ETHERMON

V2.3-2

21 June 1991

Copyright © 1991 Andrew C. Pavlin

written by
Andrew C. Pavlin
General Electric Co.
Ocean & Radar Systems Department
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.

2

1. Requirements for Use

To operate, ETHERMON requires a VAX/VMS system at version 4.x or higher, and at least one Ethernet 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 and NETMBX (or SETPRV) privilege, 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 43). The system must not be running any IEEE 802.3 application using promiscuous mode on the selected Ethernet interface (if running in fallback Ethernet 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 me to bother turning it on, and most systems can't run in loopback or echo without also turning off real transmissions to 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 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. I can be reached at:

General Electric Company
Court Street Plant, Bldg. 5, Room Z3
Syracuse, NY 13221 U.S.A.
telephone: (315) 456-1875
8*256-1875 (GE/RCA DIALCOMM)
fax: (315) 456-1478
GE DECnet: C5VR::PAVLIN
Internet: PAVLIN@syr.ge.com
PAVLIN@c5vr.dnet.ge.com
ACPAULIN@sunrise.acs.syr.edu

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 Scribe Systems Inc.

SUN and NFS are trademarks of Sun Microsystems Inc.

U-B is a trademark 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 and AppleTalk 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. Building the Program

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 file and INCLUDE files defining COMMON blocks:
    - ETHERMON.FOR - main source file
    - ETHERMON_ADDR.INC - Ethernet address table
    - ETHERMON_BRIDGES.INC - bridge table and bridge interconnection map
    - 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_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_MSG.MSG - the MESSAGE source file (containing the error message definitions).
4. 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).
5. 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.
6. 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.
    - ETHERMON.PS - the user’s guide in PostScript output format.

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 off, 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
$ MACRO ETHERMON_PARSE
$ MESSAGE ETHERMON_MSG
$ LINK ETHERMON,ETHERMON_PARSE,ETHERMON_MSG
```

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

```
$ @ETHERMON
```

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

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 file; it cannot be redirected by use of logical names. If you wish to change this behavior, the subroutine HELP in ETHERMON.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). 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
                    9: 10-00 Berkeley: trailer
                   10: 10-01 IP trailer 1 blk
                   11: 10-02 IP trailer 2 blks
_ETB0:   DEBNA      1: 06-00 XEROX: NS IDP
                    3: 08-00 Internet IP (DoD)
                    4: 08-06 Internet ARP
                    5: 10-00 Berkeley: trailer
                    6: 10-01 IP trailer 1 blk
                    7: 10-02 IP trailer 2 blks
Ethernet_port_: eta0
ETHERMON V2.3-2, 21-Jun-1991
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 V2.3-2, 21-Jun-1991
ETHERMON>
```

8

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, 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).

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 physical 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 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)
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 Corporation under protocol type 06-00 (Xerox NS IDP) (prefixes 00-DD-00, 00-DD-01, and 00-DD-0F only)
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.

