UNISYS

A Series System Software

Support Reference Manual

Priced Item

July 1992
Printed in U S America
8600 0478–100

UNISYS

A Series System Software

Support Reference Manual

Release Mark 4.0.0

Priced Item

July 1992
Printed in U S America
8600 0478–100

Correspondence regarding this publication may be forwarded using the Product Information card at the back of the manual, or may be addressed directly to Unisys, Technical Publications, 25725 Jeronimo Road, Mission Viejo, CA 92691-2792.

Page Status

Page	Issue
iii through iv	–110
v through vi	–100
vii through viii	–110
ix through xiii	–100
xiv through xvi	–110
xvii through xix	–100
xx	Blank
xxi through xxii	–100
xxiii through xxiv	–110
1–1 through 1–9	–100
1–10	Blank
2–1 through 2–3	–100
2–4	Blank
3–1 through 3–10	–100
4–1 through 4–142	–100
5–1 through 5–3	–100
5–4	Blank
6–1 through 6–3	–100
6–4	Blank
7–1 through 7–2	–110
7–3 through 7–6	–100
7–7	–110
7–8	–100
7–9	–110
7–10	–100
7–11 through 7–16	–110
7–16A through 7–16C	–110
7–16D	Blank
7–17 through 7–69	–100
7–70	Blank
8–1 through 8–7	–100
8–8	–110
8–9 through 8–53	–100
8–54	Blank
9–1 through 9–17	–100
9–18	Blank
10–1	–110
10–2 through 10–9	–100
10–10	Blank
11–1 through 11–2	–110
11–2A through 11–2B	–110
11–3 through 11–13	–100

continued

continued

Page	Issue
11–14	Blank
12–1 through 12–2	–100
12–3 through 12–6	–110
12–7 through 12–8	–100
12–9 through 12–13	–110
12–14 through 12–32	–100
12–33 through 12–36A	–110
12–36B	Blank
12–37 through 12–70	–100
12–71	–110
12–72 through 12–268	–100
12–269	–110
12–270 through 12–275	Deleted by –110
12–276	Blank
12–277 through 12–278	–110
12–279 through 12–293	–100
12–294 through 12–298	–110
12–298A through 12–298C	–110
12–298D	Blank
12–299 through 12–301	–100
12–302 through 12–304	–110
13–1 through 13–16	–100
A–1 through A–8	–100
Glossary–1 through 13	–100
Glossary–14	Blank
Bibliography–1 through 2	–110
Index–1 through 16	–110

Unisys uses an 11-digit document numbering system. The suffix of the document number (1234 5678–*xyz*) indicates the document level. The first digit of the suffix (*x*) designates a revision level; the second digit (*y*) designates an update level. For example, the first release of a document has a suffix of –000. A suffix of –130 designates the third update to revision 1. The third digit (*z*) is used to indicate an errata for a particular level and is not reflected in the page status summary.

About This Manual

Purpose

This manual brings together a set of various software utilities that are used to monitor, analyze, and control the computer system.

Scope

The utilities described in this manual are for A Series systems. These utilities perform such functions as recording system events, testing or analyzing data communications (data comm) facilities, initiating and analyzing memory dumps, and managing system resources.

Information on memory dump procedures can be found in the *A Series System Operations Guide.*

Audience

This manual is a reference manual intended primarily for use by systems support personnel. It is also useful to operations personnel, programmers, and managers.

Prerequisites

The users of this manual should know the information in both the *A Series System Operations Guide* and the *A Series System Administration Guide.*

How to Use This Manual

The majority of the sections of this manual discuss system utilities used for system monitoring. Each section is named after the utility it documents. A tab marks the beginning of each section.

Unless stated otherwise, all books referred to are for Unisys A Series systems.

Appendix A describes how to use railroad syntax diagrams that appear in Unisys manuals.

Organization

This manual comprises the following sections and appendix. In addition, a glossary, a bibliography, and an index appear at the end of the manual.

Section 1. BARS

This section describes the BARS utility program, which is used to monitor the performance of the system by sampling various system utilization information.

Section 2. DCAUDITOR

This section describes the DCAUDITOR program that performs analysis of an NSPAUDIT file produced by the data comm subsystem procedures of the operating system.

Section 3. DCSTATUS

This section describes the DCSTATUS utility, which makes use of the DCSYSTEMTABLES installation intrinsic to produce an analysis of the data comm tables maintained by the operating system and the network support processor (NSP) data comm subsystem.

Section 4. DUMPANALYZER

This section describes the DUMPANALYZER utility, which can be invoked with various options to analyze a memory dump.

Section 5. HARDCOPY and PRINTCOPY

This section describes the HARDCOPY and PRINTCOPY utility programs, which provide a way to capture system input commands and system messages within a disk file and print the contents of that file.

Section 6. HDU System Balancing

This section describes the functions that are used to monitor and change system utilization for the A 12 and A 15 systems.

Section 7. LOGANALYZER

This section describes the LOGANALYZER utility program, which produces reports of SUMLOG entries based on specified parameters.

Section 8. LOGGER

This section describes the LOGGER utility program, which produces reports that aid in the analysis of system performance and utilization.

Section 9. Peripheral Test Driver (PTD)

This section describes the peripheral test driver (PTD) program that interprets op-codes that are found in test case S-code files created to test peripheral equipment on a system.

Section 10. REPORT_LOG_ENTRIES

This section describes the exported master control program (MCP) procedure REPORT_LOG_ENTRIES, which DCALGOL programs can use to monitor the logging of selected types of log entries.

Section 11. STATUS_CHANGE_REQUEST

This section describes the exported MCP procedure STATUS_CHANGE_REQUEST, which DCALGOL programs can use to monitor changes in process status and system initialization status.

Section 12. SUMLOG

This section describes the system summary log file, SUMLOG, which is used on A Series systems to record information concerning jobs previously run, past operating system activity, and other related information concerning the past status of the machine environment.

Section 13. System Stability Reporting

This section explains how to use the system stability reporting (SSR) interactive support tool (ISTUTILITY) to record information about system disruptions.

Appendix A. Understanding Railroad Diagrams

This appendix explains how to use the railroad syntax diagrams that appear in Unisys manuals.

Related Product Information

***A Series System Administration Guide* (8600 0437)**

This guide provides the reader with information required to make decisions about system configuration, peripheral configuration, file management, resource use, and other matters related to system administration. This guide is written for users with some, little, or no A Series experience who are responsible for making decisions about system administration.

***A Series System Commands Operations Reference Manual* (8600 0395)**

This manual gives a complete description of the system commands used to control system resources and work flow. This manual is written for systems operators and administrators.

A Series System Operations Guide **(8600 0387)**

This guide describes the basic concepts and procedures required to operate Micro A through A 6 systems and, more generally, all A Series systems. This guide is written for A Series operators, especially those with little or no experience.

