitly using the one Ethernet interface _XEA0:
ETHERMON V2.4-1, 29-Oct-1992
ETHERMON>
```

Once ETHERMON is started and has initialized the chosen interface, it will attempt to locate an initialization file with the logical name ETHERMON$INIT; if the logical name is incomplete or undefined, the initialization file's default filespec is ETHERMON$INIT.COM in the current default directory. This text file may contain any legal ETHERMON commands, although commands that expect certain data to be in the address table automatically should take into account that ETHERMON has not been collecting data very long yet. The commands in this file will execute before ETHERMON begins prompting for commands, but *after* any initial command typed on the invocation command line; if the invocation line initial command is an indirect file, the default initialization file will be skipped.

3. Operation

3.1. Theory of Operation

ETHERMON operates by using the most basic property of the Ethernet promiscuous mode: that **ALL** Ethernet messages are received by the promiscuous user. ETHERMON begins by connecting to the specified Ethernet interface and enabling the necessary privileges needed to perform the privileged I/O operations. The interface is started in promiscuous mode as a IEEE 802.3 mode receiver (Ethernet mode is a subset thereof; if Ethernet mode is used because you are running under VAX/VMS V4.x, the program uses protocol type FF-FF). It then posts an asynchronous read to the device; the read completion AST analyzes a received message and queues up another read, forever (or until the program is terminated).

As of version V2.4 of ETHERMON, certain enhancements were needed to work around a bug/"feature" in VAX/VMS V5.4-3 and above. These required directly accessing the raw Ethernet packet buffers in non-paged pool to ensure that the Ethernet MAC-level header was returned to ETHERMON correctly. This was implemented by writing a customized, streamlined version of the $QIO system service that is optimized for issuing asynchronous reads to an already assigned and protection-checked Ethernet port channel. This special version also adds an "intercept" routine to catch the completed read requests as they are sent to VMS I/O postprocessing; the intercept routine corrects the system buffer pointers to return the MAC-level headers in the format tthey were returned in earlier versions of VAX/VMS. This allows for more accurate interpretation of the MAC headers, but requires that:

- the user must hold the CMKRNL privilege, in addition to the other privileges required to use ETHERMON.
- the shareable image ETHERMON_FASTSHR.EXE must be located in the same directory as the ETHERMON.EXE file (or its location identified by an executive-mode logical name), and must be installed as a shared, protected image with the INSTALL utility (see section 2.2 on page 7 for details).

All the fancy features of ETHERMON are based on how the received messages are analyzed. Normally, received messages are analyzed to see if the sending nodes have been listed in the table of known addresses kept by ETHERMON (this is only done for messages with protocol types where ETHERMON knows how to interpret the message body). If a sending node (and, in some cases, the destination node) are not found in the addresses table, the addresses are added to the table with appropriate identification; i.e., for Internet IP messages, the Internet logical IP address is stored in the table with the Ethernet address. An Ethernet address may have more than one equivalence name in the address table (since a node may be used for more than one purpose on the Ethernet). This table of equivalence names is used by the DISPLAY, SHOW ADDRESS_NAMES, and SUMMARIZE commands to supply more meaningful identifications with the Ethernet addresses reported to the user.

The combinations of addresses and protocol types that can be recognized and decoded by ETHERMON are as follows:

1. any single-node (not multicast) address under protocol type 60-03 (DECnet Phase IV) (symbolic names obtained from the local system's NETACP process)
2. any single-node address transmitting DEC MOP System ID packets (type 7) under the DEC MOP Remote Console protocol (60-02, either Ethernet or IEEE 802.3 SNAP protocol subtype)
3. any single-node address under protocol type 08-00 (DoD Internet IP) (assuming the IP

message header is valid)

4. any source address under protocol type 08-06 (Internet ARP) (assuming the ARP header is correct; in ARP reply cases, the destination address can be deciphered as well) (proxy ARP is not processed because it is not trusted)
5. source and multicast destination addresses for messages under protocol type 60-07 (DEC Local Area VAXcluster) that are sending to LAVC Node ID multicast addresses (AB-00-04-01-*xx*-*yy*), where *yyxx* is the hexadecimal cluster group ID plus 256 (decimal)
6. any address under protocol type 80-35 (Internet reverse ARP) (RARP response packet types only)
7. any single-node source or destination address under protocol type 08-88 (Xyplex terminal network)
8. addresses assigned to Ungermann-Bass, Inc., under protocol type 06-00 (Xerox NS IDP) (prefixes 00-DD-00, 00-DD-01, and 00-DD-0F only), including NIU's and XNS-based Access/One equipment
9. all addresses assigned to DEC hardware under protocol type 80-38 (DEC Bridge management)
10. all addresses assigned to DEC hardware under protocol type 60-04 (DEC LAT)
11. certain source and destination addresses under protocol type 80-9B (AppleTalk, either Ethernet or IEEE 802.3 SNAP protocol subtype)
12. all source addresses under protocol type 80-F3 (AppleTalk ARP, either Ethernet or IEEE 802.3 SNAP protocol subtype)
13. all source addresses under protocol type 80-80 (Vitalink TransLAN III) sending to destination addresses 09-00-02-04-00-01 or 09-00-7C-04-00-01 (3Com/Bridge or Vitalink bridge logging multicast)

When a DISPLAY or SUMMARIZE command is in progress, messages are counted up in categories corresponding to the search scheme selected (by source address, destination address, message body size, or protocol type). The resulting summations are reported as the output data from these commands. In a TRACE command, the messages are passed directly to the specified output file.

3.1.1. Filtering

For more selective viewing of data, **filters** may be used to choose a subset of the data on the network for analysis. In ETHERMON, packets may be accepted or rejected by source address, destination address, an address in either source or destination field, protocol type, packet size, or a combination thereof. The filtering applies to data processed by the TRACE command, the SUMMARIZE commands, and the DISPLAY commands (when the /NOFILTER qualifier is not used).

The internal structure of an ETHERMON filter is two Ethernet MAC addresses and a protocol type. The source address is used either for an actual source address or an address that will checked for in both the source and destination MAC header address fields. The destination address is used only for destination mapping; therefore, you could have a single filter that accepts packets to or from a particular node and to a particular destination, but not one that matched a to-or-from address and a source address.

Protocol type matching is done independently of whether or not a packet is plain Ethernet or IEEE 802.3 SNAP extended type; only the Ethernet protocol type field or the last two bytes of the SNAP header are used for matching.

An Ethernet address may be specified as only the 3-byte manufacturer prefix (which will also ignore the least significant bit of the first byte; you must specify that bit as zero for a manufacturer prefix filter to work). This way, you may filter on an entire manufacturer’s class of systems.

There is a single packet size filter (which may not be saved with the SAVE/FILTERS command); it specifies a range of packet data body sizes (not counting the MAC header) which will be accepted for further processing.

Filters may be enabled and disabled independently of each other. Newly created filters (with ADD FILTER) default to being enabled; filters loaded with the LOAD/FILTER command are initially disabled. The SET FILTER command may be used to enable or disable a filter.

A filter may be set up to exclude a matching message from processing, or to only process the message if it matches. If multiple filters are enabled, a packet is accepted only if it matches at least one ONLY address and ONLY protocol (if such are enabled) and does not match any of the enabled EXCLUDE filters. For example, if you wanted to accept packets only from any of 10 workstations, you would create 10 filters, each specifying ONLY=SOURCE=*workstation-address* for one of the workstations, and enable all of these filters.

Filters may be entered using DECnet or IP addresses; however, these are always translated to corresponding physical Ethernet addresses. You cannot presently filter on the actual Internet IP address field in a packet body, or on other information in the body of a packet, with the current version of ETHERMON (although two offline analysis tools are provided to use with TRACE files to provide such additional filtering; see section 4 for more information).

3.2. Commands

Once ETHERMON is running, initially nothing will be going on (except for automatic address/name data collecting); the utility will prompt the user for command input. ETHERMON will continue to execute until the EXIT command or a control-Z is typed to terminate the program.

The defined commands are listed below; explanations of their functions and detailed syntax are given in the following sections.

```
@indirect-command-file
ADD FILTER
CLEAR FILTER
CLEAR FLAGS
DEFINE/KEY
DELETE/KEY
DISPLAY DESTINATIONS
DISPLAY PROTOCOLS
DISPLAY SIZES
DISPLAY SOURCES
DISPLAY STATISTICS
EXIT
HELP
LOAD
MAP
MONITOR
PING
QUERY
SAVE
SET ADDRESS_NAME
SET FILTER
SET INTERVAL
SET KEYPAD
SET MANUFACTURER
SET MESSAGES
SET PROTOCOL_NAME
SHOW ADDRESS_NAMES
SHOW BRIDGES
SHOW FILTERS
SHOW KEY
SHOW MANUFACTURERS
SHOW MESSAGES
SHOW PERFORMANCE
SHOW PORT_STATUS
SHOW PROTOCOL_NAMES
SPAWN
SUMMARIZE DESTINATION
SUMMARIZE INTERNET
SUMMARIZE PROTOCOL
SUMMARIZE SIZE
SUMMARIZE SOURCE
SUMMARIZE STATISTICS
TRACE
WAIT
```

3.2.1. Command Parameters

Certain command parameters have a constant format throughout ETHERMON. To simplify this user's manual, these formats are described here, rather than individually for each command.

3.2.1.1. Ethernet addresses

Ethernet addresses are 6-byte addresses. Users can specify them as command parameter and qualifier values in any of the following formats:

1. a physical Ethernet address in standard Xerox notation of the form *aa-bb-cc-dd-ee-ff*, where each letter group is a one- or two-digit hexadecimal number.
2. a physical manufacturer prefix code in standard Xerox notation of the form *aa-bb-cc*, where each letter group is a one- or two-digit hexadecimal number. You should always enter this as a even address (LSB of first number even), as the multicast bit is always cleared before doing manufacturer comparisons. Note that this is only legal for the ADD/SET FILTER command, the SHOW ADDRESS_NAMES command, and the SET MANUFACTURER command; other commands require a full node address.
3. a DECnet node number in the form *aa.bbb*, where *aa* is the decimal DECnet area number (in the range 1 to 63) and *bbb* is the decimal DECnet node number within the area (in the range 1 to 1023); the DECnet address translates directly to an Ethernet address of the form AA-00-04-00-*xx-yy*, where *yyxx* is the hexadecimal composite DECnet address ((area * 1024) + number).
4. a DECnet node name (a 1- to 6-character alphanumeric string); the node name must already be listed in the address table (so that the command can look up the equivalent Ethernet address).
5. an Internet logical IP address (four decimal numbers separated by periods, each ranging from 0 to 255, in the format *a.b.c.d*); the Internet address must already be listed in the address table (so that the command can look up the equivalent Ethernet address).
6. an Ungermann-Bass device serial number in the format UB#*nnnnn*, where *nnnnn* is the serial number of the device; ETHERMON will automatically determine which manufacturer prefix the device uses from the serial number range, and compute an address of the form 00-DD-00-*aa-bb*-00 or 00-DD-01-*aa-bb-cc*.

Some commands can accept wildcards; in these cases, wildcards may only be used in the physical hexadecimal Ethernet and Internet logical address formats, and is done by replacing a number in a field with an asterisk ('*').

3.2.1.2. Protocol types

Ethernet protocols are 2-byte code numbers. Users can specify them as command parameter and qualifier values in either of the following formats:

1. a physical Ethernet protocol number in standard Xerox notation of the form *aa-bb*, where each letter group is a one- or two-digit hexadecimal number.
2. for certain commonly used protocol types, a symbolic name may also be used. The defined symbolic names are:
   - AARP - AppleTalk Address Resolution Protocol (80-F3)
   - APPLETALK - AppleTalk Phase I or Phase II EtherTalk (80-9B)
   - ARP - DoD Internet Address Resolution Protocol (08-06)
   - DECNET - Digital Equipment Corporation's DECnet Phase IV protocol (60-03)

- DOMAIN - Apollo DOMAIN protocol (80-19)
- IP - DoD Internet Protocol (08-00)
- LAT - Digital Equipment Corporation’s Local Area Terminal service protocol (60-04)
- LAVC - Digital Equipment Corporation’s Local Area VAXcluster protocol (60-07)
- LOOPBACK - the Ethernet-specification loopback test protocol (90-00)
- MOP_DLA - Digital Equipment Corporation’s Maintenance Operations Protocol for Dump/Load Assistance (60-01)
- MOP_RC - Digital Equipment Corporation’s Maintenance Operations Protocol for Remote Console operations (60-02)
- RARP - DoD Internet Reverse Address Resolution Protocol (80-35)
- XNS - Xerox Network Services: Internet Datagram Protocol (06-00)

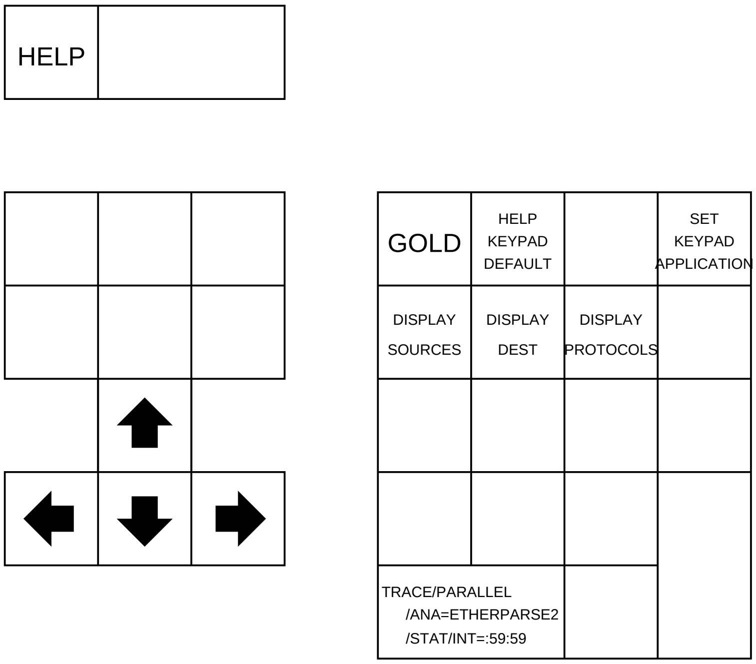
Protocol type numbers may not be wild-carded. At present, the command parser cannot tell the difference between an IEEE 802.3 SNAP extended protocol and an Ethernet protocol, so if you have a protocol used in both modes, it is specified the same way, and the ETHERMON software will treat both forms identically. This limitation will be removed in a future release of ETHERMON.

A list of protocol types known to ETHERMON can be obtained by issuing the SHOW PROTOCOL_NAMES/ALL command.

3.2.2. Command Editing and Entry

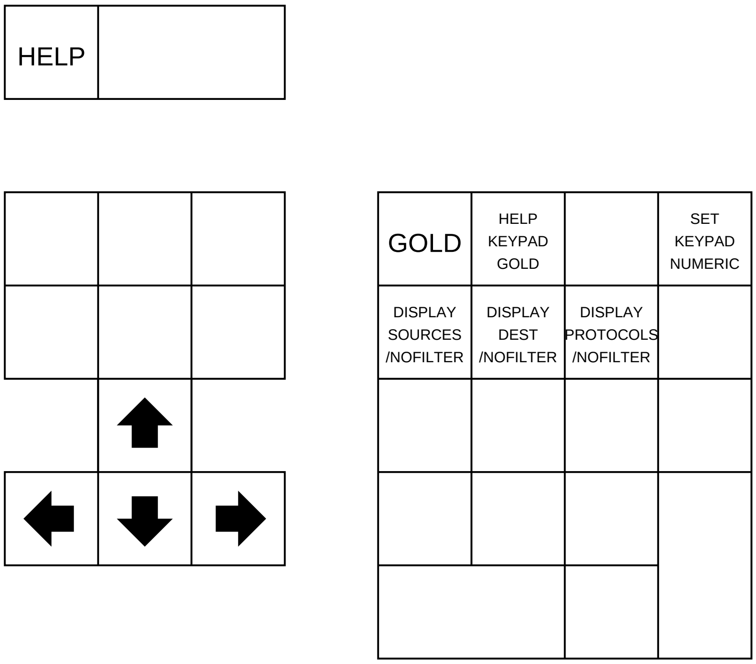
3.2.2.1. Keypad Operations

On ANSI-compatible video terminals with numeric and editing keypads, the keypad may be used to enter command lines with single keystrokes, using the command recall and line-editing features of the VMS terminal driver and Screen Management facility (SMG$). Some keypad keys are predefined and may not be changed by the user; others are undefined, and may be given definitions via the DEFINE/KEY command. The keypad is defined in two states (similar to other VMS utilities): the default state, and the GOLD state (entered by pressing the PF1 key before pressing another key). As such, there are two sets of key definitions, one by default and the other GOLDed. For the VT220/LK201 keyboard layout, the default definitions for the numeric keypad and editing keypad as as shown below:



On VT100 keyboards, use only the numeric keypad keys; the cursor (arrow) keys will still operate, but are located in a different place on the keyboard.

When the GOLD key has been used, the initial key definitions are:



Note that all of these key definitions are not user-changeable.

For convenience in defining several key definitions, the DEFINE/KEY commands may be placed in an indirect command file (just as any another command may be used).

3.2.2.2. Control Keys

There are several control keys on the keyboard which control special functions of ETHERMON. Most of them are standard VMS editing keys (DEL, control-R, control-V, etc.), but a few of them are special to ETHERMON.

Control-C is the command abort key; rather than interrupting all of ETHERMON like the control-Y key would do, it merely interrupts the current ETHERMON internal command or indirect command file in progress (if any). This way, commands that continue to execute for a very long time (or indefinitely) may be stopped prematurely. For commands that cannot be aborted, the control-C is ignored. **Note that control-C is now the only way to end a DISPLAY command (unlike earlier versions of ETHERMON, where any key would stop a DISPLAY).**

Control-Z is the program termination command; it performs the same function as the EXIT command does; however, it will not terminate ETHERMON while a DISPLAY command is running until the DISPLAY is first terminated with a control-C.

3.2.3. @indirect-command-file

The *@file* command causes a file of ETHERMON commands to be executed. Only one level of input indirection is allowed, and parameters may not be passed to the command file. It is not recommended to put an EXIT command into the indirect file, as it will cause the program to terminate (unless, of course, that is what you want to happen). The file is assumed to have a default filetype of .COM.

Command format:

@filename

The control-C key may be used to interrupt the execution of an indirect command file, but only at points where the individual commands in the file could be interrupted.

3.2.4. ADD FILTER

The ADD FILTER command adds a new filter definition to the filter table, optionally (and defaultly) enabling the filter for use in restricting data collection. No check is performed to test if this filter duplicates an existing definition. The filter can accept either matches or mismatches independently on source and/or destination address, and/or protocol type, in any combination.

During message reception for any DISPLAY, SUMMARIZE, or TRACE command, only those messages which can pass all of the enabled exclusion filters (if any) and at least one of each enabled address and protocol type inclusion filter (if any) will be processed.

Command format:

```
ADD FILTER /ENABLE (default)
           /DISABLE
           /ONLY=([PROTOCOL=protocol],[SOURCE=address],
                  [DESTINATION=address],[ADDRESS=address],
                  [MULTICAST])
           /EXCLUDE=([PROTOCOL=protocol],[SOURCE=address],
                     [DESTINATION=address],[ADDRESS=address],
                     [MULTICAST])
```

where *protocol* is a Ethernet protocol type code, expressed as described in section 3.2.1.2, and *address* is a node address, expressed as described in section 3.2.1.1.

The use of SOURCE= specifies that the *address* is only treated as a source address; DESTINATION= specifies a destination address only; ADDRESS= implies the address regardless of whether it was a source or destination; and MULTICAST specifies any multicast destination (including the Broadcast address).

3.2.5. CLEAR FILTER

The CLEAR FILTER command removes one or all filter definitions from the filter table. The definition(s) are completely erased (not merely disabled); to disable a filter without deleting it, use the SET FILTER/DISABLE command.

Command format:

CLEAR FILTER *filternumber*

CLEAR FILTER /ALL

where *filternumber* specifies the number of the filter to delete (displayed by the ADD FILTER, SET FILTER, and SHOW FILTER commands), and /ALL specifies clearing all filters from the table.

3.2.6. CLEAR FLAGS

The CLEAR FLAGS command clears the Internet (IP) flag bits for all address table entries (see the SHOW ADDRESS_NAMES command for the /TYPE qualifier). This is primarily available to allow re-deriving of IP addresses for nodes when some of the addresses are garbled (generally due to glitchy RIP-type IP routing messages detected on the Ethernet). After entering the command, automatic IP address derivation will start over from scratch (although the Ethernet address entries themselves will not be deleted).

Command format:

CLEAR FLAGS

3.2.7. DEFINE/KEY

The DEFINE/KEY command associates an equivalence string and a set of attributes with a key on the terminal keyboard. The /KEY qualifier is required.

Command format:

DEFINE/KEY [/*qualifiers***]** *key-name equivalence-string*

where *key-name* specifies the name of the key that you are defining, and *equivalence-string* specifies the string which is to be processed when you press the key. The entire remaining portion of the line after the key name is accepted as the equivalence string, as-is, so all qualifiers for the DEFINE/KEY command (as opposed to the equivalence string) must appear before the key-name.

The defined command qualifiers for DEFINE/KEY are:

- /NOECHO - determines whether the equivalence string is displayed on your screen after the key has been pressed. By default, the equivalence string is echoed to the screen when it is pressed.
- /IF_STATE=state-name - specifies a state which must be in effect for the key definition to be in effect. If you omit the /IF_STATE qualifier, the default state is used. The state name is an alphanumeric string. States are established with the /SET_STATE qualifier. The states already known to ETHERMON are DEFAULT (implied if /IF_STATE is not used) and GOLD (keypress immediately following pressing the PF1 key).
- /LOCK_STATE - specifies that the state set by the /SET_STATE qualifier remain in effect until explicitly changed. If you do not use the /LOCK_STATE qualifier, the state set by /SET_STATE is in effect only for the next defineable key that you press or for the next read terminating character that you type. The /LOCK_STATE qualifier can only be specified with the /SET_STATE qualifier.
- /SET_STATE=state-name - causes the specified state-name to be set when the key is

pressed. The state name can be any alphanumeric string. If you omit the /SET_STATE qualifier, the current state that was locked remains in effect; if there is no locked state, the DEFAULT state becomes the new state.

- /TERMINATE - specifies whether the current equivalence string is to be terminated (that is, processed) when the key is pressed. Pressing RETURN after pressing the defined key has the same effect as using /TERMINATE.

The following table lists the key names in column one. The remaining three columns indicate the key designations on the keyboards for the three different types of terminals that allow key definitions. All definable keys on VT52 terminals are located on the numeric keypad. On VT100-type terminals, you can define the LEFT and RIGHT arrow keys as well as all the keys on the numeric keypad (although re-defining the arrow keys is not recommended). There are three types of keys that can be defined on terminals with LK201 keyboards: (1) keys on the numeric keypad, (2) keys on the editing keypad (except the UP and DOWN arrow keys), and (3) keys on the function key row across the top of the terminal. (Note that you cannot define function keys F1 through F5.)

Key-name	LK201	VT100-type	VT52
PF1	PF1	PF1	[blue]
PF2	PF2	PF2	[red]
PF3	PF3	PF3	[gray]
PF4	PF4	PF4	- -
KP0,KP1,...,KP9	0,1,...,9	0,1,...,9	0,1,...,9
PERIOD	.	.	.
COMMA	,	,	n/a
MINUS	-	-	n/a
ENTER	Enter	ENTER	ENTER
Find (E1)	Find	- -	- -
Insert Here (E2)	Insert Here	- -	- -
Remove (E3)	Remove	- -	- -
Select (E4)	Select	- -	- -
Prev Screen (E5)	Prev Screen	- -	- -
Next Screen (E6)	Next Screen	- -	- -
HELP	Help	- -	- -
DO	Do	- -	- -
F6, F7, ..., F20	F6, F7, ..., F20	- -	- -

Some defineable keys are enabled for definition all the time. Others, including KP0 through KP9, PERIOD, COMMA, and MINUS, need to be enabled for definition purposes. You must issue the SET KEYPAD APPLICATION command before using these keys (by default, ETHERMON implicitly issues this command, but its effect may be cancelled with the SET KEYPAD NUMERIC command).

The LEFT and RIGHT arrow keys and the F6 through F14 VT200 keys are reserved for command line editing. You must issue the SET TERMINAL/NOLINE_EDITING command from the DCL level before defining these keys. You can also press control-V to enable keys F7 through F14. Note that control-V will not enable the F6 key.

Following is a sample of how to use defineable keys. The user wanted a method of typing a long, complicated TRACE command in as soon as a certain event occurred. A key definition was made to define the keypad ENTER key as translating to the command. When the event alert message occurred, the user merely had to press a single key to begin recording.