3.2. 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
DISPLAY DESTINATIONS
DISPLAY PROTOCOLS
DISPLAY SIZES
DISPLAY SOURCES
DISPLAY STATISTICS
EXIT
HELP
LOAD
MAP
MONITOR
PING
SAVE
SET ADDRESS_NAME
SET FILTER
SET INTERVAL
SET MANUFACTURER
SET MESSAGES
SET PROTOCOL_NAME
SHOW ADDRESS_NAMES
SHOW BRIDGES
SHOW FILTERS
SHOW MANUFACTURERS
SHOW MESSAGES
SHOW PORT_STATUS
SHOW PROTOCOL_NAMES
SPAWN
SUMMARIZE DESTINATION
SUMMARIZE INTERNET
SUMMARIZE PROTOCOL
SUMMARIZE SIZE
SUMMARIZE SOURCE
SUMMARIZE STATISTICS
TRACE
WAIT
```

3.2.1. @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

3.2.2. 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 file (if any) will be processed.

Command format:

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

where *ab-cd* is a physical Ethernet protocol type code, in the range 00-00 to FF-FF, and *address* is a node address, expressed in one of five ways:

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.
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 ADD FILTER command can look up the equivalent Ethernet address).
5. an Internet physical 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 ADD FILTER command can look up the equivalent Ethernet address).

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

3.2.3. 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 and SHOW FILTER commands), and /ALL specifies clearing all filters from the table.

3.2.4. 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

3.2.5. 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.

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.

3.2.6. 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.

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.

3.2.7. 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.

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.

3.2.8. 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.

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.

3.2.9. 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.

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.

3.2.10. 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, and all information collected by ETHERMON (except that previously stored with the SAVE command) is discarded.

Command format:

EXIT

3.2.11. 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 ...]

3.2.12. 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
/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 or protocol 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 or protocol type are not changed.

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.

At present, manufacturer tables cannot be LOADed (but you could use a command procedure filled with SET MANUFACTURER commands).

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), and Carnegie-Mellon University TCP/IP (CMU TCP/IP) 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.

3.2.13. 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), and is not guaranteed to work for those (depending on filtering, passwords, etc.); however, references to any recognized bridge (whether or not it speaks DEC LAN Bridge) will be listed in the overall bridge routing map constructed by ETHERMON. To see the computed map, issue the SHOW BRIDGES/FULL command.

3.2.14. 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

3.2.15. 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 one of:

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 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).
3. 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 PING command can look up the equivalent Ethernet address).
4. an Internet physical 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 PING command can look up the equivalent Ethernet address).

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.

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.

3.2.16. SAVE

The SAVE command writes the current contents of the ETHERMON utility's address, filter, and protocol 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*]
/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, high byte (not low) first. After the address, delimited by spaces are the equivalences for the three 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 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 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 40 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.

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 /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.

At present, manufacturer tables cannot be SAVEd.

3.2.17. 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), *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 40 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.

3.2.18. 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=ab-cd],[SOURCE=address],
                            [DESTINATION=address],[ADDRESS=address],
                            [MULTICAST])
                     /EXCLUDE=([PROTOCOL=ab-cd],[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), *ab-cd* is a physical Ethernet protocol type code, in the range 00-00 to FF-FF, and *address* is a node address, expressed in one of five ways:

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.
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 SET FILTER command can look up the equivalent Ethernet address).
5. an Internet physical 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 SET FILTER command can look up the equivalent Ethernet address).

The use of SOURCE= specifies that the *address* is only treated as a source address; DESTINATION=

specifies a destination address only; and ADDRESS= implies the address regardless of whether it was a source or destination.

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.

3.2.19. 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.

3.2.20. 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.

3.2.21. 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 ON

SET MESSAGES 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.

3.2.22. 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).

3.2.23. 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=aa-bb]
                   [/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 ETHERMON was started will actually be displayed.

The *wildcard-address* parameter allows you to restrict address display to a subset of known addresses, where the parameter is expressed as *a-b-c-d-e-f*, where the six bytes are each either hexadecimal numbers or '*' for a wildcard. The address can also be specified as *a-b-c*, where missing terms are assumed to be wildcarded. If no address is given, the address *-*-*-*-*-* is assumed. The *wildcard-address* parameter may also be given as an Internet physical address of the form *a.b.c.d*, where each number is a legal (0 to 255) decimal byte value or the '*' wildcard symbol, and 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=aa-bb qualifier allows you to restrict address display to only those addresses that have used the protocol specified (in standard Xerox hexadecimal format). 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.

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
- MACNUM (%X02000): running AppleTalk protocols
- MACZONE (%X01000): identified as AppleTalk zone router
- MOP (%X04000): running DEC MOP protocols
- 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

3.2.24. 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]
             [/OUTPUT[=file]]
```

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.

3.2.25. 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
```

3.2.26. SHOW MANUFACTUTERS

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.

3.2.27. SHOW MESSAGES

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

Command format:

```
SHOW MESSAGES
```

3.2.28. 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 (NCP) 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.

3.2.29. 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.

3.2.30. 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.

3.2.31. 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.

3.2.32. 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.

3.2.33. 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.

3.2.34. 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.

3.2.35. 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.

3.2.36. 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.

3.2.37. 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]
      [/INTERVAL=deltatime]
      [/OUTPUT=outfilespec]
      [/PARALLEL]
      [/REPEAT[=n]]

TRACE /ABORT
```

where only one of /ASCII or /BINARY may be specified (if neither 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$DISK:[]ETHERMON.DAT for binary mode, and SYS$DISK:[]ETHERMON.LIS for

ASCII 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).

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.