BNA Version 2 Network Encoded Messages Programming Reference Manual, Volumes 1 and 2 **(3787 5127), (3787 7598), and (3787 7529)**

This manual provides in reference format the encoded formats of BNA, CP 2000/CPDLP/MACP100 SNA PUT2, CP 2000 SNA PUT5, NCF, OSI, and TCP/IP Operations Interface Messages (OIMs), depending on the version of BNA being documented. The part number for the 1.2 version of the manual is 3787 7529, the part number for the 3.0 version of the manual is 3787 5127, and the part number for the 3.1 version of the manual is 3787 7598.

Contents

Section 5. HARDCOPY and PRINTCOPY

Section 6. HDU System Balancing

Section 7. LOGANALYZER

Section 8. LOGGER

Section 10. REPORT_LOG_ENTRIES

Section 11. STATUS_CHANGE_REQUEST

Section 12. SUMLOG

Section 13. System Stability Reporting

Figures

Tables

Section 1
BARS

This section describes the BARS utility program, which provides a way to monitor system performance by sampling various system utilization information.

General Information

The BARS utility program monitors the performance of the system and displays it in the form of numeric values and bar graphs. Various system utilization information is sampled and displayed dynamically on screen-type terminals. The information displayed and the format of the screen are user-controllable.

A sample is taken and the display updated every "cycle" seconds. The values displayed are the average of the samples taken over the last "period" seconds. The default value for the cycle is five seconds. The default value for the period is 15 seconds.

The numeric values represent either actual values (for example, the number of core-to-core moves) or percentages (for example, the percent of available processor time).

The plus signs (+) and minus signs (–) that follow numeric values indicate that the values have increased or decreased, respectively, since the last cycle, even if the actual values or bar graphics have not visibly changed (that is, a fractional change, which cannot be displayed, has occurred).

Values are also represented by bars on the screen. The format of each bar consists of number signs (#), followed by (depending on the system) capital Xs, or small, solid rectangles (the DEL character), followed by periods (.), where the number signs extend to the minimum value, the Xs extend to the current value, and the periods extend to the maximum value. The minimum and maximum values are based on the values that existed when the program was initiated or updated from the terminal by the user.

When the user enters the following command, the program executes a LOAD DEFAULT command, which displays the default screen for that system:

```
E $SYSTEM/BARS
```

The program is initialized to a different display if the file DISPLAY is label equated to a saved file. For example,

```
E $SYSTEM/BARS; FILE DISPLAY(TITLE = MY/SCREEN)
```

If the file MONITOR is label equated, the program writes the raw performance data to that file as it is received from the operating system. For example,

```
RUN $SYSTEM/BARS; FILE MONITOR(TITLE = DEFAULT)
```

BARS Commands

The following commands can be used as input to the BARS utility program.

```
<bars commands>

──┬─<bye command>─────────────────────────────────────────────┤
  ├─<cycle command>──────┤
  ├─<display command>────┤
  ├─<help command>───────┤
  ├─<load command>───────┤
  ├─<newdisplay command>─┤
  ├─<pack command>───────┤
  ├─<period command>─────┤
  ├─<save command>───────┤
  └─<words command>──────┘
```

BYE Command

The BYE command ends the program.

```
<bye command>

── BYE ──────────────────────────────────────────────────────┤
```

CYCLE Command

The CYCLE command controls the sampling and display update interval.

```
<cycle command>

── CYCLE ──<integer>─────────────────────────────────────────┤
```

Explanation

The variable <integer> represents the interval in seconds.

DISPLAY Command

The DISPLAY command displays the NEWDISPLAY input required for a specified display screen.

```
<display command>

── DISPLAY ─┬───────────────┬────────────────────────────────┤
            ├─ DEFAULT ─────┤
            └─<file name>───┘
```

The following text describes the meaning of each option:

DISPLAY	Entering DISPLAY causes the NEWDISPLAY input required for the current screen (either a user-specified screen or the default screen) to be displayed.
DEFAULT	The DEFAULT option displays the NEWDISPLAY input for the default screen for that system.
<file name>	The <file name> displays the NEWDISPLAY input for the screen saved in the specified file.

Example

The following command causes the NEWDISPLAY input for the screen saved in the file called X/Y to be displayed:

```
DISPLAY X/Y ON P
```

HELP Command

The HELP command (or TEACH command) displays information that helps the user in using the BARS utility.

```
<help command>

──┬─ HELP ──┬────────────────────────────────────────────┤
  └─ TEACH ─┘
```

LOAD Command

The LOAD command loads a previously generated screen as the new screen. The maximums, minimums, and averages for the system utilization information are reinitialized.

```
<load command>

── LOAD ─┬─────────────┬──────────────────────────────────┤
         ├─ DEFAULT ───┤
         └─<file name>─┘
```

Entering LOAD causes the current display to be loaded as the new screen. Essentially, this means that the current screen does not change, though the maximum, minimum, and average values displayed are reinitialized.

The following text describes the meaning of each option:

DEFAULT	The DEFAULT option loads the default screen for that system.
<file name>	The <file name> is the name of the file that contains the screen information to be loaded.

NEWDISPLAY Command

The NEWDISPLAY command generates a new screen that is used to display system utilization information.

```
<newdisplay command>

— NEWDISPLAY ————————————————————————————————————————|
```

Explanation

In order to use the NEWDISPLAY command to modify an existing screen, which is the typical case, the user enters DISPLAY followed (optionally) by one of its options and is presented with a screen in a form suitable for modification and transmission as NEWDISPLAY input.

After the DISPLAY command is entered, the keyword NEWDISPLAY appears on a line by itself in the upper left-hand corner of the screen. The NEWDISPLAY line is followed by the screen keywords, a group of Ns, and a group of Bs. The Ns are used to designate the field for displaying the numeric value of the keyword item. The Bs are used to designate the bar graph field for that item.

The user can then modify the screen by adding, deleting, or rearranging the keywords, Ns, and Bs. Optional text placed in single quotation marks (') can also be added. The screen is then transmitted by placing the cursor at the upper left-hand corner of the screen. After the screen is transmitted, it is presented as a dynamic display, which can be saved by using the SAVE command.

Rather than modifying an existing screen, the user can also use the NEWDISPLAY command to design a new one. The keyword NEWDISPLAY must be entered in the upper left-hand corner of the screen and any other information on that line must be deleted. Then any keywords can be entered on the screen, followed by optional Ns and Bs. Optional text placed in single quotation marks can also be added.

As in the case above, after the screen is transmitted, it is presented as a dynamic display, which can be saved by using the SAVE command.

In order for NEWDISPLAY input to be accepted, the following requirements must be met:

- The keyword NEWDISPLAY must appear on a line by itself in the upper left-hand corner of the screen.
- The keywords must be valid keywords (available through the WORDS command).
- The entire screen must be transmitted by placing the cursor at the upper left-hand corner of the screen.
- Optional comments or text must be placed in single quotation marks (').