```
ETHERMON> DEFINE/KEY/TERM/NOECHO ENTER TRACE/BIN/PARA/INT=:5/OUT=GRABIT
ETHERMON>
     .
     .
     .
%ETHERMON-I-MOPMSG, 11:22:33 08-00-2B-12-34-56 sent MOP DLA msg, t=0508 ln=8
ETHERMON> <press keypad ENTER key>
%ETHERMON-I-ASYNTRCBEG, asynchronous TRACE command started...
```

3.2.8. DELETE/KEY

The DELETE/KEY command deletes key definitions that have been established by the DEFINE/KEY command. The /KEY qualifier is required.

Command Format:

DELETE/KEY [/*qualifier***]** *key-name*

where *key-name* specifies the name of the key whose definition you want to delete, and the optional qualifier is /STATE=state-name, which specifies the name of the state for which the specified key definition is to be deleted. State names can be any alphanumeric string already defined by the DEFINE/KEY command. If you omit the /STATE qualifier, key definitions in the DEFAULT state are deleted.

3.2.9. DISPLAY DESTINATIONS

The DISPLAY DESTINATIONS command brings up a continuously updating screen (similar to the VMS MONITOR utility) reporting on the current top *N* destination addresses that Ethernet messages are being sent to. The screen is updated at the rate specified by the SET INTERVAL command, and is terminated by the user typing a control-C.

Command format:

DISPLAY DESTINATIONS [/NOFILTER]

The upper part of the screen reports the display type and the system name and interface port used for collecting the data. Below that is the report of messages lost (generally because the program could not read and process them as fast as they were coming in), and the number of messages and bytes of data received before and after filtering. The remainder of the screen is a bar graph, displaying the number of post-filter messages received for the top *N* destination addresses, where the addresses are displayed as physical Ethernet addresses and the first few characters of their combined equivalence names (if any, as many as will fit for the available screen size). After the physical address is the message count in messages per second (averaged over the sampling interval) and the bar graph line for that value.

The /NOFILTER qualifier specifies that the DISPLAY will not use the currently active filters to restrict displayed data; this is useful when you have a background TRACE command running and still want to use the displays.

3.2.10. DISPLAY PROTOCOLS

The DISPLAY PROTOCOLS command brings up a continuously updating screen (similar to the VMS MONITOR utility) reporting on the current top *N* protocol types that Ethernet messages are being sent under. The screen is updated at the rate specified by the SET INTERVAL command, and is terminated by the user typing a control-C.

Command format:

DISPLAY PROTOCOLS [/NOFILTER]

The upper part of the screen reports the display type and the system name and interface port used for collecting the data. Below that is the report of messages lost (generally because the program could not read and process them as fast as they were coming in), and the number of messages and bytes of data received before and after filtering. The remainder of the screen is a bar graph, displaying the number of post-filter messages received for the top *N* protocol types, where the protocol types are displayed as physical Ethernet type codes and the first few characters of their equivalence name (if any, as many as will fit for the available screen size); if an 'i' appears after the physical protocol type number, this is an IEEE 802.3 extended (SNAP) protocol rather than an Ethernet protocol. After the physical type code is the message count in messages per second (averaged over the sampling interval) and the bar graph line for that value.

The /NOFILTER qualifier specifies that the DISPLAY will not use the currently active filters to restrict displayed data; this is useful when you have a background TRACE command running and still want to use the displays.

3.2.11. DISPLAY SIZES

The DISPLAY SIZES command brings up a continuously updating screen (similar to the VMS MONITOR utility) reporting on the current number of Ethernet messages in each of 15 size categories. The screen is updated at the rate specified by the SET INTERVAL command, and is terminated by the user typing a control-C.

Command format:

DISPLAY SIZES [/NOFILTER]

The upper part of the screen reports the display type and the system name and interface port used for collecting the data. Below that is the report of messages lost (generally because the program could not read and process them as fast as they were coming in), and the number of messages and bytes of data received before and after filtering. The remainder of the screen is a bar graph, displaying the number of post-filter messages received for the 15 size categories, ranging from 1 to 100 bytes, to 1401 to 1500 bytes, in size partitions of 100 bytes. After the size range on the display is the message count in messages per second (averaged over the sampling interval) and the bar graph line for that range.

The /NOFILTER qualifier specifies that the DISPLAY will not use the currently active filters to restrict displayed data; this is useful when you have a background TRACE command running and still want to use the displays.

3.2.12. DISPLAY SOURCES

The DISPLAY SOURCES command brings up a continuously updating screen (similar to the VMS MONITOR utility) reporting on the current top *N* source addresses that Ethernet messages are being sent to. The screen is updated at the rate specified by the SET INTERVAL command, and is terminated by the user typing a control-C.

Command format:

`DISPLAY SOURCES [/NOFILTER]`

The upper part of the screen reports the display type and the system name and interface port used for collecting the data. Below that is the report of messages lost (generally because the program could not read and process them as fast as they were coming in), and the number of messages and bytes of data received before and after filtering. The remainder of the screen is a bar graph, displaying the number of post-filter messages received for the top *N* source addresses, where the addresses are displayed as physical Ethernet addresses and the first few characters of their combined equivalence names (if any, as many as will fit for the available screen size). After the physical address is the message count in messages per second (averaged over the sampling interval) and the bar graph line for that value.

The /NOFILTER qualifier specifies that the DISPLAY will not use the currently active filters to restrict displayed data; this is useful when you have a background TRACE command running and still want to use the displays.

3.2.13. DISPLAY STATISTICS

The DISPLAY STATISTICS command brings up a continuously updating screen (similar to the VMS MONITOR utility's unsupported ETHERNET class display) reporting on the current Ethernet port statistics (or counters, to use DECnet NCP terminology). The screen is updated at the rate specified by the SET INTERVAL command, and is terminated by the user typing a control-C.

Command format:

`DISPLAY STATISTICS`

The upper part of the screen reports the display type and the system name and interface port used for collecting the data. Below that is the report of messages lost (generally because the program could not read and process them as fast as they were coming in), and the number of messages and bytes of data received before and after filtering. The remainder of the screen is a display of current, average, minimum, and maximum counts (averaged over the sampling interval) for data bytes received, data messages received, multicast bytes received, multicast messages received, single and multiple collisions, transmission delays due to potential collision, and assorted buffer errors.

There are two points to keep in mind regarding this command:

1. The data displayed by this command is from the raw counters in the Ethernet interface itself; no filtering can be applied to this data, so the /NOFILTER qualifier is meaningless (the results will be the same whether or not filtering is applied).
2. Because some of the data counters saturate or "wrap around" if the interface has been

running for a while since it was last zeroed, some of the "current data" fields in this display may appear as lines of hyphens (flagging that those particular counters have saturated). To correct this, use the SHOW PORT_STATUS/ZERO command to zero the counters; this should correct any such display errors. Conversely, if another user issues the DECnet NCP command ZERO LINE COUNTERS, it may cause the display to briefly show erroneous currnet data (which will probably corrupt the average, minimum, and maximum data). This can be corrected by re-issuing the DISPLAY STATISTICS command, so that it starts with a fresh set of data.

3.2.14. EXIT

The EXIT command terminates execution of the ETHERMON utility. This function is also performed by typing control-Z. The special promiscuous Ethernet port is shut down and deleted, any active TRACE/PARALLEL or TRACE/REPEAT command is terminated as if a TRACE/ABORT command was typed, and all information collected by ETHERMON (except that previously stored with the SAVE command) is discarded.

Command format:

EXIT

3.2.15. HELP

The HELP command obtains helpful information on the formats and usages of the ETHERMON commands. It functions similarly to standard VMS HELP.

Command format:

HELP [topic ...]

3.2.16. LOAD

The LOAD command reads a saved file of ETHERMON table information into the run-time database of the program. Four tables of dynamically updated name equivalences are maintained by the utility: the address table (storing equivalences to Ethernet physical addresses), the filter table (storing message filtering definitions), the manufacturer prefix table (storing three-byte Ethernet address prefixes), and the protocol type table (storing equivalence names for Ethernet protocol types). LOAD will read a file and merge the file contents into one of these tables (append to table end for the filters table). The input file format is rather inflexible and defined to match the files generated by the SAVE command.

Command format:

LOAD /ADDRESSES **[***inputfile***]**
/FILTERS **[***inputfile***]**
/INTERNET
/MANUFACTURERS **[***inputfile***]**
/PROTOCOLS **[***inputfile***]**

where /ADDRESSES if the default if no qualifier is specified, and *inputfile* is the optional specification of the file to read; if *inputfile* is missing or incomplete, the file specification is filled in from the default of

SYS$DISK:[]ETHERMON.ETH_*qualifier*, where *qualifier* is the name of the qualifier currently in effect.

Upon LOADing an address, manufacturer prefix, or protocol type file, the file entries are merged with any existing entries in the table; duplicate fields are overwritten, but fields not specified in the file for a particular Ethernet address, prefix, or protocol are not changed. Note that multiple protocol entries for with and without IEEE 802 mode will not be properly handled by the current version.

Upon LOADing a filter file, all file entries are added as new entries; no match-ups between existing filter table entries and the new ones are made. Also, since the filter table is rather small (only 25 entries), it is very possible for the file read to prematurely terminate with a "table full" error; this will not abort ETHERMON, but will merely stop reading any more entries from the file. Also note that all filters LOADed from a file are initially disabled and, to be used, must be turned on with the SET FILTER/ENABLE command.

If /INTERNET is specified, ETHERMON attempts to connect to the local Internet software running on the local system; currently, only TGV MultiNet, DEC VMS/Ultrix Connection (UCX, now called TCP/IP Services for VMS), Carnegie-Mellon University TCP/IP (CMU TCP/IP), and Network Research Corporation's Fusion (NRC Fusion) are supported. If the connection succeeds, ETHERMON will query the Internet software for any entries that match detected Internet addresses seen on the cable which have not received "real" name strings (as opposed to ASCII representations of the physical address). Any match-ups have their Internet name strings replaced with the name supplied by the Internet software. Note that the results of this command will be no better than the local Internet software can supply, so if your local node doesn't know much and does not have access to Domain Name Service, don't expect much from this command.

3.2.17. MAP

The MAP command attempts to query each Ethernet bridge currently known to the program for its packet forwarding tables, in order to attempt to build a interconnection "map" of the Ethernet segments visible from the current monitoring point.

Command format:

```
MAP
```

Note that the MAP command only works for bridges speaking DEC LAN Bridge protocol (80-38) or Ungermann-Bass's spanning tree protocol run over XNS, and is not guaranteed to work for those (depending on filtering, passwords, etc.); however, references to any recognized bridge (whether or not it speaks a bridge protocol that the MAP command knows how to use) will be listed in the overall bridge routing map constructed by ETHERMON. To see the computed map, issue the SHOW BRIDGES/FULL command; to plot the map as a PostScript graphic image, issue the SHOW BRIDGES/MAP command.

Note that the map is not guaranteed to be totally accurate; since bridges may not respond to queries, or the responses may get dropped due to overloading of the monitoring VAX.

3.2.18. MONITOR

The MONITOR command is a synonym for the DISPLAY command, for ease of use for heavy VMS MONITOR users. Any of the DISPLAY keywords and qualifiers may be used with MONITOR.

Command format:

MONITOR DESTINATIONS

MONITOR PROTOCOLS

MONITOR SIZES

MONITOR SOURCES

MONITOR STATISTICS

3.2.19. PING

The PING command attempts to "ping" a message off a remote node on the Ethernet, similar to the Unix "ping" utility, and reports the turnaround time if successful; however, the ETHERMON PING command can "ping" using multiple protocols.

Command format:

PING [/IEEE_LOOPBACK] [*address***]**
PING /IEEE_MOP [*address***]**
PING /INTERNET [*address***]**
PING /LOOPBACK [*address***]**
PING /MOP [*address***]**

where the address of the remote node is expressed as described in section 3.2.1.1; note that a full, non-wildcarded address must be used. If no address is specified, the Ethernet Broadcast address (FF-FF-FF-FF-FF-FF) will be used. This is useful when you wish to test for the existence of a certain group of nodes, or just to help fill the address names table (by causing relatively idle nodes to generate traffic that ETHERMON can detect).

The qualifiers specify which format of "ping" message to use. The default is /IEEE_LOOPBACK if no qualifier is given. The message formats available are:

1. /IEEE_LOOPBACK: using IEEE 802.3 SNAP protocols for the general loopback protocol type (90-00).
2. /IEEE_MOP: using IEEE 802.3 SNAP protocols using the DEC MOP Remote Console protocol (60-02) type's Request ID and System ID response message types.
3. /INTERNET: using DoD Internet IP (08-00) datagrams using the ICMP Echo Request and Echo Reply message formats. Note that the target node must have an IP address stored for it in the address table, and, most likely, replies may not come back if the Ethernet port you are monitoring on is not already speaking TCP/IP.
4. /LOOPBACK: using Ethernet protocols for the general loopback protocol type (90-00).
5. /MOP: using Ethernet protocols using the DEC MOP Remote Console protocol (60-02) type's Request ID and System ID response message types.

Note that, on a busy Ethernet, the PING command may fail because the reply packet was discarded by the Ethernet I/O driver when ETHERMON couldn't keep up with the incoming data flow. This does not necessatily mean that the target node does not exist.

3.2.20. QUERY

The QUERY command issues various types of queries to ranges of Ethernet addresses; it assumes that the automatic address recognition logic in ETHERMON will do something useful with the reply messages (if any). This allows searching for various types of remote devices by forcing them to respond to a direct status poll.

Command format:

QUERY/*querytype* **[/***qualifier***]** *address* **[***endaddress***]**

where *address* is the first address to query, expressed as described in section 3.2.1.1; *endaddress* is an optional last address to query (all Ethernet addresses between the first address and the last address inclusive will be queried); and *querytype* is the type of query to issue.

Legal query types include:

- /DECBRIDGE - issues a DEC "dump bridge statistics" query using DEC Lan Bridge protocol (80-38).
- /DECMOP - issues a DEC MOP "Request ID" query using DEC Maintenance Operations Protocols' Remote Console protocol (60-02).
- /UB_BRIDGE - issues an Ungermann-Bass 'dump bridge forwarding table B' query using the XNS protocol (as would be done by an Ungermann-Bass Network Management Console).

Additional *qualifiers* that may be specified are /LITTLE2_ENDIAN and /LITTLE3_ENDIAN, which control the sequence in which the span of addresses is traversed. By default, Ethernet addresses are assumed to be big-end first, so one will be added to the last byte, with carries trickling to the left as each byte overflows. However, some address sequences are different (DECnet addresses are little-endian over the last two bytes, for example); to traverse these in a rational order, either of the above two qualifiers may be used to indicate how many of the last bytes of the address should be treated as little-endian order; bytes to the left of the little-endian group are treated as big-endian order and of larger magnitude than the little-endian block.

Note that this command can cause large amounts of network traffic, which may be particularly damaging to your network if you use remote bridges with sub-Ethernet line speeds between the Ethernet segments. ETHERMON's QUERY command has been measured to spew out approximately 200 messages/second on a VAX 6000-440, each of which has a different destination address, which will massively strain bridges trying to locate all the addresses that do not actually exist. Bear this in mind before using this command. To prevent accidental overloading of the Ethernet, a sanity check is applied to limit the number of addresses that will be queried; if more than 1000 addresses could be queried, the user is asked to confirm the address range before proceeding. If more than 16777215 addresses are in the range, the command is unconditionally rejected.

3.2.21. SAVE

The SAVE command writes the current contents of the ETHERMON utility's address, filter, manufacturer prefix, and protocol type tables to disk files, where they may be archived for later use with the LOAD command. These tables are built from a combination of pre-defined "hardwired" entries, entries manually entered by the user, and entries automatically made by the automatic address recognition and interpretation logic. However, their contents are lost when ETHERMON is exited.

Command format:

```
SAVE /ADDRESSES      [outputfile]
     /FILTERS        [outputfile]
     /MANUFACTURERS  [outputfile]
     /PROTOCOLS      [outputfile]
```

where /ADDRESSES if the default if no qualifier is specified, and *outputfile* is the optional specification of the file to write; if *outputfile* is missing or incomplete, the file specification is filled in from the default of SYS$DISK:[]ETHERMON.ETH_*qualifier*, where *qualifier* is the name of the qualifier currently in effect.

For the /ADDRESSES qualifier, the output file is an ASCII file where one record corresponds to one entry in the address table. Each line begins with the Ethernet address expressed as a single 48-bit (12-digit) hexadecimal number, last byte (not first) first. After the address, delimited by spaces are the equivalences for the other fields, if those fields are not empty (it is possible to have a record with only an Ethernet address and no equivalences whatsoever, if that node sent or received messages only on protocols that ETHERMON cannot decipher). The second Ethernet address field is of the form

```
A:ffeeddccbbaa
```

where *ffeeddccbbaa* is the alternate Ethernet address stored in the same form as the primary Ethernet address. The DECnet field is of the form

```
D:nn.mmm
```

or

```
D:name
```

where *name* is the DECnet node name (as recognized by your local node's DECnet NETACP volatile database), or *nn.mmm* is the area and node number representation if the node name was not known. The preceding 'D:' is required and must be in upper case. The Internet field is of the form

```
I:a.b.c.d'domain-name'
```

where *a.b.c.d* is the decimal Internet IP physical address for the node and *domain-name* is the symbolic equivalence name string (surrounded by single quotes). Currently, ETHERMON has no way to translate an IP address to a domain-name (other than the LOAD/INTERNET command), so the *domain-name* field is usually a duplicate of the IP address; the user may change this by editing the SAVE file manually and LOADing the modified file back into ETHERMON. The serial number field is of the form

```
S:nnnn
```

where *nnnn* is the variable-length decimal serial number in the manufacturer's numbering scheme. The XNS network number field is of the form

```
X:ddccbbaa
```

where *ddccbbaa* is a 4-byte hexadecimal number (last byte first) of the XNS network number field. The miscellaneous field is of the form

```
Mxxxx:text optionally containing blanks
```

where *xxxx* is a hexadecimal mask indicating what types of names have been stored in the miscellaneous name string, and the *text* can be of any length (not more than 80 characters, currently) and may contain embedded blanks or any other printable characters; because of this, the miscellaneous field must be the last field on the line, and all of the line after the 'M*xxxx*:' is assumed to be part of the miscellaneous name. Other fields stored in the address table are currently not stored in the SAVE file.

For the /FILTERS qualifier, the output file is an ASCII file where one record corresponds to one entry in the filter table. Each line may contain one or more definitions, delimited by spaces, which, combined, make up that filter. The definitions are of the form

```
<mode>S:address
<mode>D:address
<mode>A:address
<mode>M
<mode>P:typecode
```

where <*mode*> is one of the letters **O** (for Only) or **E** (for Exclude), **S**, **D**, **A**, and **P** indicate source address, destination address, any address, or protocol type, **M** indicates any mulitcast destination address, *address* is either a 48-bit, 12-digit, high-byte-first physical Ethernet address (as used in the address SAVE file) or a 24-bit, 6-digit, high-byte first physical manufacturer prefix code, and *typecode* is a 16-bit, 4-digit, high-byte-first physical Ethernet protocol type code (as used in the protocol SAVE file).

For the /MANUFACTURERS qualifier, the output file is an ASCII file where one record corresponds to one entry in the manufacturer prefix table. Each line starts with a 24-bit, 6-digit hexadecimal number (high byte first) which is the manufacturer prefix code, followed by a space, an 8-character blank-padded field for the short version of the prefix name, a space, and arbitrary text (to the end of line) for the long form of the manufacturer name.

For the /PROTOCOLS qualifier, the output file is an ASCII file where one record corresponds to one entry in the protocol type table. Each line starts with a 16-bit, 4-digit hexadecimal number (high byte first) which is the protocol type code, followed by a space and arbitrary text (to the end of line) which is the equivalence name. Note that the IEEE flag (indicating that the protocol is an IEEE 802.3 SNAP subtype or not) is not stored in a SAVE file. This limitation will be removed in a future release.

3.2.22. SET ADDRESS_NAME

The SET ADDRESS_NAME command manually adds a new entry to the address name table with an equivalence name in the miscellaneous field, or overwrites the miscellaneous field of an existing entry. Hence, Ethernet addresses that cannot be recognized by the automatic interpretation logic can be manually named.

Command format:

`SET ADDRESS_NAME aa-bb-cc-dd-ee-ff name`

`SET ADDRESS_NAME aa-bb-cc-dd-ee-ff "quoted string"`

where *aa-bb-cc-dd-ee-ff* is the standard format physical Ethernet address specification (low byte first, hexadecimal numbers); symbolic address formats cannot be used here. *name* is a text string without embedded blanks or punctuation marks (other than _ or $), and *"quoted string"* is a string enclosed in double quotes (not considered part of the string) which may contain any printable character other than a double quote. Only the first 80 characters of the supplied equivalence name will be stored in the table; if a matching entry already exists in the address table and its miscellaneous field is not empty, ETHERMON will prompt to verify that you wish to overwrite the existing field contents. Note that some pieces of address information automatically computed by ETHERMON are stored in the miscellaneous field; these will be destroyed by using this command, but will be regenerated by ETHERMON when it sees new packets containing the information that derived the name data the previous time.

3.2.23. SET FILTER

The SET FILTER command modifies the configuration of a pre-defined filter, enables or disables a filter, or modifies the message size filter settings. Filters can accept either matches or mismatches independently on source and/or destination address, and/or protocol type, in any combination.

During message reception for any DISPLAY, SUMMARIZE, or TRACE command, only those messages which can pass all of the enabled exclusion filters (if any) and at least one of each enabled address and protocol type inclusion filter (if any) will be processed.

Command format:

```
SET FILTER filternum /ENABLE
                     /DISABLE
                     /ONLY=([PROTOCOL=protocol],[SOURCE=address],
                            [DESTINATION=address],[ADDRESS=address],
                            [MULTICAST])
                     /EXCLUDE=([PROTOCOL=protocol],[SOURCE=address],
                               [DESTINATION=address],
                               [ADDRESS=address],[MULTICAST])
                     /NODESTINATION
                     /NOPROTOCOL
                     /NOSOURCE

SET FILTER /SIZE=(ENABLE,
                  DISABLE,
                  MAXIMUM=nnn,
                  MINIMUM=nnn)
```

For the first form of the command, *filternum* is the index number of a pre-existing filter definition (as reported by ADD FILTER and SHOW FILTER), *protocol* is a protocol type code as described in section 3.2.1.2, and *address* is a node address as described in section 3.2.1.1. Wildcards are permitted in the address, as are manufacturer-prefix-only values. The use of SOURCE= specifies that the *address* is only treated as a source address; DESTINATION= specifies a destination address only; ADDRESS= implies the address regardless of whether it was a source or destination; and MULTICAST specifies any multicast destination (including the Broadcast address).

For the second form of the command, only one of ENABLE or DISABLE may be specified for the /SIZE qualifier, and these may be specified with or without the MAXIMUM and MINIMUM options; these options take a decimal byte count *nnn* as the minimum and maximum message sizes that may pass through the filter. There is no safety check on minimum and maximum sizes to ensure that the maximum is greater than the minimum, so the size filter could be set to exclude all messages; be careful of this.

3.2.24. SET INTERVAL

The SET INTERVAL command sets the number of seconds between screen updates for the DISPLAY commands, and the default period of data collection for the SUMMARIZE and TRACE commands.

Command format:

```
SET INTERVAL nn
```

where *nn* is the decimal integer number of seconds to be used, in the range of 2 to 59 seconds. The initial default is 4 seconds.

3.2.25. SET KEYPAD

The SET KEYPAD command switches the state of the terminal's numeric keypad to numeric or applications mode. By default, ETHERMON will switch the keypad to applications mode upon start-up, and restore the terminal to whatever state it was in before ETHERMON was invoked when ETHERMON terminates.

Command format:

```
SET KEYPAD APPLICATIONS
SET KEYPAD NUMERIC
```

3.2.26. SET MANUFACTURER

The SET MANUFACTURER command manually adds a new entry to the manufacturer prefix name table with an equivalence name, or overwrites the equivalence name of an existing entry. Hence, Ethernet interface manufacturers not already known to ETHERMON can be manually named.

Command format:

```
SET MANUFACTURER aa-bb-cc short-name long-name

SET MANUFACTURER aa-bb-cc short-name "quoted long name"
```

where *aa-bb-cc* is the first three bytes of the standard format physical Ethernet address specification (low byte first, hexadecimal numbers), *short-name* is a text string without embedded blanks or punctuation marks (other than _ or $) which is used to prefix node name information on the DISPLAY screens, and *long-name* and *"quoted long name"* are strings (optionally enclosed in double quotes, may contain any printable character other than a double quote), which is a more descriptive name for the manufacturer. Only the first 8 characters of the supplied short-name and first 24 characters of the long-name will be stored in the table; if a matching entry already exists in the manufacturer table, ETHERMON will prompt to verify that you wish to overwrite the existing names.

3.2.27. SET MESSAGES

The SET MESSAGES command enables or disables display of asynchronous "significant event" messages on the user terminal. By default, asynchronous event messages can interrupt any output on your screen any time while ETHERMON is running; however, when executing a DISPLAY command, such messages are displayed in a separate scrolling window, and the messages are not written to any SHOW or SUMMARIZE command's output when the /OUTPUT qualifier is used.

Command format:

SET MESSAGES [*msgtype***] ON**

SET MESSAGES [*msgtype***] OFF**

SET MESSAGES DEBUG

where ON specifies that significant messages should be displayed (the default), OFF specifies that messages should not be displayed, and DEBUG not only enables display, but turns on additional event messages not normally considered significant. Note that the default changes to DEBUG if you compile ETHERMON_CMD_S.FOR with the /D_LINES compiler option.

The optional *msgtype* of the SET MESSAGE command allow enabling and disabling optional sanity checks and other possibly annoying messages in a selective fashion. These message types are:

- ARP - if ARP checking is enabled, ETHERMON will report any proxy ARP messages, any ARP or RARP messages using unrecognized ARP/RARP opcodes, and any ARP or RARP packets requesting addresses for protocols other than DoD Internet IP, Berkeley trailer negotiation, AppleTalk, or Apollo DOMAIN. (ON by default)
- DUPIP - if DUPIP checking is enabled, ETHERMON will check for any new node addresses that claim the same IP address as a node already existing in the address table. (OFF by default)
- IP - if IP checking is enabled, ETHERMON will report any DoD Internet IP-speaking nodes that are transmitting more than one source address, and any nodes claiming they are IP routers. (ON by default)
- MOP - if MOP checking is enabled, any time a DEC MOP protocol message is received (other than a System ID message), ETHERMON will report the MOP protocol used, the message length, and the message type byte. (ON by default)
- NEWNODE - if NEWNODE checking is enabled, ETHERMON will report each time a new entry is added to the address table (entries loaded by the LOAD/ADDRESSES command but not previously seen by the current ETHERMON session will not be reported, as they are assumed to already be known). (OFF by default)

3.2.28. SET PROTOCOL_NAME

The SET PROTOCOL_NAME command manually adds a new entry to the protocol name table with an equivalence name, or overwrites the equivalence name of an existing entry. Hence, Ethernet protocols not already known to ETHERMON can be manually named.

Command format:

```
SET PROTOCOL_NAME aa-bb name

SET PROTOCOL_NAME aa-bb "quoted string"
```

where *aa-bb* is the standard format physical Ethernet protocol type specification (low byte first, hexadecimal numbers), *name* is a text string without embedded blanks or punctuation marks (other than _ or $), and *"quoted string"* is a string enclosed in double quotes (not considered part of the string) which may contain any printable character other than a double quote. Only the first 24 characters of the supplied equivalence name will be stored in the table; if a matching entry already exists in the protocol table, ETHERMON will prompt to verify that you wish to overwrite the existing equivalence name.

WARNING !!!! At present, the SET PROTOCOL_NAME command cannot tell the difference between an IEEE 802.3 extended protocol and an Ethernet protocol in the table, so if you have a protocol used in both modes, the SET PROTOCOL_NAME command will affect the first occurrence located in the protocol name table (not necessarily the displayed order, as the table is sorted by the SHOW PROTOCOL_NAMES command for display).

3.2.29. SHOW ADDRESS_NAMES

The SHOW ADDRESS_NAMES command displays the current contents of the address table on the user terminal, preceded and followed by the current table size. This includes all pre-defined "hardwired" names, all names extracted from the local node or VAXcluster, all names manually entered with the SET ADDRESS_NAME command, and all names automatically derived by the address recognition logic and the LOAD/INTERNET command. Any address which has been seen on your Ethernet since ETHERMON was started will have a letter displayed before the hexadecimal address string; the letter will be a "s" (for source only), "d" (for destination only), "m" (for multicast destination only), or "b" (used both as source and destination).

Command format:

```
SHOW ADDRESS_NAMES [wildcard-address]
                   [/ALL]
                   [/FULL]
                   [/OUTPUT=filespec]
                   [/PROTOCOL=protocol]
                   [/TYPE=%Xnnnn]
                   [/TYPE=(typename[,typename...])]
```

Note that the name count at the end of a display may be larger than at the beginning, due to automatic name insertion while the command is in progress. Also note that, unless the /ALL qualifier is specified, only addresses that have actually been seen on the Ethernet since the current ETHERMON session was started will actually be displayed; alternate addresses and addresses predefined by ETHERMON (but not active) will not be displayed.

The *wildcard-address* parameter allows you to restrict address display to a subset of known addresses, where the parameter is expressed in one of the formats described in section 3.2.1.1. The address can be specified as just a manufacturer prefix, where the missing terms are assumed to be wildcarded. If no address is given, the address *-*-*-*-*-* is assumed. The Internet logical address form *a.b.c.d*, where each number is a legal (0 to 255) decimal byte value or the '*' wildcard symbol, may also be used, providing at least one of the octets is not a wildcard. Specifying both an Ethernet address and an Internet address will cause only addresses meeting *both* criteria to be displayed (if any meet the criteria).

The /ALL qualifier specifies that all addresses, not just active ones, will be displayed (subject to other restrictions).

The /FULL qualifier specifies that the protocol types used by each address are displayed after that address when the table is displayed. Currently, ETHERMON can save up to 10 protocol types (Ethernet or IEEE 802.3 SNAP) per Ethernet address. The protocol type names (if any) from the protocol names table will be displayed after the physical numbers; preceding each protocol type number will be letters indicating direction of use ('s', 'd', or 'b') and IEEE mode ('i' for IEEE 802.3 SNAP, ' ' for Ethernet).

The /OUTPUT=filespec qualifier allows you to redirect the SHOW command's display to a file instead of the user terminal. If the file specification is missing or incomplete, the name SYS$DISK:[]ETHERMON_ADDRESSES.LIS is used to supply any missing components.

The /PROTOCOL=*protocol* qualifier allows you to restrict address display to only those addresses that have used the protocol specified (using one of the formats described in section 3.2.1.2). By default, all addresses (subject to other restrictions) are displayed, regardless of protocol used. Note that there is no way to specify that the protocol is IEEE 802.3 SNAP or plain-vanilla Ethernet; both kinds will be displayed if both kinds exist for a specified value.

The /TYPE=*value* qualifier allows you to specify that the listing should only include address table entries which have the specified attributes.The value can be:

1. a hexadecimal number in the form %X*nnnn*, where the numeric values correspond to the internal binary flag codes used in the address table (and seen as the M*xxxx*: number in SAVE/ADDRESS files);
2. one or more symbolic names of the flag bits. if more than one name is specified, the names must be separated by commas "," and the entire list enclosed in parenthesis "()". The symbolic type names are (with their associated binary hex values):
   - DECBRIDGE (%X00080): identified as DEC LAN Bridge
   - DECNET (%X00008): running DECnet protocols
   - DECSERVER (%X00100): identified as DEC LAT terminal server
   - DESTINATION (%X00004): node has been a destination
   - DNSSERVER (%X10000): node is an Internet DNS server
   - IARP (%X20000): node's IP address confirmed by ARP
   - INETGATEWAY (%X08000): node is an Internet IP gateway/router
   - INETNAME (%X00800): node has a symbolic Internet name defined
   - INTERNET (%X00010): running DoD Internet IP protocols
   - LAVC (%X00200): running DEC Local Area VAXcluster protocols
   - LINE2ADDR (%X80000): displayed address is for 2nd port of two-port device
   - MACNUM (%X02000): running AppleTalk protocols
   - MACZONE (%X01000): identified as AppleTalk zone router

- MOP (%X04000): running DEC MOP protocols
- OTHERADDR (%X40000): node has two addresses because it is a two-port device
- SOURCE (%X00002): node has been a source
- UB (%X00020): node is running Ungermann-Bass protocols
- VITALINK (%X00400): identified as Vitalink bridge
- XYPLEX (%X00040): running Xyplex terminal net protocols

Additional information will be displayed after the Ethernet physical address if such information has been collected on the node. Such information can include (depending on the protocols used by the node):

- the DECnet node name (as understood by the local VAX's DECnet volatile database); if the specified node number is not listed in the local DECnet volatile database, the node address in DECnet area.node format will be displayed instead.
- the DoD Internet protocol address, in dotted numeric notation. If the LOAD/INTERNET command has been used, the node may also have a host name displayed in the format known to the local TCP/IP software (assuming that the IP address was added to the address table before the LOAD/INTERNET command was issued and that the local node has access to a name/address pair for the specified IP address). If the node also speaks the Routing Information Protocol or RIP (documented in NIC.DDN.MIL Request For Comments RFC:RFC1058.TXT), an asterisk (*) will be displayed before the host name or address to indicate that the node is apparently a router.
- DEC interface characteristics (if the interface is of Digital Equipment Corporation manufacture or emulates such); this information is collected from the DEC MOP Remote Console protocol broadcasts (the System ID messages) made periodically by all DEC-compatible interfaces which announce the interface characteristics. Such entries are of the form device-type:attribute;attribute..., where the device-type is either a DEC interface type (DEUNA, DEBNA, etc.) or t=*xx*, where *xx* is a one-byte hexadecimal number representing a type code unknown to ETHERMON. The known type codes include:
    1. '01'X: deuna (UNIBUS interface card)
    2. '05'X: deqna (Q-BUS interface card)
    3. '0B'X: delua (UNIBUS interface card)
    4. '15'X: debet (DEC LAN Bridge 100)
    5. '17'X: debna (BI interface card)
    6. '21'X: lat?? (LAT terminal server)
    7. '25'X: delqa (Q-BUS interface card)
    8. '27'X: desva (VAXstation built-in interface)
    9. '2B'X: PC??1
    10. '2D'X: TrfMn (DECbridge running LAN Traffic Monitor??)
    11. '35'X: db200 (DEC LAN Bridge 200)
    12. '3F'X: vx??1
    13. '40'X: vx??2
    14. '41'X: debni (BI interface card)
    15. '42'X: demna
    16. '4B'X: deqta (Q-BUS interface card)

17. '4C'X: db150 (DEC LAN Bridge 150, improved DEBET)
18. '72'X: vt1200 (DEC X-terminal)
19. '75'X: MacII (certain brand of Apple Macintosh Ethernet interface)
20. '79'X: CSrtr (multi-protocol bridge/router manufactured by cisco Systems, Inc.)

The attributes reported by a given node vary, but can include any of the following:

1. f=*xx* - hexadecimal flags byte for the functions this interface can perform; bit values are:
   a. bit 0: loop
   b. bit 1: dump
   c. bit 2: primary loader (can only load secondary loader)
   d. bit 3: multi-block loader (can load tertiary loader or system)
   e. bit 4: boot
   f. bit 5: console carrier
   g. bit 6: data link counters
   h. bit 7: console carrier reservation
2. s=*x* - system processor type code
3. hw*aabbccddeeff* - hardware (ROM) Ethernet address of interface (if address has been changed by node's local software)
4. log*aabbccddeeff* - logical Ethernet address of interface (for interface's listing under ROM address)

- the AppleTalk node number in the form Mac*network.node*, where *network* is the AppleTalk Phase II network number (or zero if a Phase I node), and *node* is the node number within the logical network. If the node is also an AppleTalk router, the node's routing zone is appended after the network address in the form rt *zone-name*.
- the Ungermann-Bass host serial number if the node speaks XNS and has an Ungermann-Bass manufacturer's prefix, in the form #*nnnnn*, where *nnnnn* is the decimal serial number of the U-B device. The '#' may be prefixed by 'BRG' (if this node is the root bridge of the spanning tree) or 'tr' (if the node is on a Token Ring).
- the Xyplex host serial number if the node speaks the Xyplex proprietary terminal network protocol and has a Xyplex manufacturer's prefix, in the form unit*nnnnn*, where *nnnnn* is the decimal serial number of the Xyplex device.

3.2.30. SHOW BRIDGES

The SHOW BRIDGES command displays the addresses and symbolic identifications of all Ethernet devices tentatively identified as bridges. Optionally, it also displays a "best-approximation" overall bridge routing "map" as created by the MAP command.

Command format:

```
SHOW BRIDGES [/FULL]
             [/MAP[=filespec]]
             [/OUTPUT[=filespec]]
```

The /FULL qualifier specifies that the protocol types used by each bridge are displayed after that address when the table is displayed (currently, ETHERMON can save up to 10 protocol types {Ethernet or IEEE 802.3} per Ethernet address; the protocol type names {if any} from the protocol names table will be displayed after the physical numbers), along with any routing "map" information for each bridge that was computed by the last MAP command's execution.

The /OUTPUT=filespec qualifier allows you to redirect the SHOW command's display to a file instead of the user terminal. If the file specification is missing or incomplete, the name SYS$DISK:[]ETHERMON_BRIDGES.LIS is used to supply any missing components.

The /MAP=filespec qualifier tells ETHERMON to generate a graphical sketch of the bridged Ethernet, instead of a tabular report. The plot is generated as a stand-alone portable Level 1 PostScript document. Due to limited information available from the bridges, the map is not guaranteed to be totally accurate; however, it is a good attempt. If the file specification is missing or incomplete, the name SYS$DISK:[]BRIDGE_MAP.PS is used to supply any missing components.

Note that neither form of the map is guaranteed to be totally accurate; since bridges may not respond to MAP command queries, or the responses may get dropped due to overloading of the monitoring VAX; in such cases, the information used to construct the maps will be obviously incomplete. To verify that a bridge actually supplied information to the MAP command (even if it was irrelevant information), the /FULL output format will display question marks after the identification of any bridge that did not supply information.

3.2.31. SHOW FILTERS

The SHOW FILTERS command displays the current contents of the filter table, listing first the size filter thresholds and its enable/disable status. This is followed by all the currently defined filters, where each filter displays its index number, current enable/disable status, and the attributes it filters on, where Ethernet addresses and protocol types may be followed by their equivalence names (if any) from the address, manufacturer, and protocol tables.

Command format:

SHOW FILTERS

3.2.32. SHOW KEY

The SHOW KEY command displays the current key definitions as set up by the DEFINE/KEY command. All key definitions are displayed in the form of the DEFINE/KEY commands needed to re-create the definitions. Definitions that are "hard-wired" into ETHERMON have an asterisk (*) displayed before their DEFINE/KEY commands to indicate that these are non-redefineable keys.

Command format:

SHOW KEY

3.2.33. SHOW MANUFACTURERS

The SHOW MANUFACTURERS command displays the current contents of the manufacturers' address prefix table on the user terminal, followed by the current table size. This includes all pre-defined "hardwired" names and all prefix codes automatically entered by the message analyzing logic. Manufacturer address prefixes which have been seen on the local Ethernet since starting ETHERMON will have a letter displayed in front of the address prefix; the letter will be a "s" (for source only), "d" (for destination only), or "b" (used both as source and destination).

Command format:

```
SHOW MANUFACTURERS [/ALL]
                   [/OUTPUT=filespec]
```

Note that, unless the /ALL qualifier is specified, only manufacturer prefix codes that have actually been seen on the Ethernet since ETHERMON was started will actually be displayed.

The /ALL qualifier specifies that all manufacturer codes, not just active ones, will be displayed.

The /OUTPUT=filespec qualifier allows you to redirect the SHOW command's display to a file instead of the user terminal. If the file specification is missing or incomplete, the name SYS$DISK:[]ETHERMON_MANUFACTURERS.LIS is used to supply any missing components.

3.2.34. SHOW MESSAGES

The SHOW MESSAGES command displays the current settings of the SET MESSAGES command for enabling or disabling output of asynchronous "significant event" messages to the user terminal.

Command format:

```
SHOW MESSAGES
```

3.2.35. SHOW PERFORMANCE

The SHOW PERFORMANCE command is a debugging command for ETHERMON itself; it supplies internal performance data about the operation of ETHERMON in the current network environment. The output format of this command is not documented, as it may change at any time without notice; however, the use of this command will not damage other operations being performed with ETHERMON.

Command format:

```
SHOW PERFORMANCE [/INTERVAL=deltatime]
```

where /INTERVAL specifies a standard VMS *deltatime* value (without enclosing double quotes, and without a day count); if /INTERVAL is not specified, the current SET INTERVAL value is used as the sample period length.

3.2.36. SHOW PORT_STATUS

The SHOW PORT_STATUS command displays the current Ethernet port's current status. Some of the status is global for the entire interface, where other status is peculiar to ETHERMON's connection to the interface. For more information, see the *VMS I/O User's Reference Manual: Part II* (the chapter on Ethernet interfaces) for information about the IO$_SENSEMODE function's extended characteristics buffer, and the *VMS Network Control Program Manual* for information about line counters.

Command format:

```
SHOW PORT_STATUS [/ZERO]
```

where /ZERO can be optionally specified to zero the interface's counters after displaying the old data (similar to the DECnet NCP command ZERO LINE COUNTERS). Note that some port parameters may show up as question marks, i.e. *???*; this implies that those parameters are not returned by the type of interface being used.

3.2.37. SHOW PROTOCOL_NAMES

The SHOW PROTOCOL_NAMES command displays the current contents of the protocol table on the user terminal, followed by the current table size. This includes all pre-defined "hardwired" names, all names manually entered with the SET PROTOCOL_NAME command, and all protocol types actually seen on the Ethernet since ETHERMON was started. Protocol types which have been seen on the local Ethernet since ETHERMON was started will have an asterisk (*) displayed in front of the protocol type number.

Command format:

```
SHOW PROTOCOL_NAMES [/ALL]
                    [/OUTPUT=filespec]
```

Note that, unless the /ALL qualifier is specified, only protocol types that have actually been seen on the Ethernet since ETHERMON was started will actually be displayed.

The /ALL qualifier specifies that all protocol types, not just active ones, will be displayed.

The /OUTPUT=filespec qualifier allows you to redirect the SHOW command's display to a file instead of the user terminal. If the file specification is missing or incomplete, the name SYS$DISK:[]ETHERMON_PROTOCOLS.LIS is used to supply any missing components.

3.2.38. SPAWN

The SPAWN command spins off a DCL subprocess, either executing a single command or beginning DCL prompting. While the subprocess is SPAWN'ed, ETHERMON continues to record address, manufacturer, and protocol type information in its tables, and any active asynchronous TRACE command continues to collect data.

Command format:

SPAWN [*DCL-command-string***]**

Note that this command functions similarly to the DCL SPAWN command, but with fewer options. Also, you must have sufficient PRCLM and other quotas to SPAWN a subprocess while ETHERMON is consuming most of your available BYTLM resources; ETHERMON computes how much BYTLM it must leave unused to guarantee that one minimal-quota SPAWNed subprocess and one TRACE/ANALYZER command may be succesfully created, but it does not check other quotas.

3.2.39. SUMMARIZE DESTINATIONS

The SUMMARIZE DESTINATIONS command collects data over a specified time period, then generates a report of counts of messages sent to destinations, sorted by descending quantity of messages.

Command format:

```
SUMMARIZE DESTINATIONS [/INTERVAL=deltatime]
                       [/OUTPUT[=file]]
```

where /INTERVAL specifies a standard VMS *deltatime* value (without enclosing double quotes, and without a day count); if /INTERVAL is not specified, the current SET INTERVAL value is used as the sample period length.

The /OUTPUT=filespec qualifier allows you to redirect the SUMMARIZE command's display to a file instead of the user terminal. If the file specification is missing or incomplete, the name SYS$DISK:[]ETHERMON.SUM is used to supply any missing components.

Only messages passing the currently enabled filters are counted in the summation. The output report is printed to the user terminal; after the report header, each line of data gives the number of messages received, the physical Ethernet address they were sent to, and any equivalence names known for the address.

3.2.40. SUMMARIZE INTERNET

The SUMMARIZE INTERNET command collects data over a specified time period, then generates a report of counts of Internet messages sent under different second level Internet protocols (such as TCP, UDP, etc.), sorted by descending quantity of messages.

Command format:

```
SUMMARIZE INTERNET [/INTERVAL=deltatime]
                   [/OUTPUT[=file]]
```

where /INTERVAL specifies a standard VMS *deltatime* value (without enclosing double quotes, and without a day count); if /INTERVAL is not specified, the current SET INTERVAL value is used as the sample period length.

The /OUTPUT=filespec qualifier allows you to redirect the SUMMARIZE command's display to a file instead of the user terminal. If the file specification is missing or incomplete, the name

SYS$DISK:[]ETHERMON.SUM is used to supply any missing components.

Only messages passing the currently enabled filters are counted in the summation; an additional filter is automatically defined and enabled for the duration of the command to ensure that only messages under protocol type 08-00 (DoD Internet IP) are processed. The output report is printed to the user terminal; after the report header, each line of data gives the number of messages received, the second level protocol number, and the mnemonic name of the second level protocol (per NIC.DDN.MIL Request For Comments RFC:RFC1060.TXT). For the TCP (6) and UDP (17) second-level protocols, counts on a per-port basis are also kept.

3.2.41. SUMMARIZE PROTOCOLS

The SUMMARIZE PROTOCOLS command collects data over a specified time period, then generates a report of counts of messages sent under different protocol types, sorted by descending quantity of messages.

Command format:

```
SUMMARIZE PROTOCOLS [/INTERVAL=deltatime]
                    [/OUTPUT[=file]]
```

where /INTERVAL specifies a standard VMS *deltatime* value (without enclosing double quotes, and without a day count); if /INTERVAL is not specified, the current SET INTERVAL value is used as the sample period length. The /OUTPUT=filespec qualifier allows you to redirect the SUMMARIZE command's display to a file instead of the user terminal. If the file specification is missing or incomplete, the name SYS$DISK:[]ETHERMON.SUM is used to supply any missing components.

Only messages passing the currently enabled filters are counted in the summation. The output report is printed to the user terminal; after the report header, each line of data gives the number of messages received, the physical Ethernet protocol type code they were sent under, and any equivalence name known for the protocol type.

3.2.42. SUMMARIZE SIZES

The SUMMARIZE SIZES command collects data over a specified time period, then generates a report of counts of messages of various size categories, sorted by descending quantity of messages.

Command format:

```
SUMMARIZE SIZES [/INTERVAL=deltatime]
                [/OUTPUT[=file]]
```

where /INTERVAL specifies a standard VMS *deltatime* value (without enclosing double quotes, and without a day count); if /INTERVAL is not specified, the current SET INTERVAL value is used as the sample period length. The /OUTPUT=filespec qualifier allows you to redirect the SUMMARIZE command's display to a file instead of the user terminal. If the file specification is missing or incomplete, the name SYS$DISK:[]ETHERMON.SUM is used to supply any missing components.

Only messages passing the currently enabled filters are counted in the summation. The output report is printed to the user terminal; after the report header, each line of data gives the number of messages received and the message size range the messages were in.

3.2.43. SUMMARIZE SOURCES

The SUMMARIZE SOURCES command collects data over a specified time period, then generates a report of counts of messages sent from sources, sorted by descending quantity of messages.

Command format:

```
SUMMARIZE SOURCES [/INTERVAL=deltatime]
                  [/OUTPUT[=file]]
```

where /INTERVAL specifies a standard VMS *deltatime* value (without enclosing double quotes, and without a day count); if /INTERVAL is not specified, the current SET INTERVAL value is used as the sample period length. The /OUTPUT=filespec qualifier allows you to redirect the SUMMARIZE command's display to a file instead of the user terminal. If the file specification is missing or incomplete, the name SYS$DISK:[]ETHERMON.SUM is used to supply any missing components.

Only messages passing the currently enabled filters are counted in the summation. The output report is printed to the user terminal; after the report header, each line of data gives the number of messages received, the physical Ethernet address they were sent from, and any equivalence names known for the address.

3.2.44. SUMMARIZE STATISTICS

The SUMMARIZE STATISTICS command collects data over a specified time period, then generates a report of Ethernet port counter value changes.

Command format:

```
SUMMARIZE STATISTICS [/INTERVAL=deltatime]
                     [/OUTPUT[=file]]
```

where /INTERVAL specifies a standard VMS *deltatime* value (without enclosing double quotes, and without a day count); if /INTERVAL is not specified, the current SET INTERVAL value is used as the sample period length.

The /OUTPUT=filespec qualifier allows you to redirect the SUMMARIZE command's display to a file instead of the user terminal. If the file specification is missing or incomplete, the name SYS$DISK:[]ETHERMON.SUM is used to supply any missing components.

Note that, for counters that have saturated, a triple asterisk ('***') will in the right margin of the report for the saturated values; if the counters are already saturated on initiation of the SUMMARIZE command, the command will be rejected with an appropriate error.

3.2.45. TRACE

The TRACE command collects Ethernet messages after filtering and writes them to a disk file for offline processing by another program. The messages can be recorded in either ASCII or binary form.

Command format:

```
TRACE [/ASCII] [/BINARY] [/SNIFFER] [/ANALYZER=imagename]
      [/INTERVAL=deltatime]
      [/OUTPUT=outfilespec]
      [/PARALLEL]
      [/REPEAT[=n]]
      [/[NO]STATISTICS]
      [/TRUNCATE=m]

TRACE /ABORT
```

where only one of /ASCII, /BINARY, /SNIFFER, or /ANALYZER may be specified (if none of them is specified, /BINARY is the default). /INTERVAL specifies a standard VMS *deltatime* value (without enclosing double quotes, and without a day count); if /INTERVAL is not specified, the current SET INTERVAL value is used as the sample period length. /OUTPUT specifies the output file name (if file specification is incomplete or qualifier is missing, the default is SYS$DISK:[]ETHERMON.DAT for binary mode, SYS$DISK:[]ETHERMON.LIS for ASCII mode, SYS$DISK:[]ETHERMON.SNIFFER for Sniffer mode, and SYS$OUTPUT: for analyzer mode). The /PARALLEL qualifier specifies that recording should go on in the background, leaving the terminal free to execute other ETHERMON commands. The /REPEAT[=n] qualifier also specifies background recording, but specifies that after each interval, the recording file should be closed and a new version opened, keeping the recording running forever or until aborted (whichever comes first); to prevent running out of disk space, the optional *n* is used to specify a maximum version count allowed for the recording files (automatic file purging is done when more than *n* files are collected). Note that the /REPEAT qualifier is ignored if /ANALYZER is specified (only one analyzer session will be done). The /[NO]STATISTICS qualifier is used to override the default choice of reporting TRACEd packet statistics; by default, statistics are reported at the end of a synchronous (non-/PARALLEL) TRACE command, and are not at the end of a /PARALLEL or /REPEAT command's background execution. /TRUNCATE=*m* allows recording only the first *m* bytes of each packet in a binary or Sniffer-format recording file; by default, all bytes are recorded, and there is no choice for ASCII files (the 40 bytes that are reported are fewer than the minimum physical packet length anyway). It is strongly recommended that records not be truncated to shorter than the length of an IEEE 802.3 SNAP header (22 bytes), as there would not be enough information left in the file's records to be able to decipher the data.

The TRACE/ABORT command is used to cleanly shut down a TRACE/PARALLEL or TRACE/REPEAT command; it specifies that the currently open file should be closed immediately and all tracing should cease. Note that an asynchronous TRACE will also be terminated if ETHERMON is exited; parallel TRACEing does NOT spin off a background batch job or subprocess to do the parallel monitoring, but just saves the data directly from the data collection AST. Also, you cannot perform a foreground TRACE and a background TRACE at the same time. The /[NO]STATISTICS qualifier may be used on the TRACE/ABORT command to override the reporting choice made when the asynchronous TRACE session was started.

Several unsupported utilities are supplied to help analyze binary TRACE file recordings, and may be used as a template for your own expanded offline message analysis tools. Some of these may also be used as analyzer modules with the /ANALYZER qualifier. For more information on these utilities, see section 4 of this document.

For ASCII mode, the output file contains three lines for each received and filter-passed Ethernet message. The first line contains the VMS timestamp when the message was received, the source and destination addresses in standard Xerox ASCII-hexadecimal format, the protocol type (also ASCII-hex), and the message body's length in bytes. The second line is a hexadecimal dump of the first 40 bytes of the received message; the third line is the same 40 bytes dumped in ASCII (with a space before each actual character, such that the ASCII characters line up under their hexadecimal bytes; periods are displayed for unprintable characters). Preceding the dump lines is a single line reporting when the dump began; following the dump lines are three lines reporting when the dump was completed, and the quantity of data bytes and messages processed from the Ethernet before and after filtering. The file has carriage-return carriage control and variable-length records. A typical ASCII recording file looks like this:

```
  Ethernet TRACE: starting data collection at 21-JUN-1991 09:28:54.22
21-JUN-1991 09:28:54.29 s:AA-00-04-00-CA-28 d:AA-00-04-00-D1-28 p:60-07    4
1400AA000400D1280500AA000400CA280800AA3400000000000000000000000000000000000000
 . .   . . .   ( . .   . . .   ( . .   4 . . . . . . . . . . . . . . . . . . . .
21-JUN-1991 09:28:54.40 s:08-00-7C-00-30-26 d:09-00-2B-01-00-01 p:80-38    4
E1190100003108007C003B7101DE007F08007C0030260101050F1E0F3004000081000100000D
   . . . . 1 . . | . ; q .   . . . . | . 0 & . . . . . . . 0 . . . . . . . . .
21-JUN-1991 09:28:54.43 s:00-DD-01-02-96-2C d:08-00-2B-10-1A-E5 p:06-00    4
FFFF002A000500000000108002B101AE5872C0000000100DD0102962C8713C000075EC05F0039
 . . . * . . . . . . . . + . .   . , . . . . .   . . . , . .   . . ^   _ . 9
21-JUN-1991 09:28:54.44 s:00-DD-01-0A-12-A8 d:FF-FF-FF-FF-FF-FF p:06-00    7
FFFF0046000400000000FFFFFFFFFFFF800100000000100DD010A12A88020000000000000010000
 . . . F . . . . . . . . . . . . . . . . . . . .   . . .   .   . . . . . . . .
21-JUN-1991 09:28:54.47 s:AA-00-04-00-D8-28 d:AB-00-00-03-00-00 p:60-03    4
22000D020000AA000400D82803DA05000000000000000000000AA00040063280F000002AAAA0000
 " . . . . .   . . .   ( .   . . . . . . . . . .   . . . c ( . . . .   . .
21-JUN-1991 09:28:54.48 s:00-DD-01-08-50-A2 d:AA-00-04-00-3D-30 p:80-9Bi   1
0011000013601360C7A60606060101835B
 . . . . . ‘ . ‘   . . . . . . [
21-JUN-1991 09:28:54.48 s:00-DD-01-08-50-A2 d:AA-00-04-00-3D-30 p:80-9Bi   1
0011000013601360C7A60606060101EB62
 . . . . . ‘ . ‘   . . . . .   b

                              .
                              .
                              .

21-JUN-1991 09:28:57.26 s:AA-00-04-00-F5-28 d:AA-00-04-00-CE-28 p:60-07    8
5500AA000400CE280500AA000400F5280B00D2BDF3BD023E0004000A0001003000966A0900D3
 U .   . . .   ( . .   . . .   ( . .       . > . . . . . . . 0 . . j . .
  Ethernet TRACE: ending data collection at 21-JUN-1991 09:28:57.27
         15805 bytes,  234 messages analyzed
         15805 bytes,  234 messages received before filtering
```

For binary mode, the file contains one variable-length record per received and filter-passed Ethernet message. A set of structure definitions of the records for various programming languages are listed below. The FORTRAN version of this structure definition is supplied in the file ETHTRSTRUC.INC.

FORTRAN:

```
      STRUCTURE /BIN_REC/
        INTEGER*4 VMS_DATETIME(2) ! quadword system timestamp
        UNION
          MAP
            BYTE      DSTADDR(0:5) ! beginning of XEDRIVER P5 buffer: dest a
            BYTE      SRCADDR(0:5) ! source Ethernet address
          END MAP
          MAP
            INTEGER*2 W_DSTADDR(0:2)
            INTEGER*2 W_SRCADDR(0:2)
          END MAP
          MAP
            INTEGER*4 L_DSTADDR
            INTEGER*2 W_DSTADDR4
            INTEGER*4 L_SRCADDR
            INTEGER*2 W_SRCADDR4
          END MAP
        END UNION
        UNION
          MAP
            CHARACTER*8 PROTBUF
          END MAP
          MAP
            BYTE    PROTOCOL(0:1) ! Ethernet protocol type
          END MAP
          MAP
            INTEGER*2 W_PROTOCOL
          END MAP
          MAP
            BYTE    DSAP        ! IEEE 802 destination SAP
            BYTE    SSAP        ! IEEE 802 source SAP
            UNION
              MAP
                BYTE  CTL(0:1)  ! IEEE 802 CTL byte(s)
              END MAP
              MAP
                BYTE  ECTL      ! IEEE extended 802 CTL byte
                BYTE  PID(0:4)  ! IEEE extended 802 Protocol IDentification
              END MAP
            END UNION
          END MAP
        END UNION               ! end of XEDRIVER P5 buffer (20 bytes)
        INTEGER*2 LENGTH        ! length of message body
        UNION
          MAP
            BYTE      BODY(4470)! not all of this array may be sent; only
          END MAP               ! part actually used is written to file
          MAP
            CHARACTER*4470 CBODY
          END MAP
        END UNION
      END STRUCTURE ! /BIN_REC/
```

MACRO:

```
        $DEFINI REC
```

```
$EQU    REC$Q_TIME      0    ; VAX/VMS binary timestamp
$EQU    REC$B_DESTADDR  8    ; 6-byte destination address
$EQU    REC$B_SRCADDR   14   ; 6-byte source address
$EQU    REC$B_PROTOCOL  20   ; 2-byte Ethernet protocol type
$EQU    REC$B_DSAP      20   ; 1-byte IEEE 802 DSAP
$EQU    REC$B_SSAP      21   ; 1-byte IEEE 802 SSAP
$EQU    REC$B_CTL       22   ; 1- or 2-byte IEEE 802 CTL field
$EQU    REC$B_ECTL      22   ; 1-byte IEEE extended 802 CTL field
$EQU    REC$B_PID       23   ; 5-byte IEEE extended 802 protocol ID
$EQU    REC$W_LENGTH    28   ; word length of message body
$EQU    REC$T_BODY      30   ; body of message
$EQU    REC$K_MAXLEN    4500 ; maximum length of file record
        $DEFEND REC
```

Note that the above file record format definitions assume a fixed size of the protocol type field; if the protocol field of a particular Ethernet packet is shorter than the maximum size, the field is padded out with zero bytes. Also, the network CRC field is not recorded, nor is the length field in the header of an IEEE 802.3 packet, nor the padding to minimum legal packet length of an IEEE 802.3 packet with less than the minimum number of actual data body bytes. Therefore, the records in a binary-format file are NOT in the same format as they were as network packets.

To get a closer approximation of the actual network packet format, the Sniffer format can be used. It is also a binary file format, but contains ONLY the used header fields (including length if IEEE 802.3, unlike the binary format) and the message body; unused header fields, the CRC, and IEEE 802.3 padding are not stored in the file's records. No timestamps are recorded in a Sniffer-format file.

All the unsupported utilities delivered with ETHERMON (described in section 4) expect to read binary format files (not Sniffer format).

To analyze data in real-time in greater detail than TRACE/ASCII is capable of doing, or to provide additional filtering capabilities beyond the existing filtering logic in ETHERMON, the /ANALYZER qualifier may be used to spin off a subprocess that will be fed the recorded data in real-time. The subprocess is created with the following environment:

- SYS$INPUT is a VMS temporary mailbox with no logical name, specified by physical device name. this mailbox is written with network packet data in the binary file format, and is written with an end-of-file when the TRACE terminates (either due to completion timeout or a TRACE/ABORT command). the mailbox is written with resource wait mode disabled, so if the mailbox overflows, the main ETHERMON process will not be suspended, but some data will be lost (a count will be kept of SS$_MBFULL errors incurred, and reported in the TRACE/STATISTICS messages); the subprocess will not be informed of message losses.
- SYS$OUTPUT defaults to the SYS$OUTPUT file of the process running ETHERMON, but may be redirected with the /OUTPUT qualifier on the TRACE command; interpretation of this file specification is done by the selected analyzer module.
- SYS$ERROR is the ETHERMON process's SYS$ERROR.
- default disk and directory is the same as the process running ETHERMON.
- base priority is the base priority of the ETHERMON process.
- privileges are those of the ETHERMON process (including those automatically enabled by ETHERMON, unlike for subprocesses created by SPAWN).
- no CLI is present; the subprocess is created by SYS$CREPRC, not by LIB$SPAWN.

- all quotas are set to system defaults (see the *VMS System Services Reference Manual* for details on subprocess quota allocations).
- process name is the parent ETHERMON process's process name, prefixed with the string 'Eanl_' (truncated to fit in 15 characters).
- a termination mailbox is defined, and will be used to keep track of the subprocess even after the SYS$INPUT mailbox is closed and deleted.
- image dumping is enabled.
- the image file being run must be a .EXE file in the ETHERMON executable image's directory, with the filename being the name specified on the /ANALYZER=*imagefile* qualifier.
- a writeable temporary group global section containing the address table COMMON is created and may be mapped by the subprocess to allow it access to the analytical data already computed by ETHERMON. the supplied subroutine ETH_MAP_ADDRTBL (documented in section 4) may be used to map the section.

You must have sufficient PRCLM and other quotas to create a subprocess while ETHERMON is consuming most of your available BYTLM resources; ETHERMON computes how much BYTLM it must leave unused to guarantee that one minimal-quota SPAWNed subprocess and one TRACE/ANALYZER command may be succesfully created, but it does not check other quotas.

No additional parameters may be passed to the subprocess by this command (although the analyzer module could examine shareable logical name tables for information). The analyzer module may do anything it wishes with the data it receives from the mailbox.

Only one analyzer module may execute at a time; even if an analyzer module has had its input data stream terminated, the image will be allowed to run to its chosen completion (allowing post-processing of the data as it sees fit). As such, another TRACE/ANALYZER command may not be issued until the previous subprocess has terminated; if the previous subprocess is still running, the TRACE/ANALYZER command will be rejected. If an analyzer module terminates pre-maturely, the TRACE command (synchronous or asynchronous) will be aborted with a message reporting that the analyzer image died.

3.2.46. WAIT

The WAIT command allows suspending command execution of ETHERMON while maintaining data collection. This is primarily useful for batch monitoring jobs which want to use the TRACE/REPEAT command, but don't want ETHERMON to exit after initiating data collection.

Command format:

```
WAIT hh:mm:ss.cc [message]

WAIT FOREVER [message]
```

where *hh:mm:ss.cc* is a standard VMS delta time (without a days value), and FOREVER means wait until external events continue execution (no timer is started to end a FOREVER wait). The optional *message* is a string of text which will be displayed on the standard output device (terminal or batch log file) before the WAIT begins. This is useful for ETHERMON indirect command files, where the commands being executed cannot be seen by the user.

Since the WAIT command suspends execution in a main-level user-mode $HIBERnate, data collection continues. The WAIT can be terminated before its specified end time by entering any keyboard character (for interactive mode use) or by issuing a $WAKE service request at the process running ETHERMON (given appropriate access privileges). A message will be output indicating how the WAIT was aborted.

A sample DCL command procedure, ETHERMON_ALL.COM, is supplied as an example of how to use the WAIT and TRACE/REPEAT commands to create a 24-hour-a-day network monitor. The sanple looks like this:

```
$ ETHERMON :== $'F$ENV("DEFAULT")'ETHERMON
$ ETHERMON ETA          ! assuming a DEBNA or DEBNI on the system
SET MESSAGES OFF
WAIT 00:00:10 "Wait for address table to fill"
ADD FILTER/ONLY=(ADDRESS=C5VR,PROTOCOL=08-00)
TRACE/REPEAT=20/INTERVAL=00:05:00/BINARY/OUTPUT=C5VR
WAIT FOREVER "Now collecting IP data from NFS Server node..."
```

where this particular application first waits for the address table to be populated (the first WAIT command), then defines a filter to monitor the Internet IP traffic in and out of the DECnet node named C5VR, then starts a repeating asynchronous TRACE command that collects a series of files (each covering a 5 minute interval) for a total holding period of 1 hour 40 minutes. It then WAITs FOREVER (so as to maintain the 100 minute sliding window of collected data indefinitely).

4. Utilities

Seven offline analysis utilities are supplied with ETHERMON for use in deciphering binary TRACE files. These are:

1. ETHERPARSE - the first attempt at an analysis tool
2. ETHERPARSE2 - a fancier version of ETHERPARSE; it can decipher AppleTalk packet headers, Internet IP, TCP, UDP, RIP, RPC, NFS, RPC portmapper, and SNMP headers, XNS packet headers, DEC LAN Bridge packets, and ARP/RARP packets. This program may be used for a TRACE/ANALYZER subprocess.
3. ETHERPARSE_TCP - a specialized parser, it matches up packets from TCP sessions, using remote/local host-address/port-number to uniquely identify sessions, and reports on the status of the session flow control counters (SEQ and ACK, from the RFC793 specification of the TCP protocol) for each session.
4. ETHERPARSE_XREF - a specialized parser, it matches up source and destination hosts, and counts the number of packets sent by each combination, then reports a list of such pairs sorted as to most active pairs. Very slow program!
5. ETHFILT_ETHADDR - a secondary filter to help fine-tune the results of a SET FILTER command in ETHERMON; it selects only records from a recording file that contain the user-supplied Ethernet address and copies them to another file.
6. ETHFILT_IPADDR - another secondary filter to help fine-tune the results of a SET FILTER command in ETHERMON; it selects only records from a recording file that contain the user-supplied DoD Internet IP address and copies them to another file.
7. ETHFILT_NFS - yet another secondary filter; it passes only records that are speaking DoD Internet IP, and furthermore are speaking Sun Microsystems' Network File System (NFS) protocol.

There are also two support subroutines which are available to help programs connect up with ETHERMON as TRACE/ANALYZER subprocesses. These are:

1. ETH_CHECK_SUBPROC - this subroutine accepts a dummy argument (it ignores it, but the variable is required for FORTRAN syntax), and returns a LOGICAL*1 value indicating whether or not the calling program is in a CLI-less subprocess (.TRUE. if environment appears to be a TRACE/ANALYZER subprocess).
2. ETH_MAP_ADDRTBL - this subroutine does everything that ETH_CHECK_SUBPROC does, but then attempts to map the parent ETHERMON's address table COMMON global section into this process's address space; the procedure accepts a LOGICAL*1 argument that is .TRUE. if the subprocess wants read/write access to the table, and .FALSE. if read-only access is desired. The procedure returns a LOGICAL*1 value indicating whether or not the calling program is in a CLI-less subprocess (.TRUE. if environment appears to be a TRACE/ANALYZER subprocess). It signals a fatal error if everything was OK except that the global section was not properly aligned in memory (a COLLECT= linker option was not used to put the /ADDRESS_NAMES/ COMMON in a separate cluster of linker-allocated address space) or there was a mismatch between the ETHERMON main program's version of the section and the calling subprocess's version. The proper linker /OPTIONS file directive is

   ```
   COLLECT=ADDRESS_NAMES,ADDRESS_NAMES
   ```

4.1. ETHERPARSE

ETHERPARSE's source files consist of a single FORTRAN file ETHERPARSE.FOR. This program will prompt for the name of a TRACE binary recording file (default file type .DAT), the name of an output listing file (default file type .LIS), and the optional start and end times of the data to select (if not supplied, all data in the file is accepted; if supplied, they must be valid VAX/VMS absolute times). Very little error checking is done; if an invalid input file is supplied, the program will crash sloppily.

The output file contains a series of formatted packets, where each packet is begun with a blank line and a line containing the packet time, source address, destination address, protocol type (with an 'i' suffix if IEEE 802.3 SNAP protocol), and the packet length in bytes. After this, there are lines of hexadecimal and ASCII representation of the packet data (16 bytes per line), with the hex data going right-to-left, and the ASCII data going left-to-right (with periods representing unprintable characters); each line is ended with a four-hex-digit number giving the offset of this line from the beginning of the data link's data field of the packet. At the end of the report is a list of packet counts for source addresses, destination addresses, and protocol types; the left count is the total for the entire file, and second column is the count within the specified time window (identical if no time limits were given).

4.2. ETHERPARSE2

ETHERPARSE2's source files consist of a FORTRAN file ETHERPARSE2.FOR, and a MACRO source file ETHERPARSE2_CRF.MAR. Its user interface is identical to ETHERPARSE, with the exception of one additional query: does the user want all of each packet dumped in ASCII and hexadecimal, even if part of the packet was also dumped symbolically (by default, any data dumped symbolically is skipped over in generating the ASCII/hex dump). Note that, to link ETHERPARSE2, you must use a LINK command of the form:

$ LINK ETHERPARSE2,ETHERPARSE2_CRF,ETH_MAP_ADDRTBL,SYS$INPUT:/OPT
COLLECT=ADDRESS_NAMES,ADDRESS_NAMES
<ctl-Z>

The additional processing that ETHERPARSE2 supplies consists of:

1. decoding the LLP prefixes of AppleTalk packets, the DDP packet header, and (when valid) the NBP packet body.
2. decoding ARP messages (IP ARP, IP RARP, and AppleTalk ARP)
3. decoding some forms of DEC LAN Bridge control messages
4. decoding XNS network level packet headers
5. decoding DoD Internet message headers: IP (RFC 791, always), TCP and UDP (RFC's 793 and 768, when protocol type is one of these and not a continuation IP packet), ICMP (RFC 792, when protocol type is this), RIP (Routing Information Protocol, RFC 1058, when valid UDP and port number is 520), RPC (Sun Remote Procedure Call, RFC 1057; when valid after either a TCP or UDP header), Sun NFS (Network File System, RFC 1094, after valid UDP and RPC headers that specify NFS), RPC portmapper (RFC 1057, after valid TCP or UDP headers), and SNMP (Simple Network Management Protocol, RFC 1157, after valid UDP header)

Additional reports generated at the end of the formatted listing include any of the following items that had their type of data collected in the data file:

1. cross-reference of XNS socket usage by Ethernet addresses (flagged as source or destination users)
2. per-Ethernet address client usage of NFS service calls, listing the names of the procedures called, and counts for each procedure for number of calls, number of identified successful replies, and number of identified erroneous replies.

ETHERPARSE2 may be used as a TRACE/ANALYZER subprocess.

4.3. ETHERPARSE_TCP

ETHERPARSE_TCP's source files consist of a single FORTRAN file ETHERPARSE_TCP.FOR. This program will prompt for the name of a TRACE binary recording file (default file type .DAT) and the name of an output listing file (default file type .LIS). Very little error checking is done; if an invalid input file is supplied, the program will crash sloppily.

The output file contains a series of formatted packets, where each packet is displayed on a single line containing the packet time, source IP address and TCP port, destination IP address and TCP port, sequence number (with computed error flags), acknowledge number (with computed error flags), window size value, and any additional flags from the TCP header (URG, FIN, PSH, etc.). This processes the TCP header according to the documentation in RFC793. At the end of the report, a summary of packets processed and packets discarded as not being TCP/IP is printed.

4.4. ETHERPARSE_XREF

ETHERPARSE_XREF's source files consist of a single FORTRAN file ETHERPARSE_XREF.FOR. This program will prompt for the name of a TRACE binary recording file (default file type .DAT), the name of an output listing file (default file type .LIS). Very little error checking is done; if an invalid input file is supplied, the program will crash sloppily.

The output file contains a list of packet counts for source addresses, destination addresses, and protocol types. Following this list is one more list of matched pairs of source and destination addresses (displayed in Ethernet physical format and Internet IP [if valid] with packet counts. All lists (except the protocol list) are sorted in descending order of number of packets.

This program was written rather inefficiently with regard to both CPU time and virtual memory consumption; do not expect quick response from it if a large number of different combinations are present in your data sample.

4.5. ETHFILT_ETHADDR

ETHFILT_ETHADDR's source files consist of a single FORTRAN file ETHFILT_ETHADDR.FOR. Its user interface is similar to ETHERPARSE; it prompts for an input TRACE binary recrding file (default file type .DAT), and a name for the to-be-created output binary recording file (default file type .DAT). It then prompts for the Ethernet physical address of the packets (using the fixed format *aa-bb-cc-dd-ee-ff*, where the hyphens are required and each byte must be specified by two hexadecimal digits) to filter for. Any packets which do not contain the specified Ethernet address in either the source or destination address fields of the data link layer header are discarded; the remaining packets are copied to the output file, and a statistical summary of packets processed is printed on the user terminal.

4.6. ETHFILT_IPADDR

ETHFILT_IPADDR's source files consist of two versions of a FORTRAN file, ETHFILT_IPADDR_TGV.FOR and ETHFILT_IPADDR_UCX.FOR, and corresponding linker options files ETHFILT_IPADDR_TGV.OPT and ETHFILT_IPADDR_UCX.OPT. Its user interface is similar to ETHERPARSE; it prompts for an input TRACE binary recording file (default file type .DAT), and a name for the to-be-created output binary recording file (default file type .DAT). It then prompts for the Internet IP physical (numeric, not symbolic) address of the packets to filter for. Any packets which are not IP or do not contain the specified IP address in either the source or destination address fields of the IP header are discarded; the remaining packets are copied to the output file, and a statistical summary of packets processed is printed on the user terminal.

The linker options file is used to specify the object files to be linked. The two versions assume that either TGV MultiNet or DEC VMS/Ultrix Connection (UCX, now called TCP/IP Services for VMS) is the Internet software running on the local system; as such, they specify the appropriate socket library shareable image as a place to get the Internet address format conversion routines (either MULTINET:MULTINET_SOCKET_LIBRARY or SYS$SHARE:VAXCRTL). To link either image, a LINK command of the form

$ LINK/EXE=ETHFILT_IPADDR ETHFILT_IPADDR_*version*/OPTION

should be used, where *version* is either TGV or UCX.

4.7. ETHFILT_NFS

ETHFILT_NFS's source files consist of a single FORTRAN file ETHFILT_NFS.FOR. Its user interface is similar to ETHERPARSE; it prompts for an input TRACE binary recrding file (default file type .DAT), and a name for the to-be-created output binary recording file (default file type .DAT). Any packets which are not IP/UDP or do not contain RPC/NFS data are discarded; the remaining packets are copied to the output file, and a statistical summary of packets processed is printed on the user terminal.

Note that, to link ETHFILT_NFS, you must use a LINK command of the form:

$ LINK ETHFILT_NFS,ETH_CHECK_SUBPROC

ETHFILT_NFS may be used as a TRACE/ANALYZER subprocess.

5. Security Considerations

For system and network administrators who are concerned about illicit monitoring of their networks: ETHERMON does that sort of thing just dandy! Seriously, network monitoring is the whole purpose of ETHERMON; ETHERMON is supposed to be able to see all traffic that passes by on your network. However, there are constraints to help prevent *illicit* monitoring of the network with ETHERMON:

1. ETHERMON requires a fair number of sensitive user privileges to be able to run: PHY_IO, NETMBX, TMPMBX, and CMKRNL, to be precise (or SETPRV, which allows all the others to be enabled). Similarly, it requires rather large user quotas (BYTLM). All of these can only be granted by a user with system administrator privileges. As such, it is highly recommended NOT to install ETHERMON.EXE with privilege.

2. VAX/VMS V5.4-3 and higher *require* the use of an installed protected shareable image, which requires even more privileges to set up. Changes to the installed image list can be easily audited for, and the essential shareable image (ETHERMON_FASTSHR) can be placed under VMS file protections to control access to it.

3. The main executable image file of ETHERMON depends on its location to locate other files. Therefore, illicit copies of the executable file would not be able to access the installed official copy of the shareable image file (unless a system-wide executive-mode logical name was defined *and visible* to the account being used).

This is as opposed to PC and Macintosh-based monitoring tools, such as NetMinder, where simple physical access to the software package and the computer in question is enough to guarantee the ability to run a monitor. Even if a VAX/VMS user was to "roll their own" monitoring software (as ETHERMON V1.0 was originally created), a minimum of PHY_IO privilege would still be required to enable the promiscuous monitoring mode on the Ethernet interface device.

For system administrators who are concerned about un-trusted kernel-mode code (the V2.4 enhancements involving ETHERMON_FASTSHR.EXE) running on their systems, there are four safety checks in place:

1. As with the rest of ETHERMON, complete commented source code is provided for the program, allowing you to study what ETHERMON would be doing to your system. This source code is located in the file ETHERMON_FASTSHR.MAR.

2. Random users cannot take advantage of the kernel mode hooks because the shareable image does not implement a user-written system service; it is implemented as a shareable image only to keep the necessary system version dependencies out of the main ETHERMON code, and to allow a non-defeatable image rundown routine to be executed. Therefore, users of the shareable image still require all the privileges needed to run ETHERMON and the shareable image’s extensions (i.e., users could link their own code to the shareable image, but they would not gain any privilege from it).

3. ETHERMON is capable of functioning without the "fast" port interface for VAX/VMS versions V5.4-2 and below (it requires it for VAX/VMS V5.4-3 and above). To disable ETHERMON’s attempts to use the "fast" interface, simply do not keep a copy of ETHERMON_FASTSHR.EXE in the same directory as ETHERMON.EXE, and/or do not install the image appropriately (as described in section 2.2 on page 7). ETHERMON **cannot** use the shareable image unless it is installed as described above. On a failed attempt to use the shareable image, ETHERMON will print out an appropriate error message describing why it could not activate the "fast" interface, and then either continue using the old interface scheme or terminate (depending on the operating system version).

4. Because the shareable image is activated using the standard services of the VAX/VMS Image Activator, an image that was linked under an incompatible operating system version cannot be activated; however, do not just relink ETHERMON_FASTSHR if this happens,

but re-compile it as well (to ensure that any system macro changes will be accounted for). Note that there should never be any link-time errors from linking the shareable image; if there are, the image is suspect and should not be used. Also, a linker options file (as embedded in the file ETHERMON.COM) must be used to guarantee the proper structure of the shareable image.

Appendix I. Messages

I.1. ETHERMON Messages

1. `%ETHERMON-I-ADRPROTOVF, no room to store` *aa-bb* `for` *aa-bb-cc-dd-ee-ff*

   While automatically updating the address table during a message read, ETHERMON ran out of protocol type slots for a single address entry in the address table, and could not store an additional new protocol type for that address. Verify that the specified address on the Ethernet is supposed to be servicing that many protocol types simultaneously; if this is normal, edit the file ETHERMON_ADDR.INC to increase the value of the parameter ICPADR_MAXPROTS and recompile and relink ETHERMON. This message will only be displayed with the SET MESSAGES DEBUG command active.

2. `%ETHERMON-W-ADRTBLFUL, address table full - new address/name pair not added`

   A SET ADDRESS_NAME or LOAD command attempted to add a new Ethernet address to the address table, but there was no more room for it. This requires a rather complicated procedure to enlarge the size of the address table in the source code (file ETHERMON_ADDR.INC, and scattered changes to address insertion logic throughout ETHERMON.FOR), then rebuilding the program.

3. `%ETHERMON-F-ADRTBLOVF, Ethernet address table overflow - PROGRAM ABORT`

   During a DISPLAY or SUMMARIZE command, there was no longer room in the address table to add a new entry that was being counted up. The program has crashed. To correct this, the address table must be enlarged. However, this also implies that you have an exceedingly busy Ethernet, which you may wish to correct.

Note that this message will only appear while SET MESSAGES DEBUG is in effect (which can happen by default if ETHERMON_CMD_S.FOR is compiled with the /D_LINES option).

4. `%ETHERMON-I-ALLFLTDEL, all defined filters deleted`

   A CLEAR FILTERS/ALL command has been executed successfully. No action is required.

5. `%ETHERMON-E-ALRDYPRMUSR, promiscuous user already active on this interface`

   This error is caused by the fact that there is already a promiscuous user on the Ethernet interface you are attempting to use. Try another interface, or get rid of the existing promiscuous user (it could be another invocation of this program).

6. `%ETHERMON-I-ANLRPT, TRACE/ANALYZER PID=`*xxxxxxxx* `elapsed=`*hh:mm:ss.cc* `CPU=`*hh:mm:ss.cc* `pf=`*nn* `bio=`*nn*`(`*nn* `extra) dio=`*nn*

   The subprocess created by a TRACE/ANALYZER command completed with the specified accounting statistics. An additional message will follow, giving the final completion status of the subprocess. No action is necessary (unless you wish to troubleshoot a non-success completion status).

7. `%ETHERMON-I-ASYNTRCBEG, asynchronous TRACE command started...`

   A TRACE command was entered with the /PARALLEL or /REPEAT qualifier, and is now running in the background while another ETHERMON command prompt is issued. No action is required.

8. `%ETHERMON-I-ASYNTRCDON, asynchronous TRACE command completed.`

   A TRACE command with /PARALLEL completed its time interval, or the TRACE/ABORT command was entered to shut down a TRACE/PARALLEL or TRACE/REPEAT command, or ETHERMON was exited while a TRACE/PARALLEL or TRACE/REPEAT was active. No action is required.

9. `%ETHERMON-W-BADADDRNUM, invalid octet in Ethernet address number`

   A standard format physical Ethernet address was specified where one of the octets was outside the range 00 to FF. Re-enter the command with a valid address.

10. `%ETHERMON-I-BADAPPLE,` *aa-bb-cc-dd-ee-ff* `sent unknown AppleTalk layer`

    The specified node transmitted an AppleTalk/EtherTalk packet where the higher-level protocol type was not recognized (not DDP). Find why the node is sending invalid packets; if the packets turn out to be valid, please send a bug report to the author of ETHERMON, reporting what type of traffic is being mislabeled as invalid. This message can be suppressed with the SET MESSAGES OFF command.

11. `%ETHERMON-W-BADFIL, invalid record in file`

    While reading a SAVE'd address or filter file, an invalid record in the file was found; the record was ignored. If the file was manually edited, use an un-edited version of the file or correct the record's syntax.

12. `%ETHERMON-I-BADPNGADR, can't ping to a multicast address (except Broadcast)`

    The PING command was entered with a multicast address as a target. Use a valid physical (not multicast) Ethernet address when entering the PING command.

13. `%ETHERMON-W-BADPROTNUM, invalid octet in Ethernet protocol number`

    A standard format Ethernet protocol type code was specified where one of the octets was outside the range 00 to FF. Re-enter the command with the correct protocol type code.

14. `%ETHERMON-I-BRGMAPOLD, bridge map is out of date - use MAP command again`

    A new Ethernet bridge made itself known after the last time the MAP command was executed, thereby probably obsoleting the bridge interconnection map computed by the MAP command. This message will appear when using the SHOW BRIDGES/FULL command to display the bridge map. Enter the MAP command again to recompute the map with the additional data.

15. `%ETHERMON-I-BRGPRTINF, vague remote line data obtained; bridge map unreliable`

    While executing a MAP command, bridge port data was not complete for the remote side of some bridge map entries. This is a frequent occurrence if bridges using the DEC LAN Bridge protocol appear on the network, as they do not specify which Ethernet address corresponds to which side of the bridge. No action is required, but it does imply possible errors that might appear in the output of a SHOW BRIDGES/MAP command.

16. `%ETHERMON-W-BRGROUTOVF, routing table overflow for bridge #n`

    While executing a MAP command, the bridge being queried returned more valid bridge addresses than the bridge mapping table could hold. Note that this can only happen when:

    a. executing the MAP command on a large Ethernet with many bridges, and

    b. in the file ETHERMON_BRIDGES.INC file, the parameter ICPBRG_MAXBRGROUT is smaller than the parameter ICPBRG_MAXBRIDGES.

    To correct the problem, edit ETHERMON_BRIDGES.INC to set ICPBRG_MAXBRGROUT equal to ICPBRG_MAXBRIDGES (you may also wish to enlarge ICPBRG_MAXBRIDGES if you find that the SHOW BRIDGES command shows you exactly as many bridges as the current table size); then re-compile and re-link ETHERMON.

17. `%ETHERMON-I-BRGTMO, n timeouts while querying bridges - use MAP command again`

    While executing a MAP command, responses were not received the specified number of times, either due to packet drops by ETHERMON, the bridge ignoring the request, or some other network glitch that would cause packets to be dropped before they reached the monitoring system. It is suggested that you repeat the MAP command to ensure that data is received from all the bridges.

18. `%ETHERMON-I-BRGTMO2, bridge #n timed out`

    While executing a MAP command with SET MESSAGES DEBUG, a response was not received from the specified bridge, either due to a packet drop by ETHERMON, the bridge ignored the request, or some other network glitch that would cause a packet to be dropped before they reached the monitoring system. It is suggested that you repeat the MAP command to ensure that data is received from all the bridges.

19. `%ETHERMON-E-CANTDOIEEE, this VAX can't support IEEE 802 mode`

    A command was entered that requires the Ethernet interface to be running in IEEE 802.3 extended mode, but ETHERMON was started in Ethernet mode instead because the interface or the VAX system, for some reason, could not operate in IEEE 802.3 mode. Find why the VAX cannot support IEEE 802.3 extended mode on the selected Ethernet interface (typically, this is because you are using a VAX running VMS V4.7 or earlier).

20. `%ETHERMON-W-CANTSTART, can't start Ethernet port, devchar=xxxxxxxx`

    Some sort of VMS I/O driver error occurred while trying to start the Ethernet interface for promiscuous monitoring. An accompanying system message will give additional details as to the cause of the error.

21. `%ETHERMON-I-CLSINDFIL, indirect command file closed due to control-C`

    A control-C was typed while an indirect command file was executing. Not only was the current command aborted, but the command file was closed and input redirected to the standard input device SYS$INPUT. No action is required.

22. `%ETHERMON-W-CMPETHRD, error completing Ethernet read`

    Some sort of VMS I/O driver error occurred during completion of an asynchronous read of the Ethernet interface. An accompanying system message will give additional details as to the cause of the error.

23. `%ETHERMON-W-CMPNETACP, error completing status query to DECnet NETACP`

    A request was sent to the DECnet NETACP process to translate a DECnet node number to a node name, and the request completed in error. An accompanying system message will give additional details as to the cause of the error. This message can be suppressed with the SET MESSAGES OFF command.

24. `%ETHERMON-F-CRASHDUMP, crash dump of ETHERMON`

    This prefixes a fatal error report from within the per-message read and analysis code in ETHERMON. The following lines will give context of the error. This information (other than the traceback) will also be recorded in a disk file named ETHERMON.CRASH_DUMP in the current directory. If this should happen, please contact the author and send him a copy of the crash dump file's contents (the file is ASCII and easily mailed); please include the version of ETHERMON you were running and the version of VAX/VMS you were running under.

25. %ETHERMON-W-DISPJAMMD, screen stalled for > 7 minutes; data munched

    Either the user accidentally pressed the NO SCROLL key (control-S) on his/her keyboard for too long during a DISPLAY command, or ETHERMON was over-saturated with network messages continuously for too long. The DISPLAY command is terminated. Re-enter the DISPLAY command and don't leave NO SCROLL active for such a long time period.

26. %ETHERMON-W-DUPIPADR, *aa-bb-cc-dd-ee-ff* has same IP adr *a.b.c.d* as *aa-bb-cc-dd-ee-ff*

    The SET MESSAGES DUPIP ON command has been entered, and ETHERMON detected a node claiming the same IP source address as an existing node (the first Ethernet address displayed in the message is the new node, and the second Ethernet address is the older entry in the address table). Determine why two different IP hosts are using the same IP address.

27. %ETHERMON-W-ERRGETSYI, error issuing $GETSYI service call

    During ETHERMON initialization, an attempt to query VAX/VMS for system identification or configuration data failed. Use the associated VAX/VMS error message to determine why the $GETSYI service call failed.

28. %ETHERMON-E-ERROPNIND, can't open specified indirect command file

    For some reason, ETHERMON cannot open the indirect command file specified with the last @*filespec* command.

29. %ETHERMON-E-FILOPNERR, FORTRAN I/O error #nn trying to open LOAD file

    ETHERMON could not open the specified file during a LOAD command. Look up the FORTRAN IOSTAT error message number and correct the corresponding problem.

30. %ETHERMON-I-FLTADDDIS, filter #nn added and disabled

    The ADD FILTER command successfully defined a new filter and disabled it. No action is necessary.

31. %ETHERMON-I-FLTADDENA, filter #nn added and enabled

    The ADD FILTER command successfully defined a new filter and enabled it. No action is necessary.

32. %ETHERMON-I-FLTDEL, filter #nn deleted

    The CLEAR FILTER command successfully deleted a filter definition. No action is necessary.

33. %ETHERMON-I-FLTMODDIS, filter #nn modified and disabled

    The SET FILTER command successfully modified a filter definition, leaving it in the disabled state. No action is necessary.

34. %ETHERMON-I-FLTMODENA, filter #nn modified and enabled

    The SET FILTER command successfully modified a filter definition, leaving it in the enabled state. No action is necessary.

35. %ETHERMON-W-FLTTBLFUL, filter table full - new filter not added

    An ADD FILTER or LOAD command failed because there was not enough space in the filter table. Remove some unused filter definitions with the CLEAR FILTER command and try the ADD FILTER or LOAD command again.

36. %ETHERMON-I-FNDINETGW, *aa-bb-cc-dd-ee-ff* is an Internet IP gateway

    Sufficient data has been received to determine that the specified node is an Internet IP gateway, or router. No action is necessary. This message can be suppressed with the SET MESSAGES IP OFF command.

37. `%ETHERMON-I-FNDMLTIPA,` *aa-bb-cc-dd-ee-ff* `has multiple Internet IP addresses`
    `first is` *a.b.c.d*`, second is` *e.f.g.h*