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. 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
1400AA000400D1280500AA000400CA280800AA340000000000000000000000000000000000
 . .   . . .   ( . .   . . .   ( . .   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
E1190100003108007C003B7101DE007F08007C0030260101050F1E0F300400008100010000D
   . . . . 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
FFFF002A00050000000108002B101AE5872C0000000100DD0102962C8713C000075EC05F0039
 . . . * . . . . . . . . + . .   . , . . . . .   . . . , . .   . . ^   _ . 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
FFFF0046000400000000FFFFFFFFFFF800100000000100DD010A12A880200000000000010000
 . . . 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
22000D020000AA000400D82803DA0500000000000000000AA00040063280F000002AAAA0000
 " . . . . .   . . .   ( .   . . . . . . . . . . . .   . . . 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
001100001360136OC7A60606060101835B
 . . . . . ' . '     . . . . . . [
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
001100001360136OC7A60606060101EB62
 . . . . . ' . '     . . . . . .   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.

FORTRAN:

```
  STRUCTURE /BIN_REC/
    INTEGER*4 DATETIME(2)  ! quadword system timestamp
    BYTE      DESTADDR(0:5)! start, XEDRIVER P5 buffer: dest addr
    BYTE      SRCADDR(0:5) ! source Ethernet address
    UNION
     MAP
      BYTE    PROTOCOL(0:1)! Ethernet protocol type
     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 ID
       END MAP
      END UNION
     END MAP
    END UNION              ! end, XEDRIVER P5 buffer (20 bytes)
    INTEGER*2 LENGTH       ! length of message body
    BYTE      BODY(1500)   ! not all of this array may be sent;
                           ! only part actually used is written
                           ! to file
  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    1530 ; maximum length of file record
        $DEFEND REC
```

3.2.38. 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

Five 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.
3. 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.
4. 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.
5. 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.