Example

```
IDLE NN BBBBBBBB

'on PACK, system XYZ'
```

The preceding code produces a display like the following:

```
Idle 45 ###XX...

 on PACK, system XYZ
```

PACK Command

The PACK command displays the unit numbers, family indexes, and labels of the packs currently on line. On A Series I/O processor (IOP) systems, the channel or base unit identification of the pack string is also displayed. PK and PERPK are synonyms for PACK.

```
<pack command>

──┬─ PACK ──┬──────────────────────────────────────────────┤
  ├─ PK ────┤
  └─ PERPK ─┘
```

PERIOD Command

The PERIOD command changes the value of the period. If PERIOD is less than CYCLE, no averaging of system utilization information is done; that is, the exact value of each sample is displayed.

```
<period command>

── PERIOD ──────────────────────────────────────────────────┤
```

Explanation

Several values are computed as running averages using the following formula:

```
NEWAVERAGE := ((OLDAVERAGE * (PERIOD - CYCLE)) + (NEWVALUE * CYCLE))
              / PERIOD
```

The variable <integer> represents the period in seconds.

SAVE Command

The SAVE command saves the current display in a specified file.

```
<save command>

── SAVE ──<file name>─┬────────────────────────┬────────────┤
                      └─ ON ──<family name>────┘
```

Example

The following command saves the current display in the specified file (namely, X/Y):

```
SAVE X/Y ON P
```

WORDS Command

The WORDS command displays the allowed keywords for the NEWDISPLAY input, plus a short description of the meaning of each.

```
<words command>
— WORDS ——————————————————————————————————————————————|
```

Keywords

The following table includes all allowed keywords for the BARS utility:

Keyword	Description
LASTCYCLE	Number of seconds since last display
AVGCYCLE	Average number of seconds between displays
IDLE	Percentage of available processor time spent idling
MCP	Percentage of available processor time spent on nonvisible user or operating system processes
USER	Percentage of available processor time spent on visible user or operating system processes
OVHD	Percentage of available processor time spent stack switching or stack searching, and IOfinish time
PBIT	Percentage of total available processor time spent performing presence bit operations
INIT	Percentage of total available processor time spent doing initial presence bit operations
FREE	Percentage of total available processor time spent performing presence bit operations after resumption
SAVE	Save memory
OLAY	Overlayable memory
AVAIL	Available memory
OVRFL	Overlay-file words for overlayable memory
CCACT	Number of core-to-core moves
COACT	Number of overlays
OCACT	Number of overlay-file presence bit operations
PCACT	Number of code file presence bit operations

continued

continued

Keyword	Description
CLACT	Number of core-to-limbo overlays
CCWRD	Number of words moved core-to-core
COWRD	Number of words overlayed
OCWRD	Number of words read from overlay files
PCWRD	Number of words read from code files
CLWRD	Number of read-only words overlayed
TRFC	Percentage of time when overlay I/O in process
MIX	Number of tasks in mix
STED	Number of tasks suspended by the system
READY	Number of tasks waiting for processors
SCHED	Number of scheduled tasks

In addition, a keyword is shown for each online labeled pack, which allows the number of I/O operations waiting for that unit to be displayed.

MONITOR File Format

If you give the file MONITOR a label equation when you run the BARS program, the resulting performance data is written to that file as the data is received from the operating system. You can analyze the resulting captured performance data by using the MONITOR file information that follows.

The MONITOR file has the following format:

```
Record 1
Record 2
Record 3
Record 1
Record 2
Record 3
•
•
•
```

Each record is 180 words long. The record contents are displayed below:

Record Number	Information
1	Operational information
2	Utilization times and counts information
3	Queue information

The contents of MONITOR records 1, 2, and 3 are detailed below.

MONITOR Record 1 (Operational Information)

You can obtain information in this record by making a type 2 SYSTEMSTATUS call. Refer to the *A Series SYSTEMSTATUS Programming Reference Manual* for more information.

The following table shows the structure of record 1:

Word	Data Type	Name	Corresponding Type 2 SYSTEMSTATUS Word
0	Integer	MIXCOUNT	6
1	Integer	AVAILCORE	9
2	Integer	OLAYCORE	10
3	Integer	SAVECORE	11
4	Real	CCOLAY	15
5	Real	CCWORDS	16
6	Real	COOLAY	17
7	Real	COWORDS	18
8	Real	OCOLAY	19
9	Real	OCWORDS	20
10	Real	PCOLAY	21
11	Real	PCWORDS	22
12	Real	CLOLAY	23
13	Real	CLWORDS	24
14	Integer	OLAYCHANNELS	27
15	Integer	SUSPENDEES	29
16	Integer	BATCHOLAYDISKSIZE	41

MONITOR Record 2 (Utilization Times and Counts Information)

You can obtain the information in this record by making a type 7, subtype 1 SYSTEMSTATUS call. Refer to the *SYSTEMSTATUS Reference Manual* for more information.

The following table shows the structure of record 2:

Word	Data Type	Contents
0	Integer	Writes the current time to the MONITOR file in increments of 2.4 microseconds.
1-39	Integer	Equivalent to words 0 through 38 of a type 7, subtype 1 SYSTEMSTATUS CALL
40	Integer	Total time spent by all processors in operating system tasks

continued

continued

Word	Data Type	Contents
41	Integer	Total time spent by all processors in user tasks

MONITOR Record 3 (Queue Information)

You can obtain the information in this record by making a type 23 SYSTEMSTATUS call. Refer to the *SYSTEMSTATUS Reference Manual* for more information.

The structure of record 3 is equivalent to the structure of the type 23 SYSTEMSTATUS call.

Section 2
DCAUDITOR

DCAUDITOR is a program that performs analysis of an NSPAUDIT file produced by the data comm subsystem procedures of the operating system.

The user can identify the items to be audited by using the ID (Initialize Data Comm) system command audit options. When one more more of the audit options is set, the system takes the following actions:

- Begins writing the requested types of audit information to a file called NSPAUDIT/DCCONTROL/<NSP unit ID>
- The next time data comm is initialized, writes data comm initialization information to a file called NSPAUDIT/DCINITIAL/<NSP unit ID>

DCAUDITOR performs detailed analysis for NSP requests and results; however, only the TYPE/CLASS field is analyzed for DCWRITE formatted requests and results.

DCAUDITOR RUN Statement

The run statement of DCAUDITOR is as follows:

```
RUN *SYSTEM/DCAUDITOR("<DCAUDITOR options>"); VALUE = <nnn>
```

In this statement, <DCAUDITOR options> is a string of options separated by spaces and <nnn> is the unit number of the NSP whose audit file is to be analyzed.