    During the monitoring, messages specifying more than one legal non-broadcast DoD Internet IP address was detected being sent to or received by a single Ethernet node; this may imply that the node is an Internet IP gateway (but not yet confirmed as such), or that the node is misconfigured. The first address displayed is the one the node originally appeared as, and the second is the second address listed. Note that third, fourth, etc., addresses will not be displayed; only enough addresses to determine the multi-address characteristics of the node will be reported. This message can be suppressed with the SET MESSAGES IP OFF command.

38. `%ETHERMON-I-HISYSGEN, SYSGEN parameters are high:`
    `DEFMBXBUFQUO=`*nnn* `PQL_MBYTLM=`*nnn* `PQL_DBYTLM=`*nnn*

    While initializing ETHERMON, the program happened to notice that the values for one or more of the SYSGEN parameters DEFMBXBUFQUO, PQL_MBYTLM, and PQL_DBYTLM seemed to be unusually large. This may or may not have an affect on the operation of ETHERMON; however, these parameters are used in the BYTLM quota calculations (documented under the LOWBYTLM error message), and may cause a following LOWBYTLM error to occur. SHould you get a LOWBYTLM message immediately after receiving the HISYSGEN message, check whether the SYSGEN parameters are supposed to be that large; if they are, you will need to increase the BYTLM quota for your user account to compensate.

39. `%ETHERMON-I-INCBRGDAT, incomplete data used for map - use MAP command again`

    While the MAP command was being executed, it discovered that incomplete infomation was stored for some of the bridges known to ETHERMON. The additional information was automatically filled in, but not necessarily in time for previous queries to other bridges to take advantage of the new information. The MAP command should be re-executed to re-run the data collection with the new information available.

40. `%ETHERMON-F-INCONSIST, internal inconsistency in command parser - submit PR`

    While parsing a command entered by the user, the parser disconvered an internal inconsistency in the parsing logic. If this message occurs, please submit a problem report to the author, giving the exact command line you entered (character for character). Also please include the version of ETHERMON you were running and the version of VAX/VMS you were running under.

41. `%ETHERMON-I-IMOPMSG,` *hh:mm:ss* *aa-bb-cc-dd-ee-ff* `sent iMOP` *prot* `msg, t=`*xxxx* `ln=`*nn*

    The specified node transmitted a DEC MOP (Maintenance Operations Protocol) message (using IEEE 802.3 SNAP formatting) of a significant type. *prot* specified whether the message was a Dump/Load Assistance (60-01) or Remote Console (60-02) protocol type, the "t=*xxxx*' specifies the first word of the message in hexadecimal (high byte first), and "ln=*nn*" gives the length of the message in bytes. This message can be suppressed with the SET MESSAGES MOP OFF command.

42. `%ETHERMON-W-INVDSPINT, invalid display interval of nn sec; not changed`

    The SET INTERVAL command specified an invalid or out-of-range number of seconds for the display update interval; the old setting was not changed. Re-enter the command with a valid number between 2 and 59.

43. `%ETHERMON-W-INVFLTNUM, specified filter is undefined or out of range`

    In a SET FILTER or CLEAR FILTER command, there is no filter corresponding to the specified filter index number. Re-enter the command with a valid filter index (use the SHOW FILTER command to check what filter numbers are defined).

44. `%ETHERMON-E-INVNODSPC, invalid network node specification`

    An ADD FILTER or SET FILTER command specified an Ethernet address symbolically by DECnet node number, DECnet node name, or Internet IP address, but there was a syntax error in the specification. Re-enter the command with a valid-syntax address.

45. `%ETHERMON-W-INVTRCTYP, only 1 file format qualifier may be given on TRACE`

    A TRACE command was entered with more than one of the qualifiers /ASCII, /BINARY, /SNIFFER, or /ANALYZER. Only one of these qualifiers may be used at a time. Re-enter the TRACE command with only one of the file format qualifiers.

46. `%ETHERMON-I-LOWBYTLM, current BYTLM=`*nnnn* `lower than recommended minimum` *mmmm*

    This message will appear during ETHERMON start-up if your process does not have sufficient BYTLM quota left to allow ETHERMON to execute. It may appear as an informational message (ETHERMON will run anyway, with possible degraded performance) or as a fatal error (ETHERMON cannot run and will terminate). Get your BYTLM quota raised, or don't run as many other applications that consume BYTLM quota from your process. Note that the recommended minimum for efficient operation is at least 10000 bytes higher than the absolute minimum required for the program to run at all. Note that the recommended minimum may seem rather large if other layered products have modified the SYSGEN parameters DEFMBXBUFQUO, PQL_MBYTLM, or PQL_DBYTLM. The absolute minimum is determined by the formula

    ```
    value = MAX(PQL_MBYTLM,PQL_DBYTLM)
            + DEFMBXBUFQUO
            + buffer_size_of_port + 50
            + 1100
            + buffer_size_of_port + 50
            + 4 * (buffer_size_of_port + 50)
            + 4 * 512
            + 252
    ```

    where the SYSGEN parameters are used to reserve enough quota for the SPAWN command to work, the first *buffer_size_of_port*+50 is for the normal $QIO read buffer needed by ETHERMON, the 1100 is for other I/O buffers (terminal I/O, etc.) needed by ETHERMON, and the second *buffer_size_of_port*+50 is to account for at least one pre-allocated receive buffer for the Ethernet I/O driver (the NMA$C_PCLI_BFN extended characteristic), the 4*(*buffer_size_of_port*+50) is for two maximum-size messages in a mailbox for the TRACE/ANALYZER command and the write $QIO by ETHERMON and the read by the subprocess, the 4*512 is for standard output and standard error output by the TRACE/ANALYZER command, and the 252 is for TRACE/ANALYZER termination handling; for each additional *buffer_size_of_port*+50 bytes available above this minimum, the NMA$C_PCLI_BFN value used is incremented by 1, to a maximum value of 127. The value of *buffer_size_of_port* is 1492 for Ethernet devices, and 4468 for FDDI devices (the NMA$C_PCLI_MBS extended characteristic). Note also that these calculations are performed after some BYTLM quota has already been consumed to create the cloned device units for the Ethernet/FDDI interface and the DECnet network mailbox.

47. `%ETHERMON-W-MDUPIP,` *aa-bb-cc-dd-ee-ff* `(`*name*`) claims` *aa-bb-cc-dd-ee-ff* `has`
    `same IP adr` *a.b.c.d* `as` *aa-bb-cc-dd-ee-ff*

    The SET MESSAGES DUPIP ON command has been entered, and ETHERMON detected a node claiming the same IP source address as an existing node (the first Ethernet address displayed in the message is the new node, and the second Ethernet address is the older entry in the address table). Determine why two different IP hosts are using the same IP address.

48. `%ETHERMON-W-MNFTBLFUL, manufacturer table full - new code/name pair not added`

    A SET MANUFACTURER command attempted to add a new Ethernet manufacturer prefix to the manufacturer table, but there was no more room for it. Increase the value of the manufacturer name table size parameter ICPMNF_MAXNAMES in ETHERMON_MANUF.INC, then rebuild the ETHERMON program.

49. `%ETHERMON-F-MNFTBLOVF, Ethernet manufacturer table overflow - PROGRAM ABORT`

    The manufacturer prefix table did not have enough free space to store all the different manufacturer prefixes seen on your Ethernet. Increase the value of the manufacturer prefix table size parameter ICPMNF_MAXNAMES in ETHERMON_MANUF.INC, then rebuild the ETHERMON program. However, this also implies that you have an exceedingly busy Ethernet running far too many unrecognized manufacturer prefix codes, which you may wish to correct. Use the SHOW MANUFACTURERS command to dump the contents of the manufacturer prefix table after it has filled up to maximum while SET MESSAGES ON or SET MESSAGES OFF (not SET MESSAGES DEBUG) is in effect.

Note that this message will only appear while SET MESSAGES DEBUG is in effect (which can happen by default if ETHERMON_CMD_S.FOR is compiled with the /D_LINES option).

50. `%ETHERMON-I-MOPMSG,` *hh:mm:ss* *aa-bb-cc-dd-ee-ff* `sent MOP` *prot* `msg, t=`*xxxx* `ln=`*nn*

    The specified node transmitted a DEC MOP (Maintenance Operations Protocol) message (using Ethernet formatting) of a significant type. *prot* specified whether the message was a Dump/Load Assistance (60-01) or Remote Console (60-02) protocol type, the "t=*xxxx*' specifies the first word of the message in hexadecimal (high byte first), and "ln=*nn*" gives the length of the message in bytes. This message can be suppressed with the SET MESSAGES MOP OFF command.

51. `%ETHERMON-W-MSGLEN, specified message buffer length out of range`

    A SET FILTER/SIZE command was entered with the MAXIMUM or MINIMUM option, but the numeric size was outside the range of 46 to 4470. Re-enter the command with a valid message size.

52. `%ETHERMON-W-NETACPDAT, insufficient data returned by NETACP`

    A query was sent to the DECnet NETACP process, but the returned data was too short to be intelligible. Contact your local DEC software support center, as this implies a weird fault within DECnet; however, please notify the author as well.

53. `%ETHERMON-I-NEWNODE, new Ethernet sender` *aa-bb-cc-dd-ee-ff* `detected`

    The SET MESSAGES NEWNODE ON command has been entered, and a new sending node has just been added to the address table. No action is required (except possibly to enter the SET MESSAGES NEWNODE OFF command if you do not wish to see these messages).

54. `%ETHERMON-E-NOASYNTRC, no asynchronous TRACE active to abort - command ignored`

    The TRACE/ABORT command was entered when there was no active TRACE/PARALLEL or TRACE/REPEAT background command in progress. No action is required (except possibly to determine why there was no active TRACE command when you expected one to be active).

55. `%ETHERMON-I-NODECNET, can't access DECnet services`

    At ETHERMON start-up, a connection to a DECnet network mailbox could not be made, preventing ETHERMON from translating DECnet addresses with the help of NETACP. DECnet may not be running on your system, or you may not have NETMBX (or SETPRV) privilege. If DECnet is supposed to be on your system but isn't, contact your system manager to find why it is not currently running.

56. `%ETHERMON-W-NOFASTFB, can't activate ETHERMON_FASTSHR.EXE; using unsupported interface`

    The "fast" $QIO interface for ETHERMON could not be started. However, you are using a sufficiently old version of VAX/VMS for the old "plain" $QIO interface to still function. Unless you wish to use the "fast" interface, no action is required; if you do wish to use the "fast" interface, see the actions described for the %ETHERMON-F-NOFASTSHR message.

57. `%ETHERMON-F-NOFASTSHR, can't activate ETHERMON_FASTSHR.EXE - PROGRAM ABORT`

    The "fast" $QIO interface for ETHERMON could not be started for some reason, and, because you are using a version of VAX/VMS later than V5.4-2, the old-style $QIO interface could not be used successfully. A VMS system error message should accompany this message, explaining why the "fast" interface could not be activated. The expected VMS error message (and corresponding action) should be one of the following:

    - `-RMS-F-FNF`

      The shareable image file ETHERMON_FASTSHR.EXE could not be found, either by executive-mode logical name translation, or by looking in the directory where the ETHERMON.EXE file being executed was located. Define the logical name or move the file to the same directory as the ETHERMON executable image.

    - `-SYSTEM-F-SHRIDMISMAT`
      `-LIB-F-KEYNOTFOU`

      The installed shareable image file ETHERMON_FASTSHR.EXE is of an incompatible patch level for the current version of ETHERMON. Re-compile and re-link both ETHERMON and ETHERMON_FASTSHR under the current operating system with current source files, then re-install ETHERMON_FASTSHR.

    - `-SYSTEM-F-EXQUOTA`

      You do not have enough BYTLM quota left to create the non-paged pool data structures needed to support the "fast" interface. This error should never occur (ETHERMON should have reserved enough BYTLM when it saved quota for spawning subprocesses); if it does, please submit a problem report to the author, giving the exact command line you entered (character for character). Also please include the version of ETHERMON you were running and the version of VAX/VMS you were running under.

    - `-SYSTEM-F-INSFMEM`

      Your system does not have enough non-paged pool left to allocate space for the data structures needed by the "fast" interface. Exit ETHERMON and re-SYSGEN your system (if you get this error, your system is far too tight on non-paged pool to consider running ETHERMON in any mode).

- `-SYSTEM-F-LKWSETFUL`

  Your process's working set is too small to be able to lock down the additional memory pages required for the "fast" interface to run successfully. In this case, you definitely do not have a large enough working set for ETHERMON to be able to run adequately. Enlarge your working set limit and/or quota (or have your system administrator grant your account more quota), then try again.

- `-SYSTEM-F-NOPRIV`
  `-SYSTEM-F-NOTALLPRIV`

  The user has all the privileges necessary to run ETHERMON in the old mode, but does not have the CMKRNL (or SETPRV) privilege necessary to use the "fast" mode interface. Have your system administrator grant your account the necessary privileges.

- `-SYSTEM-F-PROTINSTALL`

  The shareable image file ETHERMON_FASTSHR.EXE was not installed as a protected shareable image with the INSTALL utility. Install the image as described in section 3.1.

- `-SYSTEM-F-SYSVERDIF`

  The shareable image file ETHERMON_FASTSHR.EXE was compiled and linked under an incompatible version of VAX/VMS. Re-compile and re-link ETHERMON_FASTSHR under the current operating system, then re-install it.

- `-SYSTEM-F-VECFULL`

  For some strange reason, your account environment has so many privileged images using the kernel-mode user-rundown facility that there are no vector entries left in the rundown vector table. If this message occurs, please submit a problem report to the author, giving the exact command line you entered (character for character). Also please include the version of ETHERMON you were running, the version of VAX/VMS you were running under, and the names and descriptions of any privileged layered products that are installed on your system.

58. `%ETHERMON-E-NOFILE, can't find specified LOAD file`

    The file specified on a LOAD command cannot be found. Re-enter the LOAD command with the correct file specification (watch the defaults).

59. `%ETHERMON-W-NOGBLSEC, couldn't create address table global section`

    During start-up of ETHERMON, an attempt was made to create a temporary group global section containing the address table, such that the table might be shared with TRACE/ANALYZER subprocesses. However, the section creation failed. An accompanying system error message from the $CRMPSC system service should explain why the section could not be created. ETHERMON will function without the section; however, address table data cannot be shared in real-time with subprocesses unless the section is created. As of ETHERMON V2.4, creating the global section requires 2009 contiguous free global pages, and (preferably) the same number of free global page-file pages; these pages will be released when ETHERMON is exited. If the contiguous global pages are available, but the page file space is not, ETHERMON will attempt to create its own paging file (named SYS$SCRATCH:ETH_ADDRTBL.GBLSEC); this file will be automatically deleted when ETHERMON exits.

60. `%ETHERMON-I-NOINET, can't find known Internet software to query`

    Upon receiving a LOAD/INTERNET command, ETHERMON could not find a recognized type of Internet software to query for name information. The LOAD/INTERNET command only works with the products DEC VMS/Ultrix Connection (UCX, now called TCP/IP Services for VMS), TGV MultiNet, NRC Fusion, and CMU TCP/IP; if you have no Internet software, or something other than these, the command probably will not work.

Should you have another manufacturer's Internet software and know how to query it to translate Internet physical addresses to *host.network* name format, please contact the author so I can add support for your configuration to ETHERMON.

61. `%ETHERMON-E-NOINETDST, specified destination hasn't spoken Internet yet`

    A PING/INTERNET command was issued to a node which does not have an IP address assigned to it yet. Verify that the correct Ethernet address was given on the PING command, or determine why the specified node has not yet been seen to speak DoD Internet IP protocol (08-00).

62. `%ETHERMON-I-NONSTDMOP,` *aa-bb-cc-dd-ee-ff* `had non-std MOP type` *xxxx*`, len=`*nn*

    A DEC MOP (Maintenance Operations Protocol) message with an undocumented message type was sent by the specified node. You may wish to investigate why the non-standard message packet was sent. This message will only be displayed with the SET MESSAGES DEBUG command.

63. `%ETHERMON-I-ONEPORT, implicitly using the one Ethernet interface` *_ddc0:*

    ETHERMON was invoked with a RUN command, or as a foreign command but without an Ethernet device name, and it has searched for the expected types of Ethernet devices on the system, and found only one such device, which it automatically selected for use. No action is required (unless you wanted to use some non-standard third-party interface; in that case, you must specify its device name on the foreign command line).

64. `%ETHERMON-I-PNGNOTSENT, PING message was not transmitted during wait interval`

    For some reason, during the period that ETHERMON waits for a PING to be acknowledged, the Ethernet device driver did not even finish sending the PING request. Please notify the author if this message appears, and describe what type of Ethernet interface hardware and device driver software you were using. Note that this message was reported falsely in versions V2.2 to V2.3-1 of ETHERMON when PINGing to the Broadcast address; the error has been corrected as of V2.3-2.

65. `%ETHERMON-I-PORT, enter VMS device name of the Ethernet interface to use`

    ETHERMON was invoked with a RUN command, or as a foreign command but without an Ethernet device name, and it has searched for the expected types of Ethernet devices on the system, and found more than one such device. Following this message will be a list of VMS device names and a prompt to enter a device name. Enter one of the displayed device names to continue execution.

66. `%ETHERMON-F-PORTASNFAI, error assigning channel to Ethernet interface`

    During ETHERMON initialization, an I/O channel could not be assigned to the specified or defaulted Ethernet interface device. Check that your account is permitted access to the Ethernet device, and verify that the device name was typed in properly and any logical name used translates properly to a valid Ethernet interface device name. The associated VMS error message will help describe why the $ASSIGN system service failed.

67. `%ETHERMON-F-PROTBLOVF, Ethernet protocol table overflow - PROGRAM ABORT`

    The protocol type table did not have enough free space to store all the different protocol types seen on your Ethernet. Increase the value of the protocol name table size parameter ICPPRT_MAXNAMES in ETHERMON_PROT.INC, then rebuild the ETHERMON program. However, this also implies that you have an exceedingly busy Ethernet running far too many unrecognized protocol types, which you may wish to correct. Use the SHOW PROTOCOL_NAMES command to dump the contents of the protocol table after it has filled up to maximum while SET MESSAGES ON or SET MESSAGES OFF (not SET MESSAGES DEBUG) is in effect.

Note that this message will only appear while SET MESSAGES DEBUG is in effect (which can happen by default if ETHERMON_CMD_S.FOR is compiled with the /D_LINES option).

68. `%ETHERMON-I-PROXYARP,` *aa-bb-cc-dd-ee-ff* `is proxy-ARPing for` *aa-bb-cc-dd-ee-ff*

    A node on the Ethernet was detected sending an Internet ARP message for another node (not for itself). You may wish to verify that proxy-ARP is supposed to be enabled for the selected node. This message can be suppressed with the SET MESSAGES ARP OFF command.

69. `%ETHERMON-W-PRTTBLFUL, protocol table full - new protocol/name pair not added`

    A SET PROTOCOL_NAME or LOAD command was entered, but there was no room in the protocol name table to add another new entry. Increase the value of the protocol name table size parameter ICPPRT_MAXNAMES in ETHERMON_PROT.INC, then rebuild the ETHERMON program.

70. `%ETHERMON-W-QIONETACP, error queuing status query to DECnet NETACP`

    An attempt to send a query to the DECnet NETACP process to translate a DECnet node number was made, but the query could not be queued. An accompanying system message will give more details about the error. This message can be suppressed with the SET MESSAGES OFF command.

71. `%ETHERMON-E-SATURATED, port counters at limits; do SHOW PORT_ST/ZERO & try again`

    A DISPLAY or SUMMARIZE command was entered; however, the Ethernet interface port counters that were needed to compute some of the display data were saturated and need to be reset. The command was not processed (allowing the user to decide whether or not the counters should be reset). If you decide that the old counter values do not need to be preserved, you can enter the SHOW PORT_STATUS/ZERO command to zero the counters, then re-try the DISPLAY or SUMMARIZE command.

72. `%ETHERMON-E-STRTANLZ, couldn’t start TRACE/ANALYZER command`

    A TRACE/ANALYZER command was entered; however, for some system reason, the environment for the analyzer subprocess could not be created. The command was not processed. An accompanying system error message should explain why the command could not be processed (usually indicating some system resource or quota limitation).

73. `%ETHERMON-W-TOOMANYADR, too many addresses for QUERY command; command ignored`

    The QUERY command was issued with a range of addresses that was too large to practically deal with, so the command was ignored. Use a smaller range of addresses if you are going to use the QUERY command (less than one whole manufacturer’s prefix worth is the absolute limit).

74. `%ETHERMON-E-TOOMANYIND, can't nest indirect command files`

    An indirect command file was invoked, which in turn attempted to invoke another file. Only one level of indirection is permitted; remove the indirect file reference from the first indirect file.

75. `%ETHERMON-I-TRACERPT, post-filter: nn bytes, nn messages recorded to file`

    The specified number of messages and bytes of data were recorded in a TRACE command's output file. No action is necessary.

76. `%ETHERMON-I-TRACERPT2, pre-filter: nn bytes, nn messages received on net`

    The specified number of messages and bytes of data were received from the Ethernet during a TRACE command. No action is necessary.

77. `%ETHERMON-I-TRACERPT3, pre-ETHERMON: nn messages lost before analysis`

    The specified number of network packets were dropped by the Ethernet interface device driver during a TRACE command, because ETHERMON couldn't keep up with the data rate. No action is necessary (except possibly to use a more powerful VAX in future network monitoring).

78. `%ETHERMON-I-TRACERPTA, pre-analyzer: nn messages lost due to mailbox full`

    The specified number of network packets were dropped by ETHERMON during a TRACE command, because the analyzer module subprocess couldn't keep up with the data rate and let its input mailbox overflow. No action is necessary (except possibly to use a more powerful VAX in future network monitoring, and/or to tune the execution of the analyzer module).

79. `%ETHERMON-E-TRCBSY, couldn't start TRACE command; previous TRACE still busy`

    A TRACE command was entered; however, a previous asynchronous TRACE command is still running. Abort the running TRACE with TRACE/ABORT, or wait for it to finish executing. If there is no active TRACE running, please send a problem report to the author, and describe your system's VAX/VMS version and ETHERMON version.

80. `%ETHERMON-I-UNDDECNOD, node nn.nn seen on Ethernet; not in DECnet database`

    ETHERMON requested the local DECnet NETACP to translate a DECnet node number to a node name, but NETACP did not have a corresponding name in its database. Check with your system or network manager as to why the specified DECnet node is appearing on your Ethernet but is not listed in the local system's DECnet node database. This message will only be displayed if you have entered the SET MESSAGES DEBUG command.

81. `%ETHERMON-I-UNKARPOPC,` *aa-bb-cc-dd-ee-ff* `sent unknown ARP opcode: prot=`*pp-pp* `opc=`*pp-pp*

    An Internet ARP or RARP message was received, but the ar$op field (see NIC.DDN.MIL Request For Comments RFC826.TXT) for details of ARP message formats) was not one of the values 01-00, 02-00, 03-00, 04-00 (the legal values for ARP and RARP), or 05-00, 06-00, or 07-00 (dynamic RARP). You may wish to determine if there is a node transmitting erroneous packets on your Ethernet; if not, please inform the author of your discoveries so that a future version of ETHERMON can properly deal with the extensions to the ARP protocol. This message can be suppressed with the SET MESSAGES ARP OFF command.

82. `%ETHERMON-I-UNKETHCHR, unknown Ethernet port characteristic code=xxxx, xxxx`

    An undeciphered port characteristics code was received while performing an IO$_SENSEMODE operation on the interface. This message can only appear if you have compiled ETHERMON.FOR with the /D_LINES option on the FORTRAN compiler. Check for compatability of your Ethernet interface's I/O driver with the standard DEC driver, as documented in the *VMS I/O User's Reference Manual: Part II*.

83. `%ETHERMON-I-UNKETHCNT, unknown Ethernet port counter code=xxxx, xxxx`

    An undeciphered port counter ID code was received while performing an IO$_SENSEMODE with IO$M_RD_COUNT operation on the interface. This message can only appear if you have compiled ETHERMON.FOR with the /D_LINES option on the FORTRAN compiler. Check for compatability of your Ethernet interface's I/O driver with the standard DEC driver, as documented in the *VMS I/O User's Reference Manual: Part II*.

84. `%ETHERMON-I-UNKIPARP,` *aa-bb-cc-dd-ee-ff* `sent unknown ARP pkt: prot=`*pp-pp* `Hlen=`*n* `Plen=`*n*

    An Internet ARP or RARP message was received, but the values of the arhrd, arhln, and/or ar$pln fields (see NIC.DDN.MIL Request For Comments RFC:RFC826.TXT) for details of ARP message formats) were not legal for IP and/or Ethernet addresses. You may wish to determine if there is a node transmitting erroneous packets on your Ethernet; if not, please inform the author of your discoveries so that a future version of ETHERMON can properly deal with the extensions to the ARP protocol. This message can be suppressed with the SET MESSAGES ARP OFF command.

I.2. System Messages Frequently Seen

1. `%SMG-E-KEYDEFPRO, key definition is protected - cannot be changed or deleted`

    You attempted to re-define a key (using the DEFINE/KEY command) or delete a key definition (using DELETE/KEY) for a key with a permanently defined function. The attempt has failed, and the key definition has not changed. Use a different key for your local definition.

2. `%SYSTEM-F-BADPARAM, bad parameter`

    This error is caused by one of two reasons:

    a. There is already a promiscuous user on the Ethernet interface you are attempting to use. Try another interface, or get rid of the existing promiscuous user (it could be another invocation of this program).

    b. You are attempting to run under Version V4.x of VAX/VMS, which does not recognize IEEE 802.3 mode operation of the Ethernet interface. ETHERMON can be retrofitted to run in Ethernet mode; however, all IEEE 802.3 extended messages seen on the cable will be discarded (data that would appear as Ethernet protocol type AA-AA). To make the change, uncomment all lines in ETHERMON.FOR that begin with the comment string "C4.x", and recompile and relink ETHERMON. You may also need to comment out the lines ending in "! V5.x". Note that these edits should not need to be done; ETHERMON should work under either V4.x or V5,x after re-compiling and re-linking. Note also that, if you are using the "fast" $QIO interface, you should be able to successfully interpret IEEE 802.3 messages even if you are using VAX/VMS V4.x. This configuration, although theoretically workable, is not supported.

3. `%SYSTEM-F-EXBYTLM, exceeded byte count quota`

   Under some installations of VAX/VMS with some homebrew and/or third-party software installed, sometimes the SPAWN command will fail with this error. You will also find on these systems that VAX DEBUG cannot run in subprocess mode. The reason for this is improper values of the SYSGEN parameters controlling default mailbox parameters (DEFMBXBUFQUO and MAXBUF); MAXBUF should always be larger than DEFMBXBUFQUO, and the user BYTLM quota must be at least 10000 bytes larger than DEFMBXBUFQUO for ETHERMON to operate (assuming the DEC default value for PQL_MBYTLM; preferably, your user BYTLM quota should be considerably larger).

4. ```
   %RMS-F-EXT, ACP file extend failed
       -SYSTEM-F-EXDISKQUOTA, disk quota exceeded
   ```