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 single 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
```

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.

4.3. 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.

4.4. 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) 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.

4.5. 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.

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.

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 buffer 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.

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-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.

7. %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.

8. %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.

9. %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.

10. %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.

11. %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.

12. %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.

13. %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 BRIDGE/FULL command to display the bridge map. Enter the MAP command again to recompute the map with the additional data.

14. %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.

    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.

15. %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 4.7 or earlier).

16. %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.

17. %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.

18. `%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.

19. `%ETHERMON-F-CRASHDUMP, crash dump of ETHERMON from ETHER_READ`

    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.

20. `%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.

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

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

22. `%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.

23. `%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 I/O error message number and correct the corresponding problem.

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

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

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

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

26. `%ETHERMON-I-FLTDEL, filter #nn deleted`

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

27. `%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.

28. `%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.

29. %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.

30. %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 OFF command.

31. %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 OFF command.

32. %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 OFF command.

33. %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.

34. %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).

35. %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.

36. %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 7750 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 or PQL_MBYTLM. The absolute minimum is determined by the formula

    ```
    value = MIN(PQL_MBYTLM,PQL_DBYTLM)
            + DEFMBXBUFQUO
            + 2000 + 1550
    ```

    where the SYSGEN parameters are used to reserve enough quota for the SPAWN command to work, the 2000 is for the normal $QIO read buffer and other I/O buffers (terminal I/O, etc.) needed by ETHERMON, and the 1550 is to account for at least one pre-allocated receive buffer for the Ethernet I/O driver (the NMA$C_PCLI_BFN extended characteristic); for each additional 1550 bytes available above this minimum, the NMA$C_PCLI_BFN value used is incremented by 1, to a maximum value of 127. Note also that these calculations are performed after some BYTLM quota has already been consumed to create the cloned device units for the Ethernet interface and the DECnet network mailbox.

37. %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. Edit the manufacturer name table size parameter ICPMNF_MAXNAMES in ETHERMON_MANUF.INC, then rebuild the ETHERMON program.

38. %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 OFF command.

39. %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 1500. Re-enter the command with a valid message size.

40. %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.

41. %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).

42. `%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.

43. `%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).

44. `%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 DEC VMS/Ultrix Connection (UCX), TGV MultiNet, 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.

45. `%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).

46. `%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.

47. `%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).

48. `%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.

49. `%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.

50. %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.

51. %ETHERMON-F-PROTBLOVF, Ethernet protocol buffer table overflow - PROGRAM ABORT

    During a DISPLAY or SUMMARIZE command, the protocol type counter table did not have any more free space. This requires a rather difficult fix to enlarge the table and rebuild the ETHERMON program. However, this also implies that you have an exceedingly busy Ethernet, which you may wish to correct.

52. %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 OFF command.