Options

```
<DCAUDITOR options>

──┬────────────────────────┬──────────────────────────────┤
  ├─ DCINITIAL ────────────┤
  ├─ DCCONTROL ────────────┤
  ├─<backspace count>──────┤
  ├─ LINES ──<range>───────┤
  ├─ LSNS ──<range>────────┤
  └─ STATIONS ──<range>────┘

<backspace count>

──<integer>───────────────────────────────────────────────┤

<range>

──<integer-1>──┬─────────────────┬────────────────────────┤
               └─<integer-2>─────┘
```

The following text describes the meaning of each option:

DCINITIAL and DCCONTROL	The DCINITIAL and DCCONTROL options specify which file is to be analyzed. DCINITIAL selects the NSPAUDIT file created during NSP initialization, and DCCONTROL selects the NSPAUDIT file created after the NSP has initialized. If these options are not specified, the file that is label-equated to SCAUDITF is analyzed.
<backspace count>	The <backspace count> option restricts the analysis to <backspace count> records of the NSPAUDIT file. Valid integers are 1 through 1048575.
LINES	The LINES option allows the selective analysis of NSP requests and results that pertain to a range of line numbers. A line number is assigned by the Network Definition Language II (NDLII) compiler and is the ordinal number of the line in the SOURCENDLII, starting at 1. If this option is used, no DCWRITE requests and results are displayed.
LSNS	The LSNS option allows the selective analysis of NSP requests and results that pertain to a range of logical station numbers. A logical station number is assigned by the operating system and is the ordinal number of the station in the SOURCENDLII, starting at 2. If this option is used, no DCWRITE requests and results are displayed.
STATIONS	The STATIONS option allows a selective analysis of NSP requests and results that pertain to a range of station numbers. A station number is assigned by the NDLII compiler and is the ordinal number of the station in the SOURCENDLII, starting at 1. If this option is used, no DCWRITE requests and results are displayed.

Sample Report

Figure 2–1 is an example of output from the DCCONTROL command. Messages for DCWRITE formatted requests and results are printed in hexadecimal (hex) characters.

```
DCAUDITOR INPUT: DCCONTROL

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        DCAUDITOR LEVEL IS: 04.031.1                                                                                          %
%        MCP NAME WAS: *SYSTEM/MCP/39030E                                                                                      %
%        MCP LEVEL WAS: 39.031.3933                                                                                            %
%        MACHINE TYPE WAS: A15                                                                                                 %
%        THE NSP FIRMWARE LEVELS ARE: (PROM = 28.0), (RAM = 28.5)                                                              %
%        DATACOM FILE PREFIX WAS: SJA15C                                                                                       %
%        THIS AUDIT TRAIL WAS PRODUCED ON THURSDAY, JULY 12, 1990 AT 12:58:28                                                  %
%        THIS AUDIT TRAIL WAS PRODUCED FROM THE FOLLOWING FILE:*NSPAUDIT/DCCONTROL/0107 ON DISK                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


********************************************************************************************************************************
##### DATACOM DUMP REQUESTED #####
 ACTIVE LSN = 1
********************************************************************************************************************************
##### DCPREQUEST #####      IOLENGTH = 14 WORDS     TIMESTAMP = 12:58:29.07
$$$$$ STATION QUEUED FOR LSN 227 $$$$$
 DCP MESSAGE TYPE = OUTPUT REQUEST
        070000800000 000000000000 0A0029000000 000000000000 000000210000 000000000000 105940220100 710024030500 DF0006070E01
        0B010504D401 0E0006070E01 0F0006070E01 110006070E00 000000000000
********************************************************************************************************************************
##### DCPREQUEST #####      IOLENGTH = 14 WORDS     TIMESTAMP = 12:58:29.08
$$$$$ STATION QUEUED FOR LSN 227 $$$$$
 DCP MESSAGE TYPE = OUTPUT REQUEST
        070000800000 000000000000 0A0029000000 000000000000 000000210000 000000000000 106164220100 730024030500 DF0006070E01
        0B010504D401 0E0006070E01 0F0006070E01 110006070E00 000000000000
********************************************************************************************************************************
##### DCPREQUEST #####      IOLENGTH = 6 WORDS     TIMESTAMP = 12:58:30.48
$$$$$ PENDING QUEUED FOR LSN 227 $$$$$
 DCP MESSAGE TYPE = OUTPUT REQUEST
        070000800000 000000000000 0A0029000000 000000000000 000000210000 000000000000
********************************************************************************************************************************
##### SCREQUEST #####      IOLENGTH = 84 BYTES     TIMESTAMP = 12:58:30.50
$$$$$ SIDELINKED TO THE ABOVE REQUEST IN PENDING Q $$$$$
 REQUEST NUMBER  (DOUBLE) VALUE = 31658798(01E3132E)
 REQUEST TYPE  (ENUMERATION) VALUE = 23(17): OUTPUT
 STATION NUMBER  (INTEGER) VALUE = 482(01E2)
 TEXT SIZE  (INTEGER) VALUE = 41(0029)
 OUTPUT STATUS REQUIRED  (BOOLEAN) VALUE = TRUE (01)
 PRIORITY  (BYTE) VALUE = 0(00)  "?"
 SUPPRESS EDIT  (BOOLEAN) VALUE = FALSE(00)
 SIGNAL  (BYTE ARRAY) LENGTH = 3 BYTES.
          (0) 00 00 00                                                                   ???
 REPLY  (BYTE ARRAY) LENGTH = 14 BYTES.
          (0) 00 00 00 00 00 0C 3A 19 15 00 00 00 0A 00
 TEXT  (TEXT ARRAY) LENGTH = 41 BYTES.
        (0000) 10 62 4C 22 01 00 09 00 24 03 05 00 DF 00 06 07 0E 01 0B
        (0020) 05 04 D4 01 0E 00 06 07 0E 01 0F 00 06 07 0E 01 11 00 06
        (0040) 0E
********************************************************************************************************************************
##### DCPREQUEST #####      IOLENGTH = 6 WORDS     TIMESTAMP = 12:58:30.52
$$$$$ PENDING QUEUED FOR LSN 227 $$$$$
 DCP MESSAGE TYPE = OUTPUT REQUEST
        070000800000 000000000000 0A0029000000 000000000000 000000210000 000000000000
********************************************************************************************************************************
```

Figure 2–1. DCCONTROL Option Output

Section 3
DCSTATUS

DCSTATUS is a DCALGOL program that makes use of the DCSYSTEMTABLES installation intrinsic to produce run-time "snapshots" of the data comm tables maintained by the operating system and the data comm subsystem. (For further information about the DCSYSTEMTABLES installation intrinsic, refer to the *A Series DCALGOL Programming Reference Manual.*)

DCSTATUS analyzes elements of the data comm subsystem for the A Series systems.

No attempt is made in this section to interpret the results generated by the DCSTATUS program, because understanding these results requires an understanding of Network Definition Language II (NDLII), as well as a general familiarity with the network support processor (NSP).

Execution

The DCSTATUS program can be invoked as follows:

- Through the Command and Edit (CANDE) *DCSTATUS* command
- Through the DIAGNOSTICMCS *DP* command
- Through a CANDE or Work Flow Language (WFL) *RUN* statement

CANDE DCSTATUS Command

The CANDE *DCSTATUS* command can be entered from a remote terminal to execute DCSTATUS and produce a run-time analysis of the current state of the data comm subsystem.

