ETHERMON

V2.2

10 January 1991

written by
Andrew C. Pavlin
General Electric Co.
Government Electronic Systems Division
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 *VAX/VMS I/O User's Manual Part II* (DEUNA, DELUA, DEQNA, DELQA, DEBNT, etc.). The user account running ETHERMON must have SETPRV or PHY_IO and NETMBX privilege, and a buffered I/O byte limit quota (BYTLM) of at least 16000. 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 4.2 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.

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 - GESD
Court Street Plant, Bldg. 5, Room Z3
P.O. Box 4840
Syracuse, NY 13221
(315) 456-1875 (phone)
8*256-1875 (GE/RCA DIALCOMM)
GE DECnet: C5VR::PAVLIN
Internet: PAVLIN%c5vr.dnet.ge.com
ACPAULIN@sunrise.acs.syr.edu

1.2. Acknowledgements

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

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

MULTIBUS II is a trademark of Intel Corporation.

Scribe is a trademark of Scribe Systems Inc.

SUN is a trademark 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 is a trademark of Xerox Corporation.

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

Macintosh and AppleTalk are trademarks of Apple Computer, Inc.

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 file set contains:

1. ETHERMON.FOR, ETHERMON_ADDR.INC, ETHERMON_ETH.INC, ETHERMON_FILTER.INC, ETHERMON_PRS.INC, ETHERMON_PROT.INC, ETHERMON_STATS.INC, ETHERMON_MANUF.INC - the FORTRAN source file and INCLUDE files defining COMMON blocks.
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, and creates the ETHERMON help library.
5. ETHERMON.HLP - the HELP command source text (used to build a VMS help library file).
6. ETHERMON.HLB - the VMS help library file for ETHERMON.
7. ETHERMON.MSS - the user's guide (this document) in Scribe manuscript format.
8. ETHERMON.LPT - the user's guide in "dumb" line printer format.
9. ETHERMON.PS - the user's guide in PostScript 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:

```
$ 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 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).

When invoking ETHERMON with a RUN command, the start-up sequence 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 Ethernet interface exists on your system, ETHERMON will not query but will automatically select that one device and inform you of that action.

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 on protocol type FF-FF in promiscuous mode as a IEEE 802.3 mode receiver (Ethernet mode is a subset thereof). 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 identification 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)
2. any single-node address under protocol type 08-00 (DOD Internet IP) (assuming the IP message header is valid)
3. 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)
4. 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)
5. any address under protocol type 80-35 (Internet reverse ARP)
6. any single-node source or destination address under protocol type 08-88 (Xyplex terminal network)
7. 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)
8. all addresses assigned to DEC hardware under protocol type 80-38 (DEC Bridge management)
9. all addresses assigned to DEC hardware under protocol type 60-04 (DEC LAT)
10. certain destination addresses under protocol type 80-9B (AppleTalk)
11. all source addresses under protocol type 80-F3 (AppleTalk ARP)
12. 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
DISPLAY DESTINATIONS
DISPLAY PROTOCOLS
DISPLAY SIZES
DISPLAY SOURCES
DISPLAY STATISTICS
EXIT
HELP
LOAD
MONITOR
SAVE
SET ADDRESS_NAME
SET FILTER
SET INTERVAL
SET MANUFACTURER
SET PROTOCOL_NAME
SHOW ADDRESS_NAMES
SHOW FILTERS
SHOW MANUFACTURERS
SHOW PORT_STATUS
SHOW PROTOCOL_NAMES
SPAWN
SUMMARIZE DESTINATION
SUMMARIZE INTERNET
SUMMARIZE PROTOCOL
SUMMARIZE SIZE
SUMMARIZE SOURCE
TRACE
```

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).

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 address, 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 filters will be processed.

Command format:

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

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. DISPLAY DESTINATIONS

The DISPLAY DESTINATIONS command brings up a continuously updating screen (similar to the VMS MONITOR utility) reporting on the current top 15 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

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 15 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.

3.2.5. DISPLAY PROTOCOLS

The DISPLAY PROTOCOLS command brings up a continuously updating screen (similar to the VMS MONITOR utility) reporting on the current top 8 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

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 8 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.

3.2.6. 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
```

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.

3.2.7. DISPLAY SOURCES

The DISPLAY SOURCES command brings up a continuously updating screen (similar to the VMS MONITOR utility) reporting on the current top 15 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
```

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 15 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.

3.2.8. 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.

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

Command format:

```
EXIT
```

3.2.10. 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 ...]
```

3.2.11. 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.

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, don’t expect much from this command.

3.2.12. 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 may be used with MONITOR.

Command format:

```
MONITOR DESTINATIONS

MONITOR PROTOCOLS

MONITOR SIZES

MONITOR SOURCES

MONITOR STATISTICS
```

3.2.13. 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 32 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 'Mxxxx:' 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*>**M**
<*mode*>**P:typecode**

where <*mode*> is one of the letters **O** (for Only) or **E** (for Exclude), **S**, **D**, and **P** indicate source address, destination 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-nyte 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.

At present, manufacturer tables cannot be SAVEd.

3.2.14. 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 32 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.

3.2.15. 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 address, 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 filters will be processed.

Command format:

```
SET FILTER filternum /ENABLE
                     /DISABLE
                     /ONLY=([PROTOCOL=ab-cd],[SOURCE=address],
                            [DESTINATION=address],[MULTICAST])
                     /EXCLUDE=([PROTOCOL=ab-cd],[SOURCE=address],
                               [DESTINATION=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).