   While writing an output disk file (a SHOW, SUMMARIZE, or TRACE command, typically), your process exceeded your allowed disk quota. Free up some disk space and try again.

5. `%SYSTEM-F-EXQUOTA, exceeded quota`

   Typically, this will occur when issuing a SPAWN or TRACE/ANALYZER command, because the user account will not have enough BYTLM or PRCLM quota (although other quotas may be exceeded) left to create the subprocess. Get your quotas raised by your system administrator or reduce the number of active subprocesses you are running.

6. `%SYSTEM-F-NOSUCHNODE, remote node is unknown`

   A symbolic node address was specified on an ADD FILTER or SET FILTER command, but the specified DECnet node name or Internet IP address could not be found in the address name table. Either specify the address correctly, or add the address specification to the address table. This message may appear when using the ADD or SET FILTER commands immediately after starting ETHERMON, before ETHERMON has seen the node speak the corresponding protocol; in that case, just wait a little while until the remote node speaks the appropriate protocol (in batch jobs, use the WAIT command to stall a few seconds; an example is given on page 49).

7. %HELP-E-OPENIN, error opening *disk:[dir]*ETHERMON.HLB; as input
       -RMS-E-FNF, file not found

   Upon issuing the HELP command, the help library file ETHERMON.HLB could not be found. This is generally due to placing the .HLB file in a different directory from the ETHERMON.EXE file; all ETHERMON run-time files expect to be in the same directory (location of source files is irrelevant, except that they all must be in the current default directory at build time). Either move the help library file to the directory containing the .EXE file, or edit appropriately the subroutine HELP in the file ETHERMON.FOR and rebuild ETHERMON.
   ```

Appendix II. Known Ethernet Data

There are a great deal of numbers associated with the Ethernet. Some of these numbers are in the DEC/Xerox/Intel and/or IEEE specifications. Some of them were assigned by various authoritative bodies. Some of them were just claimed by squatter's rights (thankfully very few of these). The information used to pre-load ETHERMON's protocol type and manufacturer prefix tables, and, in general, analyze the data packets (both in ETHERMON and the unsupported utilities), came from many sources:

- *VMS I/O User's Reference Manual: Part II*, Version 5.0, Digital Equipment Corporation, April 1988, order number AA-LA85A-TE, in the chapter on Ethernet interfaces (previous editions also used)
- *VMS Network Control Program Manual*, Version 5.0, Digital Equipment Corporation, April 1988, order number AA-LA50A-TE, in the sections on the SHOW LINE command and on line counters (previous editions also used)
- *VMS System Services Reference Manual: Part II*, Version 5.0, Digital Equipment Corporation, April 1988, order number AA-LA69A-TE (previous editions also used)
- *DECnet Digital Network Architecture Phase IV: Maintenance Operations Functional Specification*, MOP Version 3.0.0, Digital Equipment Corporation, December 1983, order number AA-X436A-TK.
- DoD Internet Requests For Comments (RFC's), as obtained from the Network Information Center (the NIC, with Internet address NIC.DDN.MIL) via anonymous FTP:
    1. RFC 768, User Datagram Protocol, Postel, J.B., August 1980
    2. RFC 791, Internet Protocol, Postel, J.B., September 1981
    3. RFC 792, Internet Control Message Protocol, Postel, J.B., September 1981
    4. RFC 793, Transmission Control Protocol, Postel, J.B., September 1981
    5. RFC 826, Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware, Plummer, D.C., November 1982
    6. RFC 1057, RPC: Remote Procedure Call Protocol specification: Version 2, Sun Microsystems, Inc., June 1988
    7. RFC 1058, Routing Information Protocol, Hedrick, C.L., June 1988
    8. RFC 1060, Assigned Numbers, Reynolds, J.K.; Postel, J.B., March 1990 (the now-obsoleted RFC 1010, May 1987, was also used in earlier versions of ETHERMON)
    9. RFC 1094, NFS: Network File System Protocol specification, Sun Microsystems, Inc., March 1989
    10. RFC 1157, Simple Network Management Protocol (SNMP), Case, J.D.; Fedor, M.; Schoffstall, M.L.; Davin, C., May 1990
- *Inside AppleTalk*, 2nd ed., Sidhu, G., Andrews, R.F., and Oppenheimer, A.B., 1990.
- other obscure vendor documents whose titles have been lost in the mists of time
- sheer trial and error by many parties ("Ah hah! Another unknown manufacturer prefix... but it speaks IP. What happens if I Telnet to it? Ahhh... an operating system signon message, so now I know what it is {**maybe...**}.") Thanks to all those users in the field who told me about new values they spotted, so I could add them to this program's knowledge base.

As none of these sources is either complete or infallible, the data known by ETHERMON is necessarily incomplete and prone to errors. All the information the author was able to obtain is embedded in this program (with the exception of non-SNAP IEEE 802.3 data; that is not yet properly supported by ETHERMON). If you find any errors or determine any new manufacturer prefixes or protocol types, please let the author know, so that the information can be shared with all in a new release of ETHERMON. The author's address (electronic and real-world) is given in section 1.1 on page 4.

II.1. Manufacturer Prefixes

These are the manufacturer prefixes currently known to ETHERMON, as displayed by the SHOW MANUFACTURERS/ALL command. Note that this only shows the physical prefix assignments. As far as the author knows, the corresponding multicast address blocks (same prefix except the LSB of the first byte is set) are also assigned to the same party. Generally, the name displayed is of the Ethernet *interface* manufacturer, which is not necessarily the same as the whole computer's manufacturer; in cases where two names are displayed, separated by a slash "/", this generally implies that either two manufacturers were merged on the corporate level, or that the second manufacturer is (apparently) the exclusive user of the first manufacturer's Ethernet boards in the specified address block. None of this, however, is confirmed.

```
00-00-0C Cisco[Cisco]
00-00-0F NeXT[NeXT]
00-00-10 Sytek[Sytek]
00-00-18 Multigate[Multigat]
00-00-1B Novell[Novell]
00-00-1D Cabletron[Cabltron]
00-00-20 Data Industrier AB[DIAB]
00-00-22 Visual Technology[VisTec]
00-00-2A TRW[TRW]
00-00-3D AT&T[AT&T]
00-00-44 Castelle[Castelle]
00-00-55 AT&T[AT&T]
00-00-5A S & Koch[S&K]
00-00-5D RGE[RGE]
00-00-5E I-net Asnd.Num.Authority[IAN
00-00-62 Honeywell[Honeywel]
00-00-65 Network General[N-G]
00-00-6B MIPS[MIPS]
00-00-77 MIPS[MIPS]
00-00-7A Ardent[Ardent]
00-00-81 Synoptics[Synoptic]
00-00-86 Gateway Communications[Gatew
00-00-89 Cayman Systems[Cayman]
00-00-8E Solburne (Sun clone)[Solburn
00-00-93 Proteon 802.3/802.4[Proteon]
00-00-94 Asante (Macintosh)[Asante]
00-00-9F Ameristar Technology[Amersta
00-00-A0 Sanyo Electronics[Sanyo]
00-00-A2 Wellfleet[Wellflt]
00-00-A3 Network Appl. Technology[N-A
00-00-A4 Acorn[Acorn]
00-00-A6 Network General[N-G]
00-00-A7 NCD[NCD]
00-00-A9 Network System Corp.[N-S]
00-00-AA Xerox[Xerox]
00-00-AB Logic Modeling Systems[LMS]
00-DD-07 Ungermann-Bass[U-B]
00-DD-08 Ungermann-Bass[U-B]
00-DD-09 Ungermann-Bass[U-B]
00-DD-0A Ungermann-Bass[U-B]
00-DD-0B Ungermann-Bass[U-B]
00-DD-0C Ungermann-Bass[U-B]
00-DD-0D Ungermann-Bass[U-B]
00-DD-0E Ungermann-Bass[U-B]
00-DD-0F Ungermann-Bass[U-B]
02-04-06 BBN internal[BBN]
02-07-01 MICOM/InterLAN[InterLAN]
02-60-86 Satelcom MegaPac (UK)[Satelco
02-60-8C 3Com (IBM PC)[3Com]
02-CF-1F CMC[CMC]
08-00-02 3Com/Bridge[Bridge]
08-00-03 Adv. Computer Communic.[ACC]
08-00-05 Symbolics LISP machine[Symbol
08-00-07 Apple Computer[Apple]
08-00-08 BBN[BBN]
08-00-09 Hewlett-Packard[H-P]
08-00-0A Nestar Systems[Nestar]
08-00-0B Unisys[Unisys]
08-00-0D ICL[ICL]
08-00-0E NCR[NCR]
08-00-10 AT&T[AT&T]
08-00-11 Tektronix, Inc.[Tektr]
08-00-14 Excelan[Excelan]
08-00-17 Network System Corp.[NSC]
08-00-1A Data General[DG]
08-00-1B Data General[DG]
08-00-1E Apollo[Apollo]
08-00-1F Sharp[Sharp]
08-00-20 Sun Microsystems[Sun]
08-00-22 NBI[NBI]
08-00-23 Matsushita Denso[Matsushi]
08-00-25 CDC[CDC]
```

```
00-00-B0 RAD Network Devices Ltd.[RND
00-00-B3 CIMLinc[CIMLinc]
00-00-B5 Vista[Vista]
00-00-B7 Dove[Dove]
00-00-BB Tri-Data Systems, Inc.[TriDa
00-00-BC Allen-Bradley[A-B]
00-00-C0 Western Digital[W-D]
00-00-C3 Harris[Harris]
00-00-C6 Hewlett-Packard INO(Eon)[HP
00-00-C8 Altos[Altos]
00-00-C9 Emulex (term.servers)[Emulex
00-00-D1 Adaptec Inc. ("Nodem")[Adapt
00-00-D3 Wang[Wang]
00-00-D7 Dartmouth College[Dartmth]
00-00-D8 3Com/Novell ?? PS/2[? PS/2]
00-00-DD Gould[Gould]
00-00-DE Unigraph[Unigraph]
00-00-E2 Acer Counterpoint[Acer C]
00-00-E8 Acoton Technology Corp.[Acot
00-00-EF Alantec[Alantec]
00-00-F0 Samsung[Samsung]
00-00-FD High Level Hardware[HLH]
00-01-02 BBN internal[BBN]
00-01-1B Network Peripherals Inc.[NPI
00-01-68 Wandel & Goltermann[W&G]
00-01-C8 Thomas Conrad Corp[ThoConrd]
00-17-00 Kabel[Kabel]
00-40-27 Sigma Network Systems[Sigma]
00-40-C8 Milan Technology Corp[Milan]
00-60-8C 3Com (IBM PC)[3Com]
00-80-00 IEEE[IEEE]
00-80-10 Commodore[Commodor]
00-80-19 Dayna Etherprint[Dayna]
00-80-2D Xylogics, Inc.[Xylogic]
00-80-34 SMT-Goupil[SMT-G]
00-80-52 TE Concepts[TE]
00-80-87 Okidata[Okidata]
00-80-8C Frontier Software Dvlpmt[Fro
00-80-A1 Microtest[Microtst]
00-80-A3 Lantronix[Lantrnx]
00-80-A9 Clearpoint[Clearpnt]
00-80-B2 Network Equipment Tech.[NET]
00-80-C7 Xircom[Xircom]
00-80-D3 Shiva[Shiva]
00-80-D6 NuvoLinkSC[NuvoLink]
00-AA-00 Intel[Intel]
00-DD-00 Ungermann-Bass[U-B]
00-DD-01 Ungermann-Bass[U-B]
00-DD-02 Ungermann-Bass[U-B]
00-DD-03 Ungermann-Bass[U-B]
00-DD-04 Ungermann-Bass[U-B]
00-DD-05 Ungermann-Bass[U-B]
00-DD-06 Ungermann-Bass[U-B]
08-00-26 Norsk Data[Norsk]
08-00-27 PCS Computer System GmbH[PCS]
08-00-28 Texas Instruments[TI]
08-00-2B DEC[DEC]
08-00-2E Metaphor[Metaphor]
08-00-2F Prime Computer[Prime]
08-00-36 Intergraph[Intrgrph]
08-00-37 Fujitsu-Xerox[Fujitsu]
08-00-38 Bull[Bull]
08-00-39 Spider Systems[Spider]
08-00-3E Motorola ?[Mot?]
08-00-41 DCA Digital Comm. Assoc.[DCA]
08-00-45 ???? (Xylogics?)[????]
08-00-46 Sony[Sony]
08-00-47 Sequential[Sequent]
08-00-49 Univation[Univatn]
08-00-4C Encore[Encore]
08-00-4E BICC[BICC]
08-00-51 Experdata[Experdat]
08-00-56 Stanford University[Stanford]
08-00-58 ??? (DECsystem-20)[?DEC20]
08-00-5A IBM[IBM]
08-00-67 Comdesign[Comdesgn]
08-00-68 Ridge[Ridge]
08-00-69 Silicon Graphics/Iris[Iris]
08-00-6A AT&T[AT&T]
08-00-6E Excelan/Concurrent[Cncrrnt]
08-00-70 Mitsubishi[Mitsubis]
08-00-74 Casio[Casio]
08-00-75 Danish Data ElektronikAS[DDE]
08-00-77 Retix Inc.[Retix]
08-00-7C Vitalink bridge[Vitalink]
08-00-80 XIOS[XIOS]
08-00-81 Cross Field Electronics[Cross
08-00-83 Seiko Denshi[Seiko]
08-00-86 Imagen/QMS[I/QMS]
08-00-87 Xyplex[Xyplex]
08-00-89 Kinetics[Kinetics]
08-00-8B Pyramid[Pyramid]
08-00-8D XYVision[XYVision]
08-00-8E Jupiter[Jupiter]
08-00-90 Retix Inc.[Retix]
10-00-5A IBM[IBM]
48-44-53 HDS ??[HDS ??]
80-00-07 Mac[Mac]
AA-00-00 DEC[DEC]
AA-00-01 DEC[DEC]
AA-00-02 DEC[DEC]
AA-00-03 DEC[DEC]
AA-00-04 DEC networking[DECnet]
CE-00-00 Ethernet spec (global)[gbl]
FE-FF-FF Ethernet spec (global)[gbl]
Total of 177 manufacturers defined, 177 displayed.
```

II.2. Multicast Addresses

These are the multicast address currently known to and "hardwired" into ETHERMON. There are two types of multicast addresses: those that have a single fixed address (truly hardwired), and those which are assigned a range of addresses where the specific address not only means multicast but to a specific subgroup within the total group covered by the set of multicast addresses. The latter set, in a few cases, can be recognized and identified by ETHERMON as belonging to the specific subgroup, sometimes even naming the subgroup (ex: local area VAXcluster group ID multicasts).

The fixed multicast addresses are:

```
  01-80-C2-00-00-00 bridge spanning tree
  01-DD-00-68-DE-00 U-B:??? #1
  01-DD-00-A1-BC-00 U-B:??? #2
  01-DD-00-FF-FF-FF U-B:??? #3
  01-DD-01-00-00-00 U-B:spanning tree hello
  09-00-02-04-00-01 Bridge:printer msgs
  09-00-02-04-00-02 Bridge:All Bridges (DBD/DSTD)
  09-00-07-FF-FF-FF Apple:Appletalk ARP brdcst
  09-00-09-00-00-01 H-P:Probe
  09-00-09-00-00-04 H-P:DTC
  09-00-1E-00-00-00 Apollo:DOMAIN
  09-00-2B-00-00-00 DEC:MUMPS?
  09-00-2B-00-00-01 DEC:DSM/DTP?
  09-00-2B-00-00-03 DEC:LAN Traffic Monitor
  09-00-2B-00-00-04 DEC:MAP End-System Hello
  09-00-2B-00-00-05 DEC:MAP Intermediate-Sys Hello
  09-00-2B-00-00-06 DEC:CSMA/CD Encryption
  09-00-2B-00-00-07 DEC:NetBIOS Emulator (PCSG)
  09-00-2B-00-00-0F DEC:LAT (LANCE)
  09-00-2B-01-00-00 DEC:Bridge Mgmt
  09-00-2B-01-00-01 DEC:Bridge hello
  09-00-2B-02-00-00 DEC:Lvl 2 Routers
  09-00-2B-02-01-00 DEC:DNA Name Svc Advert
  09-00-2B-02-01-01 DEC:DNA Name Svc Solicit
  09-00-2B-02-01-02 DEC:DNA Time Service?
  09-00-2B-04-00-00 DEC:LA Sys Transport (LAST)
  09-00-2B-23-00-00 DEC:Argonaut Console
  09-00-77-00-00-01 Retix:bridge spanning tree
  09-00-7C-01-00-01 Vitalink:DLS Multicast
  09-00-7C-01-00-02 Vitalink:DLS Hello Multicast
  09-00-7C-01-00-03 Vitalink:DLS Inlink Hello Multi
  09-00-7C-01-00-04 Vitalink:DLS/Non-DLS Multicast
  09-00-7C-02-00-05 Vitalink:diagnostics
  09-00-7C-04-00-01 Vitalink:bridge logging
  09-00-7C-05-00-01 Vitalink:gateway?
  09-00-7C-05-00-02 Vitalink:Network Validation
  09-00-87-00-FF-FF Xyplex:Load Assistance
  AB-00-00-01-00-00 DEC:Dump/Load Assistance
  AB-00-00-02-00-00 DEC:Remote Console
  AB-00-00-03-00-00 DEC:Lvl 1,2 Routers
  AB-00-00-04-00-00 DEC:All Endnodes
  AB-00-03-00-00-00 DEC:LAT (old)
  CF-00-00-00-00-00 gbl:Loopback Assistance
  FF-FF-FF-FF-FF-FF gbl:Broadcast
Total of 44 addresses defined, 0 active, 44 displayed.
```

The variable address blocks are:

1. the DEC Local Area VAXcluster group ID multicast addresses, of the form AB-00-04-01-*xx*-*yy*, where the hexadecimal number *yyxx* is the group ID number plus 256 (decimal).
2. the AppleTalk router zone broadcast addresses 09-00-07-*aa*-*bb*-*cc*, where *aa*-*bb*-*cc* is a dynamically negotiated-for code for a particular AppleTalk zone.

II.3. Protocol Types

These are the protocol types (IEEE 802.3 SNAP subprotocols and plain-vanilla Ethernet protocols) currently known to ETHERMON, as displayed by the SHOW PROTOCOL_NAMES/ALL command. In cases where the protocol type is assigned to a particular manufacturer, that manufacturer’s "short name" (the one in square brackets in the manufacturer prefix table) is prepended to the protocol name (if the name is known). Some protocols are assigned to manufacturers but the use thereof is unknown; these have "???" as their name, or only the manufacturer’s name is given.