<CANDE dcstatus command>

```
— DCSTATUS ─┬───────────────────────┬─┬───────────────────────┬──────────┤
            └─<dcstatus option list>┘ │ ┌←──────────────────┐ │
                                      └─┴─ ; ──<modifier>───┴─┘
```

The following text describes the meaning of each variable:

<dcstatus option list>	The <dcstatus option list> must consist of a string of standard options allowed by DCSTATUS. If <dcstatus option list> is not specified, the default is the STATION option with the <lsn> specification set to the user's logical station number (LSN). The output is directed to the user's terminal. (Refer to "DCSTATUS Options" in this section for a complete description of each option.) ***Note:*** *The ALL, NSP, and LSP options produce voluminous output.*

continued

continued

<modifier>	For a definition of <modifier>, refer to the *A Series CANDE Operations Reference Manual.*

Examples

```
DC
#RUNNING 3854
STATION 13
SYSTEM/DCSTATUS (3.7.140)  DATE : 08/14/78  TIME : 1540:08
DCPREFIX: *SYSTEM
STATION 13
DCC STATION TABLE
  ENABLED : READY : ATTACHED :
  MCS = 1: LSN = 13 : WIDTH = 80
  STATION REMOTE TYPE = 0 : RETRY COUNT = 0 : NIF INDEX = 262
  PRIMARY Q = 8,CURRENT Q = 8,STN Q = 0
  DLS : 0,2,2 ATTACHED TO FILE 1 REL STN NO = 1
  NORMAL TERMINATION
#
```

For more information, see "DCSTATUS Options" in this section.

DIAGNOSTICMCS DP Command

The DIAGNOSTICMCS *DP* command might be entered from a remote terminal to initiate DCSTATUS and produce a run-time analysis of the current state of the data comm subsystem. For information beyond that given here, refer to the *A Series DiagnosticMCS Reference Manual.*

<diagnosticmcs dp command>

```
                                                 ┌←──────── ; ────────┐
── ? ── DP ─┬─────────────────────┬─┬─ REMOTE ─┬─┴─<dcstatus option>──┴─┤
            └─ ON ──<family name>─┘ └─ SITE ───┘
```

The following text describes the meaning of each option:

ON <family name>	The ON <family name> option might be used to specify the family on which DCSTATUS resides.
REMOTE	Specifying REMOTE causes DCSTATUS output to be sent to the remote station initiating the DP.
SITE	Specifying SITE causes DCSTATUS output to be directed to the printer.
<dcstatus option>	Refer to "DCSTATUS Options" in this section for a complete description of each option.

Examples

```
?DP REMOTE(STATION 5)
## OK ##
## FILE OPEN ##

STATION 5

SYSTEM/DCSTATUS (3.7.140) DATE : 08/14/86 TIME : 0623:28
DCPREFIX : *SYSTEM
STATION 5
DCC STATION TABLE
ENABLED : READY : ATTACHED :
MCS = 1: LSN = 13 : WIDTH = 80
STATION REMOTE TYPE = 0 : RETRY COUNT = 0 : NIF INDEX = 262
PRIMARY Q = 8,CURRENT Q = 8,STN Q = 0
DLS : 0,2,2 ATTACHED TO FILE 1 REL STN NO = 1
NORMAL TERMINATION

## FILE CLOSE ##
```

CANDE and WFL Run Statements

Either of the following CANDE statements can be entered from a remote terminal to initiate DCSTATUS:

```
RUN *SYSTEM/DCSTATUS ("<dcstatus option list>")

EXECUTE *SYSTEM/DCSTATUS ("<dcstatus option list>")
```

The following WFL job deck can be entered from an operator display terminal (ODT) to initiate DCSTATUS:

```
BEGIN JOB;
  RUN *SYSTEM/DCSTATUS ("<dcstatus option list>");
END JOB
```

Refer to "DCSTATUS Options" in this section for a complete description of each allowable DCSTATUS option.

The [<task identifier>] and <task equation list> specifications can be added to the WFL *RUN* statement as desired. (Refer to the *A Series Work Flow Language (WFL) Programming Reference Manual* for more information about the RUN statement.)

When either the CANDE or WFL *RUN* statement is used, DCSTATUS output is sent to the line printer unless LINE is file-equated to KIND=REMOTE.

For more information, see "DCSTATUS Options" in this section.

DCSTATUS Options

The <dcstatus option list> specifies those elements of the data comm subsystem that are to be analyzed. These options are shown in the following diagram in this subsection. The following options are arranged hierarchically so that the earlier elements listed include all those that follow: ALL, NSP, LSP, and STATION. In other words, each higher-order item in the hierarchy (they are listed from highest to lowest) is inclusive of all lower-order items. For example, if LSP is specified, the analysis is performed on all lines and stations on that line support processor (LSP). However, the options TERMINAL, TABLES, NETWORK, GRAPH, and FILE do not fit into this hierarchy. A full explanation of each option is given in the text that follows the syntax for <dcstatus option list>.

The internal file name for the output file is LINE. When the WFL *RUN* statement or the CANDE *RUN* statement is used to initiate DCSTATUS, output is sent to the line printer by default. When the CANDE *DCSTATUS* command or the DIAGNOSTICMCS *DP* command is used, the output is sent to a remote terminal by default. The output format is modified to fit a 72-character line width. The file LINE can be file-equated in any of the above cases if an output device different from the default is wished.

The DCSYSTEMTABLES intrinsic does not lock the various tables that it accesses. Therefore, the contents of the tables might change while the intrinsic is accessing them. In addition, more than one call on the DCSYSTEMTABLES intrinsic is made to obtain the contents of all the various data comm tables. If the data comm tables maintained by the operating system change between calls, the results produced by DCSTATUS can appear internally inconsistent.

By using the FILE statement, analysis can be performed on DATACOMINFO files other than the currently active DATACOMINFO file. The following are the only allowable options for inactive DATACOMINFO files:

- *NSP <unit ID> NDL* or *NSP NDL*
- *STATION <station number> NDL* or *STATION NDL*
- TERMINAL
- GRAPH
- NETWORK

```
<dcstatus option>

──┬─ ALL ────────────────────────────────────────────────────────────┤
  ├─ NSP ──┬─<unit ID>──────────────────────────┤
  │        ├──────────┬─ NDL ───────────────────┤
  │        └─<unit ID>┘                         │
  ├─ LSP ──<unit ID>──┬─────────────────────────┤
  │                   └─┬─ . ─┬─<adapter number>┤
  │                     └─ : ─┘                 │
  ├─ STATION ─┬───────┬─┬───────────────────────┤
  │           └─<LSN>─┘ └─ NDL ─────────────────┤
  ├─ TERMINAL ─┬────────────────────────────────┤
  │            └─<terminal number>──────────────┤
  ├─ TABLES ────────────────────────────────────┤
  ├─ GRAPH ─┬───────────────────────────────────┤
  │         └─<DC file prefix>──────────────────┤
  ├─ NETWORK ─┬─────────────────────────────────┤
  │           └─<DC file prefix>────────────────┤
  └─ FILE ─┬────────────────────────────────────┤
           └─<DC file prefix>───────────────────┘

<DC file prefix>

──<identifier>─┬──────────────────────────┬──────────────────────────┤
               └─ ON ──<family name>──────┘
```