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.

3.2.16. 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 nnn

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

3.2.17. 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.

3.2.18. 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).

3.2.19. 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. 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), or "b" (for used both as source and destination).

Command format:

```
SHOW ADDRESS_NAMES [wildcard-address]
                   [/ALL]
                   [/FULL]
                   [/OUTPUT=filespec]
                   [/PROTOCOL=aa-bb]
```

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 /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 8 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.

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. This has no relation to the PROTOCOL=aa-bb option on the ADD and SET FILTER commands.

3.2.20. 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
```

3.2.21. 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" (for 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.

3.2.22. 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 *VAX/VMS I/O User's Guide Part II* in the chapter on Ethernet interfaces for information about the IO$_SENSEMODE extended characteristics buffer, and the *VAX/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 interface being used.

3.2.23. 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.

3.2.24. 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 collect address, manufacture, and protocol type information for its tables.

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.

3.2.25. 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]
```

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.

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.

3.2.26. 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]
```

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.

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 SRI-NIC.ARPA Request For Comments RFC:RFC1060.TXT).

3.2.27. 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]
```

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.

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.

3.2.28. 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]
```

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.

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.

3.2.29. 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]
```

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.

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.

3.2.30. 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]
```

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); and /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 unsupported utility ETHERPARSE is supplied to help analyze binary TRACE file recordings, and may be used as a template for your own expanded offline message analysis tools.

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.

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
```

4. Messages

4.1. 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 can only appear if you have compiled ETHERMON.FOR with the /D_LINES qualifier on the FORTRAN compiler.

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), 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-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.

6. **%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.

7. **%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.

8. **%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.

9. **%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.

10. **%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).

11. **%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.

12. **%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.

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

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

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

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

15. **%ETHERMON-I-FLTDEL, filter #nn deleted**

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

16. **%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.

17. **%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.

18. **%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.

19. **%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.

20. **%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).

21. **%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.

22. **%ETHERMON-W-LOWBYTLM, current BYTLM=***nnnn* **lower than recommended minimum 13096**

    This message will appear during ETHERMON start-up if your process does not have sufficient BYTLM quota left to make ETHERMON happy. It may appear as a warning (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.

23. **%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.

24. **%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.

25. **%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.

26. **%ETHERMON-I-NODECNET, can't access DECnet services**

    At ETHERMON start-up, a connection to DECnet 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 SETPRV or NETMBX 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.

27. **%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).

28. **%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 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.

29. **%ETHERMON-I-ONEPORT, implicitly using the one Ethernet interface** *_dda0:*

    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 it on the foreign command line).

30. **%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.

31. **%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.

32. **%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.

33. **%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.

34. **%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.

35. **%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.

36. **%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.

37. **%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 DECnet node database. This message can only appear if you have compiled ETHERMON.FOR with the /D_LINES qualifier on the FORTRAN compiler.

38. **%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 *VAX/VMS I/O User's Guide Part II*.

39. **%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 *VAX/VMS I/O User's Guide Part II*.

4.2. 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 reported 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".

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

   Under some versions of VMS with some homebrew and/or third-party software installed, sometimes the SPAWN command will fail with this error. You will also find that VMS DEBUG cannot run in subprocess mode. The reason for this is unknown; if you know, please let the world know what is going on.

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.

Table of Contents

1. Requirements for Use 3
   1.1. Statement of Liability 3
   1.2. Acknowledgements 3
2. Building the Program 5
3. Operation 7
   3.1. Theory of Operation 7
   3.2. Commands 9
      3.2.1. @indirect-command-file 10
      3.2.2. ADD FILTER 10
      3.2.3. CLEAR FILTER 11
      3.2.4. DISPLAY DESTINATIONS 11
      3.2.5. DISPLAY PROTOCOLS 11
      3.2.6. DISPLAY SIZES 12
      3.2.7. DISPLAY SOURCES 12
      3.2.8. DISPLAY STATISTICS 13
      3.2.9. EXIT 13
      3.2.10. HELP 13
      3.2.11. LOAD 13
      3.2.12. MONITOR 14
      3.2.13. SAVE 15
      3.2.14. SET ADDRESS_NAME 16
      3.2.15. SET FILTER 16
      3.2.16. SET INTERVAL 17
      3.2.17. SET MANUFACTURER 18
      3.2.18. SET PROTOCOL_NAME 18
      3.2.19. SHOW ADDRESS_NAMES 19
      3.2.20. SHOW FILTERS 20
      3.2.21. SHOW MANUFACTUTERS 20
      3.2.22. SHOW PORT_STATUS 20
      3.2.23. SHOW PROTOCOL_NAMES 21
      3.2.24. SPAWN 21
      3.2.25. SUMMARIZE DESTINATIONS 21
      3.2.26. SUMMARIZE INTERNET 22
      3.2.27. SUMMARIZE PROTOCOLS 22
      3.2.28. SUMMARIZE SIZES 23
      3.2.29. SUMMARIZE SOURCES 23
      3.2.30. TRACE 23
4. Messages 27
   4.1. ETHERMON Messages 27
   4.2. System Messages Frequently Seen 31