53. %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. Edit the protocol name table size parameter ICPPRT_MAXNAMES in ETHERMON_PROT.INC, then rebuild the ETHERMON program.

54. %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.

55. %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.

56. %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.

57. %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.

58. %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.

59. %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 RFC: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 OFF command.

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

    An undeciphered port characteristics code was received while performing an IO$_SENSEMODE operation on the interface. This can only appear if you have compiled ETHERMON 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*.

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

    An undeciphered port counter ID code was received while performing an IO$_SENSEMODE with IO$M_RD_COUNT operation on the interface. This can only appear if you have compiled ETHERMON 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*.

62. %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 OFF command.

I.2. System Messages Frequently Seen

1. %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.

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

   Under some installation of 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 due to 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 5598 bytes larger than DEFMBXBUFQUO for ETHERMON to operate (assuming the default value for PQL_MBYTLM; preferably, your user BYTLM quota should be considerably larger).

3. `%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 33).

4. `%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, in the chapter on Ethernet interfaces (previous editions also used)
- DoD Internet Requests For Comments (RFC's), as obtained from SRI's Network Information Center (the NIC, with Internet address NIC.DDN.MIL) via anonymous FTP:
    1. RFC 768, User Datagram Protocol, August 1980
    2. RFC 791, Internet Protocol, September 1981
    3. RFC 792, Internet Control Message Protocol, September 1981
    4. RFC 793, Transmission Control Protocol, September 1981
    5. RFC 1057, RPC: Remote Procedure Call Protocol specification: Version 2, June 1988
    6. RFC 1058, Routing Information Protocol, June 1988
    7. RFC 1060, Assigned Numbers, March 1990 (RFC 1010, May 1987, was also used in earlier versions of ETHERMON)
    8. RFC 1094, NFS: Network File System Protocol specification, March 1989
    9. RFC 1157, Simple Network Management Protocol (SNMP), 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 ("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...**}.")

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 3.

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]                02-07-01 MICOM/InterLAN[InterLAN]
00-00-0F NeXT[NeXT]                  02-60-86 Satelcom MegaPac (UK)[Satelco
00-00-10 Sytek[Sytek]                02-60-8C 3Com (IBM PC)[3Com]
00-00-1B Novell[Novell]              02-CF-1F CMC[CMC]
00-00-1D Cabletron[Cabltron]         08-00-02 3Com/Bridge[Bridge]
00-00-20 Data Industrier AB[DIAB]    08-00-03 Adv. Computer Communic.[ACC]
00-00-22 Visual Technology[VisTec]   08-00-05 Symbolics LISP machine[Symbol
00-00-2A TRW[TRW]                    08-00-07 Apple Computer[Apple]
00-00-3D AT&T[AT&T]                  08-00-08 BBN[BBN]
00-00-55 AT&T[AT&T]                  08-00-09 Hewlett-Packard[H-P]
00-00-5A S & Koch[S&K]               08-00-0A Nestar Systems[Nestar]
00-00-5E I-net Asnd.Num.Authority[IAN 08-00-0B Unisys[Unisys]
00-00-65 Network General[N-G]        08-00-10 AT&T[AT&T]
00-00-6B MIPS[MIPS]                  08-00-11 Tektronix, Inc.[Tektr]
00-00-77 MIPS[MIPS]                  08-00-14 Excelan[Excelan]
00-00-7A Ardent[Ardent]              08-00-17 Network System Corp.[NSC]
00-00-89 Cayman Systems[Cayman]      08-00-1A Data General[DG]
00-00-8E Solburne (Sun clone)[Solburn 08-00-1B Data General[DG]
00-00-93 Proteon 802.3/802.4[Proteon] 08-00-1E Apollo[Apollo]
00-00-94 Asante (Macintosh)[Asante]  08-00-20 Sun Microsystems[Sun]
00-00-9F Ameristar Technology[Amersta 08-00-22 NBI[NBI]
00-00-A2 Wellfleet[Wellflt]          08-00-25 CDC[CDC]
00-00-A3 Network Appl. Technology[N-A 08-00-26 Norsk Data[Norsk]
00-00-A6 Network General[N-G]        08-00-27 PCS Computer System GmbH[PCS]
00-00-A7 NCD[NCD]                    08-00-28 Texas Instruments[TI]
00-00-A9 Network System Corp.[N-S]   08-00-2B DEC[DEC]
00-00-AA Xerox[Xerox]                08-00-2E Metaphor[Metaphor]
00-00-AB Logic Modeling Systems[LMS] 08-00-2F Prime Computer[Prime]
00-00-B0 RAD Network Devices Ltd.[RND 08-00-36 Intergraph[Intrgrph]
00-00-B3 CIMLinc[CIMLinc]            08-00-37 Fujitsu-Xerox[Fujitsu]
00-00-B5 Vista[Vista]                08-00-38 Bull[Bull]
00-00-B7 Dove[Dove]                  08-00-39 Spider Systems[Spider]
00-00-BB Tri-Data Systems, Inc.[TriDa 08-00-41 DCA Digital Comm. Assoc.[DCA]
00-00-BC Allen-Bradley[A-B]          08-00-45 ???? (Xylogics?)[????]
00-00-C0 Western Digital[W-D]        08-00-46 Sony[Sony]
00-00-C6 Hewlett-Packard INO(Eon)[HP 08-00-47 Sequential[Sequent]
00-00-C8 Altos[Altos]                08-00-49 Univation[Univatn]
00-00-C9 Emulex (term.servers)[Emulex 08-00-4C Encore[Encore]
00-00-D7 Dartmouth College[Dartmth]  08-00-4E BICC[BICC]
00-00-D8 3Com/Novell ?? PS/2[? PS/2] 08-00-56 Stanford University[Stanford]