The following text describes the meaning of each option:

ALL	Produces a complete analysis of the data comm network. Analysis of all NSP, LSP, and station tables, together with an analysis of the NDLII for each station and terminal, is performed.
NSP	Produces an analysis of line support processors (LSPs), lines, and stations on all network support processors (NSPs), or on a specific NSP. Use of the NDL option causes reporting to be based on the information contained in the DATACOMINFO file instead of that contained in the current data comm tables.
LSP	Produces an analysis of the lines and stations on the designated line support processor (LSP). The <adapter number> option produces an analysis of the designated adapter on the line and its stations.
STATION	Produces a station analysis. If no <lsn> is specified, all stations are analyzed. The normal sources of information for the STATION option are the operating system data comm tables. If the NDL option is specified, the source of information is the DATACOMINFO file.
TERMINAL	Produces a listing of the NDLII specifications of all terminals or of the designated terminal. If no <terminal number> is specified, all terminals are analyzed; otherwise, the specified terminal is analyzed. Terminals are numbered in the sequence in which they appear in the NDLII terminal definitions.
TABLES	Produces a raw hexadecimal dump of the data communications controller (DCC) tables and the NSP line and station tables.
GRAPH	Produces a graph of the data comm network showing the relationship between the NSPs, LSPs, lines (names and addresses), and stations (names and LSNs). Because the graph information is obtained from the network definition files specified by the <DC file prefix>, the GRAPH option can be used whether data comm is running or not. If the <DC file prefix> is not specified, the one currently being used by the system is GRAPHed.

continued

continued

NETWORK	Produces a brief tabular network configuration report. Information in the report includes NSP, LSP, line/adapter, station, terminal, and MCS data. Because the network information is obtained from the network definition files specified by the <DC file prefix>, the NETWORK option can be used whether data comm is running or not. If the <DC file prefix> is not specified, the one currently being used by the system is analyzed.
FILE	When the <DC file prefix> is specified, limited analysis can be performed on any inactive DATACOMINFO file. If the <DC file prefix> is not specified, the one currently being used by the system is used. For example, suppose DCSTATUS is supplied with the following <dcstatus option list>: ("FILE A/B; NETWORK; GRAPH; FILE; NETWORK; GRAPH") The first NETWORK and GRAPH reports are generated using A/B/DATACOMINFO; the remaining reports use the DATACOMINFO files that are currently being used by the system.

Examples

The following are introductions to the examples that appear on the following pages:

1. Figure 3–1 shows a portion of what DCSTATUS would produce on the line printer using the NETWORK option. A table of information about the network is given that includes the headings NSP#, LSP#, LINE#, STATION type, LSN#, adapter#, Receive Address (RA), Transmit Address (TA), terminal type, synchronous or asynchronous mode, bits-per-second transmitted, class declared for the terminal in the DATACOMINFO file and the name of the message control system (MCS) in use.
2. Figure 3–2 shows a portion of what would be produced on the terminal when the GRAPH option is specified. A chart of the data comm network is given showing the NSP, LSPs, designated lines, and line stations. The chart includes the type of terminal associated with each line station along with the logical station number (LSN) for each terminal. Other information such as line adapter numbers is also given.
3. Figure 3–3 shows the output from the DCSTATUS option TERMINAL on the line printer or at a remote terminal. The DATACOMINFO settings for the terminals are given in categories such as MAXINPUT, MAXOUTPUT, RECEIVE-ADDRESS-SIZE, and so forth.
4. Figure 3–4 shows the output from the DCSTATUS option STATION on a line printer. The station name and terminal name are given along with information about the DCC station table.

```
NETWORK
                                   *****************************************************************
                                   *        D A T A C O M M   N E T W O R K   S U M M A R Y        *
                                   *                                                               *
                                   *            DCPREFIX: SJA15C.                                  *
                                   *                                                               *
                                   *            DATE : 08/06/90          TIME : 13:05:32           *
                                   *****************************************************************
 NSP: LSP:AD  D: L : S  STATION              LSN RA TA TERMINAL          LINEID             LINE MODE   BPS  SW MCS
******************************************************************************************************************************
 107: 116:00  0:000:000 X25TOCDN             484 ?? ?? X25TERMINAL       X25071600           119 BIT   9600  F  SYSTEM/BNAMCS
 107: 116:01  0:001:000 BDLCMPCP061          483 ?? ?? DCDLP_BDLCTERM    BNA071601           118 BIT   9600  F  SYSTEM/BNAMCS


 NSP: LSP:AD  D: L : S  STATION              LSN RA TA TERMINAL          LINEID             LINE MODE   BPS  SW MCS
******************************************************************************************************************************
 110: 121:00  1:000:000 TB132                485 HP HP TD19200           TD102100            120 ASYN 19200  F  SYSTEM/COMS
 110: 121:00  1:000:001 TB240                486 KH KH TD19200           TD102100            120 ASYN 19200  F  SYSTEM/COMS
 110: 121:00  1:000:002 TB243                487 KK KK TD19200           TD102100            120 ASYN 19200  F  SYSTEM/COMS
 110: 121:00  1:000:003 TB245                488 KM KM TD19200           TD102100            120 ASYN 19200  F  SYSTEM/COMS
 110: 121:00  1:000:004 TB247                489 KO KO TD19200           TD102100            120 ASYN 19200  F  SYSTEM/COMS
 110: 121:00  1:000:005 TB242                490 KJ KJ TD19200           TD102100            120 ASYN 19200  F  SYSTEM/COMS
 110: 121:00  1:000:006 TB248                491 KP KP TD19200           TD102100            120 ASYN 19200  F  SYSTEM/COMS
 110: 121:00  1:000:007 TB250                492 KR KR TD19200           TD102100            120 ASYN 19200  F  SYSTEM/COMS
 110: 121:01  1:001:000 DCDLPBSC2780STA      152 ?? ?? BISYNC2780TERM    DCDLP_BSCLINE        67 SYNC  9600  F  SYSTEM/CANDE
 110: 121:02  1:002:000 PB100S10             493 ?? ?? DCDLP_AP9208TERM  AP102T02            121 ASYN 19200  F  SYSTEM/COMS
 110: 121:03  1:003:000 TB234                494 KB KB TD19200           TD102103            122 ASYN 19200  F  SYSTEM/COMS
 110: 121:03  1:003:001 TB239                495 KG KG TD19200           TD102103            122 ASYN 19200  F  SYSTEM/COMS
 110: 121:03  1:003:002 TB252                496 KT KT TD19200           TD102103            122 ASYN 19200  F  SYSTEM/COMS
 110: 121:03  1:003:003 TB254                497 KV KV TD19200           TD102103            122 ASYN 19200  F  SYSTEM/COMS
 110: 121:03  1:003:004 TB255                498 KW KW TD19200           TD102103            122 ASYN 19200  F  SYSTEM/COMS
 110: 121:03  1:003:005 TB263S10             499 ZI ZI TD19200           TD102103            122 ASYN 19200  F  SYSTEM/COMS
 110: 121:03  1:003:006 TB268                500 LJ LJ TD19200           TD102103            122 ASYN 19200  F  SYSTEM/COMS


 NSP: LSP:AD  D: L : S  STATION              LSN RA TA TERMINAL          LINEID             LINE MODE   BPS  SW MCS
******************************************************************************************************************************
***** NO STATIONS ON NSP 112 *****
```