```
02-00 XEROX: PUP
02-01 XEROX: PUP Addr. Trans
04-00 Nixford
06-00 XEROX: NS IDP
08-00 Internet IP (DoD)
08-01 Internet IP (X.75)
08-02 Internet IP (NBS)
08-03 Internet IP (ECMA)
08-04 Chaosnet
08-05 X.25 Level 3
08-06 Internet ARP
08-07 XNS Compatibility
08-1C Symbolics: private
08-88 Xyplex: terminal net
08-89 Xyplex: ???
08-8A Xyplex: ???
09-00 U-B: Net Debug
0A-00 XEROX: 802.3 PUP
0A-01 XEROX: 802.3 PUP AdrTrans
0B-AD Banyan Systems
10-00 Berkeley: trailer
10-01 IP trailer 1 blk
10-02 IP trailer 2 blks
10-03 IP trailer 3 blks
10-04 IP trailer 4 blks
10-05 IP trailer 5 blks
10-06 IP trailer 6 blks
10-07 IP trailer 7 blks
10-08 IP trailer 8 blks
10-09 IP trailer 9 blks
10-0A IP trailer 10 blks
10-0B IP trailer 11 blks
10-0C IP trailer 12 blks
10-0D IP trailer 13 blks
10-0E IP trailer 14 blks
10-0F IP trailer 15 blks
16-00 VALID systems
3C-00 3Com:NBP VC datagram
3C-01 3Com:NBP sysctl datagram
80-66 U.Mass.
80-67 Veeco Integrated Auto
80-68 General Dynamics
80-69 AT&T
80-6A Autophon
80-6C ComDesign
80-6D Compugraphic
80-6E Landmark Graphics
80-6F Landmark Graphics
80-70 Landmark Graphics
80-71 Landmark Graphics
80-72 Landmark Graphics
80-73 Landmark Graphics
80-74 Landmark Graphics
80-75 Landmark Graphics
80-76 Landmark Graphics
80-77 Landmark Graphics
80-7A Matra
80-7B Danak Data Elektronik
80-7C U.M.:Merit Internodal
80-7D Vitalink
80-7E Vitalink
80-7F Vitalink
80-80 Vitalink: TransLAN III
80-81 Counterpoint Computers
80-82 Counterpoint Computers
80-83 Counterpoint Computers
80-9B AppleTalk
i80-9B AppleTalk
80-9C Datability
80-9D Datability
80-9E Datability
80-9F Spider Sys.LAN Monitor
80-A3 Nixdorf
80-A4 Siemens Gammasonics
80-A5 Siemens Gammasonics
80-A6 Siemens Gammasonics
80-A7 Siemens Gammasonics
80-A8 Siemens Gammasonics
```

3C-02 3Com:NBP connect req
3C-03 3Com:NBP connect resp
3C-04 3Com:NBP connect complete
3C-05 3Com:NBP close req
3C-06 3Com:NBP close resp
3C-07 3Com:NBP datagram
3C-08 3Com:NBP datagram brdcst
3C-09 3Com:NBP claim NetBIOS name
3C-0A 3Com:NBP delete NetBIOS nam
3C-0B 3Com:NBP rem sts req
3C-0C 3Com:NBP rem sts resp
3C-0D 3Com:NBP reset
42-42 PCS Basic Block
52-08 BBN: SIMnet
60-00 DEC: Experimental
60-01 DEC: MOP Dump/Load
i60-01 DEC: MOP Dump/Load
i60-02 DEC: MOP Remote Console
60-02 DEC: MOP Remote Console
60-03 DEC: DECnet Phase 4
60-04 DEC: LAT
60-05 DEC: Diagnostics
60-06 DEC: Customer/VAXELN
60-07 DEC: LAVC
60-08 DEC: AMBER
60-09 DEC: MUMPS
60-10 3Com: ???
60-11 3Com: ???
60-12 3Com: ???
60-13 3Com: ???
60-14 3Com: ???
70-00 U-B: NIU Boot
70-01 U-B: Network Management
70-02 U-B: NIU Diagnostic
70-03 Interlan: ???
70-05 U-B: Bridge
70-06 U-B: Network Management
70-20 LRT: ???
70-21 LRT: ???
70-22 LRT: ???
70-23 LRT: ???
70-24 LRT: ???
70-25 LRT: ???
70-26 LRT: ???
70-27 LRT: ???
70-28 LRT: ???
70-29 LRT: ???
70-30 Proteon: ???
70-34 Cabletron
80-03 Cronus VLN
80-04 Cronus Direct
80-05 HP Probe
80-06 Nestar
80-08 AT&T
80-10 Excelan
80-13 SilGraph: diag
80-14 SilGraph: net games
80-A9 Siemens Gammasonics
80-AA Siemens Gammasonics
80-AB Siemens Gammasonics
80-AC Siemens Gammasonics
80-AD Siemens Gammasonics
80-AE Siemens Gammasonics
80-AF Siemens Gammasonics
80-B0 Siemens Gammasonics
80-B1 Siemens Gammasonics
80-B2 Siemens Gammasonics
80-B3 Siemens Gammasonics
80-C0 DCA: ???
80-C1 DCA: Data Exchange
80-C2 DCA: ???
80-C3 DCA: ???
80-C6 Pacer Software
80-C7 Applitek
80-C8 Intergraph
80-C9 Intergraph
80-CA Intergraph
80-CB Intergraph
80-CC Intergraph
80-CD Harris
80-CE Harris
80-CF Taylor Instruments
80-D0 Taylor Instruments
80-D1 Taylor Instruments
80-D2 Taylor Instruments
80-D3 Rosemount Corp.
80-D4 Rosemount Corp.
80-D5 IBM: SNA Service
80-DD Varian Associates
80-DE I.S. TRFS
80-DF Integrated Solutions
80-E0 Allen Bradley
80-E1 Allen Bradley
80-E2 Allen Bradley
80-E3 Allen Bradley
80-E4 Datability
80-E5 Datability
80-E6 Datability
80-E7 Datability
80-E8 Datability
80-E9 Datability
80-EA Datability
80-EB Datability
80-EC Datability
80-ED Datability
80-EE Datability
80-EF Datability
80-F0 Datability
80-F2 Retix: Router Mgmt
80-F3 ARP AppleTalk
i80-F3 ARP AppleTalk
80-F4 Kinetics: Appletalk?
80-F5 Kinetics: Appletalk?
80-F7 Apollo: ???

```
 80-15 SilGraph: reserved         80-FF Wellfleet Communications
 80-16 SilGraph: XNS NameSvc      81-00 Wellfleet Communications
 80-19 Apollo: remote file sys    81-01 Wellfleet Communications
 80-2E Tymeshare: ??              81-02 Wellfleet Communications
 80-2F Tigan Inc.: ?              81-03 Wellfleet Communications
 80-35 Reverse ARP                81-07 Symbolics: private
 80-36 Aeonic Systems: ??         81-08 Symbolics: private
 80-38 DEC: LAN Bridge            81-09 Symbolics: private
 80-39 DEC: DSM/DTP               81-2B Talaris TLAP data
 80-3A DEC: Argonaut console      81-30 Waterloo Microsystems
 80-3B DEC: VAXeln (hah!)         81-31 VG Laboratory Systems
 80-3C DEC: DNA Name Service      81-37 Novell: old NetWare IPX
 80-3D DEC: Encryption            81-38 Novell: NetWare IPX
 80-3E DEC: DNA Time Service      81-39 KTI: ???
 80-3F DEC: LAN Traffic Monitor   81-3A KTI: ???
 80-40 DEC: NetBios emulator      81-3B KTI: ???
 80-41 DEC: LA Sys Transport      81-3C KTI: ???
 80-42 DEC: -unassigned-          81-3D KTI: ???
 80-44 Planning Research Corp.    81-4C SNMP
 80-46 AT&T                       81-4F TE Concepts
 80-47 AT&T                       85-2B Talaris TLAP status
 80-49 Expert Data                90-00 Loopback
 80-5B Stanford V Exper           90-01 Bridge: XNS mgmt
 80-5C Stanford V Product         90-02 Bridge: TCP/IP sys
 80-5D Evans & Sutherland         90-03 Bridge: loop detect
 80-60 Little Machines            AF-AF Logicraft: ???
 80-62 Counterpoint Computers     FF-00 BBN VITAL-LANBridge
 80-65 U.Mass.
Total of 247 protocols defined, 247 displayed.
```

Appendix III. Release Notes

III.1. V2.4-1

Changes, enhancements, and bug-fixes for V2.4-1 are:

1. fix compile-time subscript range error in SET_MANUFNAM (appeared only when compiling /CHECK=BOUNDS).
2. fix zero-length string error for filespec in LOAD and SAVE commands (appeared at run-time only after compiling /CHECK=BOUNDS).
3. fix typo in specifying control-C trap mask in ENABLE_CONTROL_C_TRAP subroutine; this prevented control-C’s from being trapped as out-of-band characters but instead specified other characters.
4. fix SET MESSAGE xxx ON command so that turn-ons would actually happen. formerly, only SET MESSAGE xxx OFF would work.
5. fix global section creation logic for cases when an explicit mapping file is needed (because insufficient global pagefile space is available). also make sure that explicit mapping file is deleted when ETHERMON terminates.
6. have control-C automatically abort using an indirect command file.
7. search for an initialization file at ETHERMON start-up time; if one exists, open it as an indirect command file and execute its commands before reading commands from standard input.
8. add more entries to known manufacturer prefix and protocol type tables.
9. add information to user’s manual regarding implementation of filters.

III.2. V2.4

Changes, enhancements, and bug-fixes for V2.4 are:

1. Add the "fast" $QIO interface that corrects for the VMS V5.4-3 (and up) Ethernet driver change. Include a performance check routine in FASTSHR and FASTCALL to get kernel-based data without forcing main ETHERMON image to be system version dependent.
2. Add control-C trapping to command parser and commands that could be aborted prematurely (DISPLAY, SHOW ADDRESSES, SHOW MANUFACTURERS, SHOW PROTOCOLS, SUMMARIZE, TRACE). Make control-C logic modular and then disable control-C traps in SPAWN command (so subprocess can field them itself).
3. Add logic to the MAP command to pick up "other-side" addresses of U-B bridges, and to take advantage of the Ungermann-Bass bridge path XNS message. Add the SHOW BRIDGES/MAP plotting command.
4. Add symbolic protocol names in command parser for usage of the /PROTOCOL qualifier (wherever it might be used).
5. Add the QUERY command, with support to query U-B bridge route tables,
6. Add recognition of Ungermann-Bass device serial numbers as symbolic Ethernet address in the syntax UB#nnnnnn.
7. Fix "feature" in V5.4-3+ drivers where they attempt to process padding in Ethernet packets received in IEEE 802.3 extended mode.

8. Clean up handling of Apollo DOMAIN ARP queries.
9. Improve command syntax error reporting by having it report which command line token the syntax error occurred at.
10. Break up the FORTRAN source file into multiple files for easier e-mail transmittal of product.
11. Added support for NRC Fusion to LOAD/INTERNET command (courtesy of Galen Tackett, DECUServe).
12. Add new information to program crash exception handler (privilege/quota, system/device info, traceback computation (non-symbolic), SYSGEN parameter data, more user environmental data). Generalize exception handling to use crash-dump handler for non-AST exceptions. Clean up exception handlers to use parameter definitions of system-defined exception codes instead of external definition (to create more compact, faster-executing FORTRAN code in handlers).
13. Clean up TRACE command code and add /STATISTICS, /TRUNCATE, /SNIFFER, and /ANALYZER qualifiers to it. Put the address table FORTRAN COMMON in a global section so /ANALYZER subprocesses can get at it. Create support subroutines to allow analyzers to verify they are running in analyzer subprocess mode and map successfully to the address table global section. Clean up "can't make global section" error message to be more meaningful to end-user. Define parameters for character array element lengths in address table COMMON. Add more stringent version checking for global section. Add logic to support creating section without using pagefile instead of just giving up and leaving user confused.
14. Correct message loss value "glitch" at initiation of a DISPLAY command.
15. Add extra informational message at ETHERMON start-up to tell users "RTFM" on failure to invoke the "fast" $QIO interface.
16. Remove erroneous and unnecessary code in LAVC packet analysis.
17. Fix privilege restoration problem in SPAWN command, where extra enabled privileges used by ETHERMON were turned off before SPAWNing a subprocess.
18. Centralize obtaining device and directory location of ETHERMON's .EXE file, so that other parts of ETHERMON can use that data without having to duplicate the code and effort to get it.
19. Add new asynchronous event report messages and extensions to SET MESSAGE command to control the new message categories. New messages are: identifying multiple nodes with duplicate IP addresses; new address table entries for source nodes.
20. Clean up the SUMMARIZE INTERNET command to report TCP & UDP port information with canonical protocol names whenever possible.
21. Correct SUMMARIZE STATISTICS and DISPLAY STATISTICS to ignore "integer overflow" of unsigned Ethernet port counter values. Correct SUMMARIZE STATISTICS averaging.
22. Make "remainder-of-line" command parameters be parsed more consistently across different commands.
23. Add DEFINE/KEY, DELETE/KEY, SET KEYPAD, and SHOW KEY commands, and pre-defined keypad commands.
24. Add the /MANUFACTURER qualifier and related support code to the LOAD and SAVE commands. Clean up flag mask processing in SAVE/ADDRESSES.
25. Fix address table scan to also check "other address" field in table (to try to prevent bridges from being entered twice in the address table).
26. Correct bandwidth calculation to account for FDDI and proper interframe gap in places

where computed bandwidth is displayed (DISPLAY, SHOW PERFORMANCE, SUMMARIZE).

27. Add table overflow checking for manufacturer prefix table in ETHER_READ.
28. Add Ethernet interface port counter saturation checks to SUMMARIZE, SHOW PERFORMANCE, SHOW PORT_STATUS, and DISPLAY commands.
29. Enlarge extended characteristics buffer for GET_PORT_STATUS and GET_PORT_COUNTERS procedures (to account for additional information returned by newer versions of VMS).
30. Fix LOAD/FILTER "feature" for address that is last item on line (was automatically converted to a manufacturers' prefix by mistake).

Index

$QIO 11

/D_LINES (FORTRAN qualifier) 33, 57, 63, 66, 68, 69

@filespec 19, 60

AARP 12, 52
ADD FILTER 19
Address 25, 29, 74
AppleTalk 12, 75
    ARP 12
    DDP 52, 58
    NBP 52
ARP 12, 52, 67, 68, 69

Batch mode 48
Bridge 12, 26, 37

CLEAR FILTER 19
CLEAR FLAGS 20
Commands
    @filespec 19, 60
    ADD FILTER 19
    CLEAR FILTER 19
    CLEAR FLAGS 20
    DEFINE/KEY 16, 20, 69
    DELETE/KEY 22, 69
    DISPLAY 18
    DISPLAY DESTINATIONS 22
    DISPLAY PROTOCOLS 23
    DISPLAY SIZES 23
    DISPLAY SOURCES 24
    DISPLAY STATISTICS 24
    EXIT 25
    HELP 25
    LOAD 25, 29, 34
    MAP 26, 38, 58, 59, 61
    MONITOR 27
    PING 27, 66
    QUERY 28
    SAVE 25, 29, 35
    SET ADDRESS_NAME 30, 34
    SET FILTER 19, 26, 31
    SET INTERVAL 22, 23, 24, 32, 39, 41, 42, 43, 44
    SET KEYPAD 21, 32
    SET MANUFACTURER 32
    SET MESSAGES 33, 57, 58, 59, 60, 61, 62, 63, 66, 67, 68, 69
    SET PROTOCOL_NAME 33
    SHOW ADDRESS_NAMES 20, 34
    SHOW BRIDGES 26, 37, 58
    SHOW FILTERS 38
    SHOW KEY 38
    SHOW MANUFACTURERS 39
    SHOW MESSAGES 39
    SHOW PERFORMANCE 39
    SHOW PORT_STATUS 25, 40, 67
    SHOW PROTOCOL_NAMES 16, 40
    SPAWN 40, 62, 69, 70
    SUMMARIZE DESTINATIONS 41
    SUMMARIZE INTERNET 41
    SUMMARIZE PROTOCOLS 42
    SUMMARIZE SIZES 42
    SUMMARIZE SOURCES 43
    SUMMARIZE STATISTICS 43
    TRACE 22, 23, 24, 25, 40, 44, 48, 51, 62

WAIT 48, 70
Control-C 18, 22, 23, 24
Control-S 59
Control-V 21
Control-Y 18
Control-Z 18, 25
Crash dump file 59

DEC LAN Bridge 12, 26
DECnet 11, 59, 62, 63, 64, 67, 68
DEFINE/KEY 16, 20, 69
DELETE/KEY 22, 69
DISPLAY 18
DISPLAY DESTINATIONS 22
DISPLAY PROTOCOLS 23
DISPLAY SIZES 23
DISPLAY SOURCES 24
DISPLAY STATISTICS 24
Domain Name Service 26

ETHERMON$INIT 9
ETHERPARSE 52
ETHERPARSE2 52
ETHERPARSE_TCP 53
ETHERPARSE_XREF 53
ETHFILT_ETHADDR 53
ETHFILT_IPADDR 54
ETHFILT_NFS 54
EXIT 25

Fast port interface 3, 5, 11
FDDI 3, 62
Filter 12, 19, 25, 26, 29, 31, 38, 53, 54

HELP 25
errors 70
Help library 8, 70

IEEE 802.3 11, 27, 35, 59, 61, 69
Initialization file 9
INSTALL 55
INSTALL utility 8, 65
Internet 11, 25, 41, 66
ARP 12, 67
RARP 12
Internet software
CMU TCP/IP 26, 65
DEC TCP/IP Services for VMS 26, 54, 65
DEC UCX 26, 54, 65
NRC Fusion 26, 65
TGV MultiNet 26, 54, 65

LAT 12
LAVC 12, 75
LOAD 25, 29, 34
Logical name 8, 55
LOGIN.COM 8

Mailbox 47, 62, 68, 69
Manufacturer prefix 25, 29, 32, 39, 72
MAP 26, 38, 58, 59, 61
Messages
ARP 33
asynchronous event 33
duplicate IP addresses 33
IP 33
MOP 33
new node detected 33

MONITOR 27
MOP 11
Multicast address 12, 19, 31, 34, 58, 74

NCP 24, 25, 40
NETACP 11, 29, 59, 63, 64, 67, 68
NFS 52, 53, 54
NIC.DDN.MIL 36, 42, 68, 69, 71

PING 27, 66
PostScript 38
Privileges
  CMKRNL 3, 11, 55, 64
  NETMBX 3, 55, 64
  PHY_IO 3, 55
  SETPRV 3, 55
  TMPMBX 3, 55
Promiscuous mode 3, 11, 55
Protocol type 25, 29, 75
  AARP (80-F3) 12, 52
  AppleTalk (80-9B) 12, 37, 52
  ARP (08-06) 12, 52, 67, 68, 69
  DECnet (60-03) 11
  FF-FF 3, 11
  ICMP 52
  IP (08-00) 11, 27, 52, 54
  LAN Bridge (80-38) 12, 26, 52, 58
  LAT (60-04) 12
  LAVC (60-07) 12
  loopback (90-00) 27
  MOP (60-02) 11, 27, 37
  NFS 52, 53, 54
  RARP (80-35) 12, 52, 68, 69
  RIP 20, 36, 52
  RPC 52, 54
  RPC portmapper 52
  SNMP 52
  TCP 42, 52, 53
  TransLAN III (80-80) 12
  UDP 42, 52, 54
  XNS (06-00) 12, 26, 37, 52, 53
  Xyplex (08-88) 12, 37

QIO 11
QUERY 28
Quotas
  BYTLM 3, 41, 48, 55, 61, 62, 64, 69, 70
  disk 70
  PRCLM 41, 48, 70
  WSDEFAULT 64

RARP 12, 52, 68, 69
RFC1057 52, 71
RFC1058 36, 52, 71
RFC1060 42, 71
RFC1094 52, 71
RFC1157 52, 71
RFC768 52, 71
RFC791 52, 71
RFC792 52, 71
RFC793 52, 53, 71
RFC826 68, 69, 71
RIP 20, 36, 52

SAVE 25, 29, 35
SET ADDRESS_NAME 30, 34
SET FILTER 19, 26, 31
SET INTERVAL 22, 23, 24, 32, 39, 41, 42, 43, 44

SET KEYPAD 21, 32
SET MANUFACTURER 32
SET MESSAGES 33, 57, 58, 59, 60, 61, 62, 63, 66, 67, 68, 69
SET PROTOCOL_NAME 33
SHOW ADDRESS_NAMES 20, 34
SHOW BRIDGES 26, 37, 58
SHOW FILTERS 38
SHOW KEY 38
SHOW MANUFACTURERS 39
SHOW MESSAGES 39
SHOW PERFORMANCE 39
SHOW PORT_STATUS 25, 40, 67
SHOW PROTOCOL_NAMES 16, 40
SPAWN 40, 62, 69, 70
SUMMARIZE DESTINATIONS 41
SUMMARIZE INTERNET 41
SUMMARIZE PROTOCOLS 42
SUMMARIZE SIZES 42
SUMMARIZE SOURCES 43
SUMMARIZE STATISTICS 43
SYLOGIN.COM 8
SYSGEN 61
SYSGEN parameters 3, 62, 69
SYSTARTUP_V5.COM 8

TRACE 22, 23, 24, 25, 40, 44, 48, 51, 62
  /ANALYZER 51, 53, 54
  file 51

Utilities
  ETHERPARSE 52
  ETHERPARSE2 52
  ETHERPARSE_TCP 53
  ETHERPARSE_XREF 53
  ETHFILT_ETHADDR 53
  ETHFILT_IPADDR 54
  ETHFILT_NFS 54

VAX/VMS
  V4.x 11, 59, 69
  V5.4-3 and up 11, 55, 64
VAXcluster 12, 75

WAIT 48, 70

XNS 12, 26, 37, 52, 53
Xyplex 12, 37

Table of Contents

1. Requirements for Use 3
   1.1. Statement of Liability 3
   1.2. Acknowledgements 4
2. Installing the Program 5
   2.1. Manifest 5
   2.2. Installation Procedure 7
3. Operation 11
   3.1. Theory of Operation 11
      3.1.1. Filtering 12
   3.2. Commands 14
      3.2.1. Command Parameters 15
         3.2.1.1. Ethernet addresses 15
         3.2.1.2. Protocol types 15
      3.2.2. Command Editing and Entry 16
         3.2.2.1. Keypad Operations 16
         3.2.2.2. Control Keys 18
      3.2.3. @indirect-command-file 19
      3.2.4. ADD FILTER 19
      3.2.5. CLEAR FILTER 19
      3.2.6. CLEAR FLAGS 20
      3.2.7. DEFINE/KEY 20
      3.2.8. DELETE/KEY 22
      3.2.9. DISPLAY DESTINATIONS 22
      3.2.10. DISPLAY PROTOCOLS 23
      3.2.11. DISPLAY SIZES 23
      3.2.12. DISPLAY SOURCES 24
      3.2.13. DISPLAY STATISTICS 24
      3.2.14. EXIT 25
      3.2.15. HELP 25
      3.2.16. LOAD 25
      3.2.17. MAP 26
      3.2.18. MONITOR 27
      3.2.19. PING 27
      3.2.20. QUERY 28
      3.2.21. SAVE 29
      3.2.22. SET ADDRESS_NAME 30
      3.2.23. SET FILTER 31
      3.2.24. SET INTERVAL 32
      3.2.25. SET KEYPAD 32
      3.2.26. SET MANUFACTURER 32
      3.2.27. SET MESSAGES 33
      3.2.28. SET PROTOCOL_NAME 33
      3.2.29. SHOW ADDRESS_NAMES 34
      3.2.30. SHOW BRIDGES 37
      3.2.31. SHOW FILTERS 38
      3.2.32. SHOW KEY 38
      3.2.33. SHOW MANUFACTURERS 39
      3.2.34. SHOW MESSAGES 39
      3.2.35. SHOW PERFORMANCE 39
      3.2.36. SHOW PORT_STATUS 40

3.2.37. SHOW PROTOCOL_NAMES 40
3.2.38. SPAWN 40
3.2.39. SUMMARIZE DESTINATIONS 41
3.2.40. SUMMARIZE INTERNET 41
3.2.41. SUMMARIZE PROTOCOLS 42
3.2.42. SUMMARIZE SIZES 42
3.2.43. SUMMARIZE SOURCES 43
3.2.44. SUMMARIZE STATISTICS 43
3.2.45. TRACE 44
3.2.46. WAIT 48

4. Utilities 51

4.1. ETHERPARSE 52
4.2. ETHERPARSE2 52
4.3. ETHERPARSE_TCP 53
4.4. ETHERPARSE_XREF 53
4.5. ETHFILT_ETHADDR 53
4.6. ETHFILT_IPADDR 54
4.7. ETHFILT_NFS 54

5. Security Considerations 55

Appendix I. Messages 57

I.1. ETHERMON Messages 57
I.2. System Messages Frequently Seen 69

Appendix II. Known Ethernet Data 71

II.1. Manufacturer Prefixes 72
II.2. Multicast Addresses 74
II.3. Protocol Types 75

Appendix III. Release Notes 79

III.1. V2.4-1 79
III.2. V2.4 79

Index 83