00-00-DD Gould[Gould]                08-00-58 ??? (DECsystem-20)[?DEC20]
00-00-DE Unigraph[Unigraph]          08-00-5A IBM[IBM]
00-00-E2 Acer Counterpoint[Acer C]   08-00-67 Comdesign[Comdesgn]
00-00-EF Alantec[Alantec]            08-00-68 Ridge[Ridge]
00-00-FD High Level Hardware[HLH]    08-00-69 Silicon Graphics/Iris[Iris]
00-01-02 BBN internal[BBN]           08-00-6A AT&T[AT&T]
00-17-00 Kabel[Kabel]                08-00-6E Excelan/Concurrent[Cncrrnt]
00-80-2D Xylogics, Inc.[Xylogic]     08-00-75 Danish Data ElektronikAS[DDE]
00-80-8C Frontier Software Dvlpmt[Fro 08-00-77 Retix Inc.[Retix]
00-AA-00 Intel[Intel]                08-00-7C Vitalink bridge[Vitalink]
00-DD-00 Ungermann-Bass[U-B]         08-00-80 XIOS[XIOS]
00-DD-01 Ungermann-Bass[U-B]         08-00-86 Imagen/QMS[I/QMS]
00-DD-02 Ungermann-Bass[U-B]         08-00-87 Xyplex[Xyplex]
00-DD-03 Ungermann-Bass[U-B]         08-00-89 Kinetics[Kinetics]
```

```
00-DD-04 Ungermann-Bass[U-B]          08-00-8B Pyramid[Pyramid]
00-DD-05 Ungermann-Bass[U-B]          08-00-8D XYVision[XYVision]
00-DD-06 Ungermann-Bass[U-B]          08-00-8E Jupiter[Jupiter]
00-DD-07 Ungermann-Bass[U-B]          08-00-90 Retix Inc.[Retix]
00-DD-08 Ungermann-Bass[U-B]          10-00-5A IBM[IBM]
00-DD-09 Ungermann-Bass[U-B]          48-44-53 HDS ??[HDS ??]
00-DD-0A Ungermann-Bass[U-B]          AA-00-00 DEC[DEC]
00-DD-0B Ungermann-Bass[U-B]          AA-00-01 DEC[DEC]
00-DD-0C Ungermann-Bass[U-B]          AA-00-02 DEC[DEC]
00-DD-0D Ungermann-Bass[U-B]          AA-00-03 DEC[DEC]
00-DD-0E Ungermann-Bass[U-B]          AA-00-04 DEC networking[DECnet]
00-DD-0F Ungermann-Bass[U-B]          CE-00-00 Ethernet spec (global)[gbl]
02-04-06 BBN internal[BBN]            FE-FF-FF Ethernet spec (global)[gbl]
Total of 134 manufacturers defined, 134 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
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:???
  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 43 addresses defined, 0 active, 43 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                  80-6C ComDesign
02-01 XEROX: PUP Addr. Trans      80-6D Compugraphic
04-00 Nixford                     80-6E Landmark Graphics
06-00 XEROX: NS IDP               80-6F Landmark Graphics
08-00 Internet IP (DoD)           80-70 Landmark Graphics
08-01 Internet IP (X.75)          80-71 Landmark Graphics
08-02 Internet IP (NBS)           80-72 Landmark Graphics
08-03 Internet IP (ECMA)          80-73 Landmark Graphics
08-04 Chaosnet                    80-74 Landmark Graphics
08-05 X.25 Level 3                80-75 Landmark Graphics
08-06 Internet ARP                80-76 Landmark Graphics
08-07 XNS Compatibility           80-77 Landmark Graphics
08-1C Symbolics: private          80-7A Matra
08-88 Xyplex: terminal net        80-7B Danak Data Elektronik
08-89 Xyplex: ???                 80-7C U.M.:Merit Internodal
08-8A Xyplex: ???                 80-7D Vitalink
09-00 U-B: Net Debug              80-7E Vitalink
0A-00 XEROX: 802.3 PUP            80-7F Vitalink
0A-01 XEROX: 802.3 PUP AdrTrans   80-80 Vitalink: TransLAN III
0B-AD Banyan Systems              80-81 Counterpoint Computers
10-00 Berkeley: trailer           80-82 Counterpoint Computers
10-01 IP trailer 1 blk            80-83 Counterpoint Computers
```

```
 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
 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
 80-9B 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
 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
```

```
  80-10 Excelan                    80-EB Datability
  80-13 SilGraph: diag             80-EC Datability
  80-14 SilGraph: net games        80-ED Datability
  80-15 SilGraph: reserved         80-EE Datability
  80-16 SilGraph: XNS NameSvc      80-EF Datability
  80-19 Apollo: remote file sys    80-F0 Datability
  80-2E Tymeshare: ??              80-F2 Retix: Router Mgmt
  80-2F Tigan Inc.: ?              80-F3 ARP AppleTalk
  80-35 Reverse ARP               i80-F3 ARP AppleTalk
  80-36 Aeonic Systems: ??         80-F4 Kinetics: Appletalk?
  80-38 DEC: LAN Bridge            80-F5 Kinetics: Appletalk?
  80-39 DEC: DSM/DTP               80-F7 Apollo: ???
  80-3A DEC: Argonaut console      80-FF Wellfleet Communications
  80-3B DEC: VAXeln (hah!)         81-00 Wellfleet Communications
  80-3C DEC: DNA Name Service      81-01 Wellfleet Communications
  80-3D DEC: Encryption            81-02 Wellfleet Communications
  80-3E DEC: DNA Time Service      81-03 Wellfleet Communications
  80-3F DEC: LAN Traffic Monitor   81-07 Symbolics: private
  80-40 DEC: NetBios emulator      81-08 Symbolics: private
  80-41 DEC: LA Sys Transport      81-09 Symbolics: private
  80-42 DEC: -unassigned-          81-30 Waterloo Microsystems
  80-44 Planning Research Corp.    81-31 VG Laboratory Systems
  80-46 AT&T                       81-37 Novell: old NetWare IPX
  80-47 AT&T                       81-38 Novell: ???
  80-49 Expert Data                81-39 KTI: ???
  80-5B Stanford V Exper           81-3A KTI: ???
  80-5C Stanford V Product         81-3B KTI: ???
  80-5D Evans & Sutherland         81-3C KTI: ???
  80-60 Little Machines            81-3D KTI: ???
  80-62 Counterpoint Computers     81-4C SNMP
  80-65 U.Mass.                    90-00 Loopback
  80-66 U.Mass.                    90-01 Bridge: XNS mgmt
  80-67 Veeco Integrated Auto      90-02 Bridge: TCP/IP sys
  80-68 General Dynamics           90-03 Bridge: loop detect
  80-69 AT&T                       AF-AF Logicraft: ???
  80-6A Autophon                   FF-00 BBN VITAL-LANBridge
Total of 230 protocols defined, 230 displayed.
```