Figure 3–1. NETWORK Option Output

```
GRAPH

                              ON-LINE DATACOM NETWORK ANALYSIS PROGRAM
                              ======= ======= ======= ======== =======

          SYSTEM/DCSTATUS (3.9.30)          DATE : 08/06/90   TIME : 13:28:53
                   DCPREFIX: MPA15C

********************           ********************           ********************           ********************
*                  *           *                  *           * LINE = 119       *           * DLS = 0:0:0      *
*     NSP 107      *************     LSP  116     ************* X25071600        ************* X25TOCDN         *
*                  *           *                  *    *      * ADAPTER = 0      *           * X25TERMINAL      *
*                  *           *                  *    *      *                  *           * LSN = 484        *
********************           ********************    *      ********************           ********************
                                                       *
                                                       *      ********************           ********************
                                                       *      * LINE = 118       *           * DLS = 0:1:0      *
                                                       ******** BNA071601        ************* BDLCMPCP061      *
                                                              * ADAPTER = 1      *           * DCDLP BDLCTERM   *
                                                              *                  *           * LSN =-483        *
                                                              ********************           ********************

********************           ********************           ********************           ********************
*                  *           *                  *           * LINE = 120       *           * DLS = 1:0:0      *
*     NSP 110      *************     LSP  121     ************* TD102100         ************* TB132            *
*                  *           *                  *    *      * ADAPTER = 0      *    *      * TD19200          *
*                  *           *                  *    *      *                  *    *      * LSN = 485        *
********************           ********************    *      ********************    *      ********************
                                                       *                                  *
                                                       *                                  *      ********************
                                                       *                                  *      * DLS = 1:0:1      *
                                                       *                                  ******** TB240            *
                                                       *                                  *      * TD19200          *
                                                       *                                  *      * LSN = 486        *
                                                       *                                  *      ********************
                                                       *                                  *
                                                       *                                  *      ********************
                                                       *                                  *      * DLS = 1:0:2      *
                                                       *                                  ******** TB243            *
                                                       *                                  *      * TD19200          *
                                                       *                                  *      * LSN = 487        *
                                                       *                                  *      ********************
```

Figure 3-2. GRAPH Option Output

```
TERMINAL
                                      ON-LINE DATACOM NETWORK ANALYSIS PROGRAM
                                      ======= ======= ======= ======== =======



                    SYSTEM/DCSTATUS (3.9.30)       DATE : 08/06/90   TIME : 13:22:18
                                 DCPREFIX: SJA15C

TERMINAL PLOTTERTERM:
COMMENTS = "";
           TYPE = TTYTYPE;
           SCREEN = TRUE;
           WRAPAROUND = FALSE;
           LINEWIDTH = 80;
           MAXINPUT = 1920;
           MAXOUTPUT = 1920;
           PAGECOUNT = 0;
           PAGESIZE = 24;
           RECEIVEDELAY = 0;
           TRANSMITDELAY = 0;
           RECEIVEADDRLENGTH = 0;
           TRANSMITADDRLENGTH = 0;

TERMINAL ADMTERM:
COMMENTS = "";
           TYPE = ADMTYPE;
           SCREEN = TRUE;
           WRAPAROUND = TRUE;
           LINEWIDTH = 80;
           MAXINPUT = 1920;
           MAXOUTPUT = 1920;
           PAGECOUNT = 0;
           PAGESIZE = 24;
           RECEIVEDELAY = 0;
           TRANSMITDELAY = 0;
           RECEIVEADDRLENGTH = 0;
           TRANSMITADDRLENGTH = 0;
```

Figure 3–3. TERMINAL Option Output

```
STATION

                              ON-LINE DATACOM NETWORK ANALYSIS PROGRAM
                              ======= ======= ======= ======== =======

           SYSTEM/DCSTATUS (3.9.30)        DATE : 08/06/90   TIME : 13:29:18
                        DCPREFIX: SJA15C

   STATION 2

        STATION  NAME = ADMSTA
        TERMINAL NAME = ADMTERM
        DCC STATION TABLE
        500140026050   WRAPAROUND : ENABLED : READY :
                       MCS = 5: LSN = 2 : WIDTH = 80
        6F020A640001   STATION REMOTE TYPE = 2: RETRY COUNT = 10: NIF INDEX = 1
        000000000000
                       PRIMARY Q = 0,CURRENT Q = 0,STN Q = 0,PSEUDOMCS = 0
        000000000000   DLS :  UNASSIGNED
   STATION 3

        STATION  NAME = ADMSTA1
        TERMINAL NAME = ADMTERM
        DCC STATION TABLE
        541040036050   WRAPAROUND : ENABLED : READY : ATTACHED :
                       MCS = 1: LSN = 3 : WIDTH = 80
        6F020A640002   STATION REMOTE TYPE = 2: RETRY COUNT = 10: NIF INDEX = 2
        840000025025   PARTICIPATES : ALL RESULTS :
                       PRIMARY Q = 37,CURRENT Q = 37,STN Q = 0,PSEUDOMCS = 0
        000000000000   DLS :  UNASSIGNED
   STATION 4

        STATION  NAME = ADMSTA2
        TERMINAL NAME = ADMTERM
        DCC STATION TABLE
        541040046050   WRAPAROUND : ENABLED : READY : ATTACHED :
                       MCS = 1: LSN = 4 : WIDTH = 80
        6F020A640003   STATION REMOTE TYPE = 2: RETRY COUNT = 10: NIF INDEX = 3
        840000025025   PARTICIPATES : ALL RESULTS :
                       PRIMARY Q = 37,CURRENT Q = 37,STN Q = 0,PSEUDOMCS = 0
        000000000000   DLS :  UNASSIGNED
```

Figure 3–4. STATION Option Output

Section 4
DUMPANALYZER

The DUMPANALYZER utility produces user-specified subsets of information from a memory dump or a program dump and analyzes that information according to parameters given by default or supplied by the user.

A Series systems have the ability of performing an enhanced memory dump. The enhanced memory dump mechanism is capable of creating a COMPLETE, ALLINUSE, or PARTIAL dump.

A COMPLETE dump is considered to be the entire memory image. An ALLINUSE dump is a refined COMPLETE dump and captures only the in-use areas (those present in memory). A PARTIAL dump is a subset of an ALLINUSE dump. By default, the PARTIAL dump includes the following areas:

- The dumping stack and all the areas owned by the stack.
- The area occupied by all the stacks in a running system, that is, the area from BOSR to LOSR of each stack.
- All the areas owned by the MCP.
- The ASD tables.
- Any stack and its associated areas that are linked into the dumping stack's GRAPHHEADWORD and PROCESSFAMILYLINK, or one of the two.
- For systems that include task control processors (TCPs), some relevant TCP state information. This includes events, P.O. boxes, and approximately 93 words of statistical data for each stack being dumped.