Index

/D_LINES (FORTRAN qualifier) 46

@filespec 12, 41

AARP 10
ADD FILTER 12
Address 16, 19, 51
AppleTalk 10, 52
  ARP 10
  DDP 36, 39
  NBP 36
ARP 9, 45, 46

Batch mode 33
Bridge 9, 10, 17, 25

CLEAR FILTER 13
CLEAR FLAGS 13
Commands
  @filespec 12, 41
  ADD FILTER 12
  CLEAR FILTER 13
  CLEAR FLAGS 13
  DISPLAY DESTINATIONS 13
  DISPLAY PROTOCOLS 14
  DISPLAY SIZES 14
  DISPLAY SOURCES 15
  DISPLAY STATISTICS 15
  EXIT 16
  HELP 16
  LOAD 16, 19, 23
  MAP 17, 25
  MONITOR 17
  PING 18, 44
  SAVE 16, 19, 24
  SET ADDRESS_NAME 20, 23
  SET FILTER 13, 17, 21
  SET INTERVAL 13, 14, 15, 22, 28, 29, 30
  SET MANUFACTURER 17, 22
  SET MESSAGES 22, 39, 40, 42, 43, 44, 45, 46
  SET PROTOCOL_NAME 23
  SHOW ADDRESS_NAMES 13, 23
  SHOW BRIDGES 17, 25
  SHOW FILTERS 26
  SHOW MANUFACTURERS 26
  SHOW MESSAGES 26
  SHOW PORT_STATUS 27
  SHOW PROTOCOL_NAMES 27
  SPAWN 27, 42, 46
  SUMMARIZE DESTINATIONS 28
  SUMMARIZE INTERNET 28
  SUMMARIZE PROTOCOLS 29
  SUMMARIZE SIZES 29
  SUMMARIZE SOURCES 29
  SUMMARIZE STATISTICS 30
  TRACE 14, 15, 16, 27, 30, 33, 35
  WAIT 33, 47
Control-S 41
Control-Z 16

DEC LAN Bridge 9
DECnet 9, 40, 42, 43, 45
DISPLAY DESTINATIONS 13
DISPLAY PROTOCOLS 14
DISPLAY SIZES 14

DISPLAY SOURCES 15
DISPLAY STATISTICS 15
Domain Name Service 17

ETHERPARSE 35
ETHERPARSE2 35
ETHFILT_ETHADDR 36
ETHFILT_IPADDR 36
ETHFILT_NFS 37
EXIT 16

Filter 12, 13, 16, 17, 19, 21, 26, 36, 37

HELP 16
  errors 47
Help library 6, 47

IEEE 802.3 9, 18, 24, 40, 42, 46
Internet 9, 16, 28, 44
  ARP 9, 45
  RARP 9
Internet software
  CMU TCP/IP 17, 44
  DEC UCX 17, 37, 44
  TGV MultiNet 17, 37, 44

LAT 9
LAVC 9, 52
LOAD 16, 19, 23

Manufacturer prefix 17, 22, 26, 50
MAP 17, 25
Messages
  asynchronous event 22
MONITOR 17
MOP 9
Multicast address 51

NCP 15, 27
NETACP 9, 19, 40, 43, 45
NFS 36, 37
NIC.DDN.MIL 28, 45, 46, 49

PING 18, 44
Privileges
  NETMBX 3, 43
  PHY_IO 3
  SETPRV 3
Promiscuous mode 3, 9
Protocol type 16, 19, 52
  AARP (80-F3) 10, 36
  AppleTalk (80-9B) 10, 36
  ARP (08-06) 9, 36, 45, 46
  DECnet (60-03) 9
  FF-FF 3, 9
  ICMP 36
  IP (08-00) 9, 19, 36, 37
  LAN Bridge (80-38) 9, 36
  LAT (60-04) 9
  LAVC (60-07) 9
  loopback (90-00) 18, 19
  MOP (60-02) 9, 18, 19
  NFS 36, 37
  RARP (80-35) 9, 36, 45, 46
  RIP 13, 36
  RPC 36, 37
  RPC portmapper 36
  SNMP 36

TCP 28, 36
TransLAN III (80-80) 10
UDP 28, 36, 37
XNS (06-00) 9, 36
Xyplex (08-88) 9

Quotas
BYTLM 3, 27, 42, 46
PRCLM 27

RARP 9, 45, 46
RFC1057 36, 49
RFC1058 36, 49
RFC1060 28, 49
RFC1094 36, 49
RFC1157 36, 49
RFC768 36, 49
RFC791 36, 49
RFC792 36, 49
RFC793 36, 49
RFC826 45, 46

SAVE 16, 19, 24
SET ADDRESS_NAME 20, 23
SET FILTER 13, 17, 21
SET INTERVAL 13, 14, 15, 22, 28, 29, 30
SET MANUFACTURER 17, 22
SET MESSAGES 22, 39, 40, 42, 43, 44, 45, 46
SET PROTOCOL_NAME 23
SHOW ADDRESS_NAMES 13, 23
SHOW BRIDGES 17, 25
SHOW FILTERS 26
SHOW MANUFACTURERS 26
SHOW MESSAGES 26
SHOW PORT_STATUS 27
SHOW PROTOCOL_NAMES 27
SPAWN 27, 42, 46
SRI-NIC.ARPA
See also NIC.DDN.MIL
SUMMARIZE DESTINATIONS 28
SUMMARIZE INTERNET 28
SUMMARIZE PROTOCOLS 29
SUMMARIZE SIZES 29
SUMMARIZE SOURCES 29
SUMMARIZE STATISTICS 30
SYSGEN parameters 3, 42, 46

TRACE 14, 15, 16, 27, 30, 33, 35
file 35

Utilities
ETHERPARSE 35
ETHERPARSE2 35
ETHFILT_ETHADDR 36
ETHFILT_IPADDR 36
ETHFILT_NFS 37

VAXcluster 9, 52

WAIT 33, 47

XNS 9, 36
Xyplex 9

Table of Contents

1. Requirements for Use 3
   1.1. Statement of Liability 3
   1.2. Acknowledgements 4
2. Building the Program 5
3. Operation 9
   3.1. Theory of Operation 9
   3.2. Commands 11
      3.2.1. @indirect-command-file 12
      3.2.2. ADD FILTER 12
      3.2.3. CLEAR FILTER 13
      3.2.4. CLEAR FLAGS 13
      3.2.5. DISPLAY DESTINATIONS 13
      3.2.6. DISPLAY PROTOCOLS 14
      3.2.7. DISPLAY SIZES 14
      3.2.8. DISPLAY SOURCES 15
      3.2.9. DISPLAY STATISTICS 15
      3.2.10. EXIT 16
      3.2.11. HELP 16
      3.2.12. LOAD 16
      3.2.13. MAP 17
      3.2.14. MONITOR 17
      3.2.15. PING 18
      3.2.16. SAVE 19
      3.2.17. SET ADDRESS_NAME 20
      3.2.18. SET FILTER 21
      3.2.19. SET INTERVAL 22
      3.2.20. SET MANUFACTURER 22
      3.2.21. SET MESSAGES 22
      3.2.22. SET PROTOCOL_NAME 23
      3.2.23. SHOW ADDRESS_NAMES 23
      3.2.24. SHOW BRIDGES 25
      3.2.25. SHOW FILTERS 26
      3.2.26. SHOW MANUFACTUTERS 26
      3.2.27. SHOW MESSAGES 26
      3.2.28. SHOW PORT_STATUS 26
      3.2.29. SHOW PROTOCOL_NAMES 27
      3.2.30. SPAWN 27
      3.2.31. SUMMARIZE DESTINATIONS 28
      3.2.32. SUMMARIZE INTERNET 28
      3.2.33. SUMMARIZE PROTOCOLS 29
      3.2.34. SUMMARIZE SIZES 29
      3.2.35. SUMMARIZE SOURCES 29
      3.2.36. SUMMARIZE STATISTICS 30
      3.2.37. TRACE 30
      3.2.38. WAIT 33
4. Utilities 35
   4.1. ETHERPARSE 35
   4.2. ETHERPARSE2 35
   4.3. ETHFILT_ETHADDR 36

4.4. ETHFILT_IPADDR 36
4.5. ETHFILT_NFS 37
Appendix I. Messages 39
I.1. ETHERMON Messages 39
I.2. System Messages Frequently Seen 46
Appendix II. Known Ethernet Data 49
II.1. Manufacturer Prefixes 49
II.2. Multicast Addresses 51
II.3. Protocol Types 52
Index 55