Multiple-processor systems and single-processor systems perform memory dumps in the same manner. When one processor of a multiple-processor system performs a memory dump, the stacks of all the processors in that system are also dumped.

The DUMPANALYZER utility is capable of analyzing all types of memory dumps. DUMPANALYZER is also capable of analyzing program dumps that were directed to disk by the TODISK program dump option.

Note that DUMPANALYZER requires the presence of the SDASUPPORT system library to help analyze both memory dumps and program dumps.

General Information

This section is intended as a reference source for experienced programmers who are familiar with the master control program (MCP). Because no single memory dump is typical, no attempt is made here to explain how a memory dump is read. This ability can be acquired only through experience and a thorough knowledge of the system software.

System types are identified within this section as follows:

Entry and Medium Systems (EMS) systems	Micro A, A 1, A 2, A 3, A 4, A 5, A 6, A 9, A 10
Host data unit (HDU) systems	A 12, A 15
Resource management module (RMM) systems	A 16, A 17

DUMPANALYZER Files

DUMPANALYZER uses three files: OPTIONS, TAPEIN, and MCPCODEFILE. File-equating these files is sometimes desirable when DUMPANALYZER is run using a Command and Edit (CANDE) language or Work Flow Language (WFL) RUN statement or using Menu-Assisted Resource Control (MARC). These files are described as follows:

OPTIONS

OPTIONS is the file name of the user input file. When DUMPANALYZER is run from an ODT, OPTIONS should be file-equated to ODT. For example:

```
RUN SYSTEM/DUMPANALYZER;FILE OPTIONS(KIND=ODT)
```

TAPEIN

TAPEIN is the file containing the memory image created by the memory dump routine in the MCP.

TAPEIN is declared as a tape file titled MEMORY/DUMP. It can also be file-equated to either of the following kinds of files:

- A disk file that is titled DP/<MMDDYY>/<HHMM>/<REASON_ID>, where MMDDYY represents month, day, and year, and HHMM represents hour and minutes. This file can be a "pseudorecovery" file created during the initialization of DUMPANALYZER, or it can be a DP file. DP files are created in the following manner. If the DN (Dump Name) system command was previously used to create a dump-to-disk file, the system directs memory dumps to the dump-to-disk file instead of to tape. After the memory dump, the DUMPDISKMASTER independent runner appears in the mix and prompts the operator to indicate whether dumps should be copied out of the dump-to-disk file into DP files. The operator can also initiate DUMPDISKMASTER at a later date with the DF (Empty Dumpdisk File) system command, and direct DUMPDISKMASTER to create the DP files at that time.
- A file created using the DUMPANALYZER *SAVE* command.

continued

continued

MCPCODEFILE	The code file of the MCP that was running at the time of the dump. This file contains the MCP names and index arrays (and might contain LINEINFO information) for analyzing stack bases, process information blocks (PIBs), and file information blocks (FIBs). The file can be successfully file-equated only to MCP code files closely related to the code file that took the dump. For example, the 3.8 version of DUMPANALYZER cannot read a 3.9 dump. This file is not required when a previously saved file is equated to TAPEIN.
PSEUDORECOVERY File	During the initialization sequence of DUMPANALYZER, the MEMORY/DUMP file is accepted as input and certain data structures are built up. When initialization is complete, this data structure information is saved in a pseudorecovery file under the user's usercode. This file has the name `DP/<MMDDYY>/<HHMM>/<REASON_ID>` MMDDYY represents month, day, and year, and HHMM represents hour and minutes. If the DUMPANALYZER run is interrupted by a halt/load or terminated by an operator, this pseudorecovery file can be file-equated to TAPEIN and used as input to DUMPANALYZER. When a pseudorecovery file is used, the MCPCODEFILE must be file-equated to the MCP code file that was in use at the time of the dump. If for any reason the user wishes to exit from the DUMPANALYZER session, considerable time and resource savings will be made if the RECESS command is used rather than the STOP or BYE command. The RECESS command does not remove the pseudorecovery file, while the other two commands do. In all cases DUMPANALYZER does not remove the file that was used as input. The user should be aware that the pseudorecovery file is removed if the SAVE command is issued and successfully completed. For advice on when to use the SAVE command, refer to the following subsection, "Saved Memory Dumps."

continued

continued

DUMPANALYZER can request exclusive access to the pseudorecovery file during initialization or while processing a SAVE command. This exclusive access is necessary to prevent any interference with other instances of DUMPANALYZER using the same pseudorecovery file. If DUMPANALYZER does have exclusive access to the pseudorecovery file and another DUMPANALYZER transaction is initiated, one of the processes must wait while the message "WAITING FOR: <pseudorecovery title>" is displayed. If this delay is not acceptable, the user can initiate the second instance with a different primary family for DISK, or by using a different usercode.

Even if DUMPANALYZER is running with FAMILY substitution in effect, the program initially ignores the FAMILY substitution when searching for MCP and program code files referenced in the dump. For those files that are not found by this method, DUMPANALYZER searches again, using FAMILY substitution.

Note that the CODEFILE FAMILY form of the CODEFILE command can be used to establish the family for all object code files referenced in the dump. If this command is used, DUMPANALYZER ignores the family names that object code files had at the time of the dump, and also ignores FAMILY substitution.

Saved Memory Dumps

If a memory dump is to be sent to Unisys for analysis, the user should first run DUMPANALYZER on the dump and use the SAVE command. In general, the SAVE command should also be used if the dump is being saved for analysis at a later date when the original MCP code file might not be available.

In the DUMPANALYZER session when the SAVE command is run, the MCP code file that is file-equated to MCPCODEFILE should be identical to the one that was running on the system when the dump was created. The SAVE command creates a *saved dump*, which contains the memory image from the original dump tape together with relevant data from the MCP code file. The saved dump is usually an extremely large file and is often stored on tape.

When DUMPANALYZER is run to analyze a saved dump file, the file TAPEIN should be file-equated to the saved dump file. The MCPCODEFILE does not have to be file-equated, and the MCP code file that produced the dump does not need to be present.

Compatibility of MCP Levels

DUMPANALYZER checks to determine the difference between the DUMPANALYZER Mark level and the MCP Mark level on the dump tape. If the levels are not the same, DUMPANALYZER terminates with an error message similar to the following, where nn is the MCP operating system Mark level and *mm* is the DUMPANALYZER Mark level:

```
CANNOT ANALYZE nn MEMORY DUMP WITH mm DUMPANALYZER
```

In some cases, DUMPANALYZER allows analysis of a wrong level MCP when the run-time option MEMONLY is chosen. The error message indicates when this is the case. It is always better to use the correct level DUMPANALYZER.

If the MCP code file equated to MCPCODEFILE does not have the same timestamp as the MCP in use at the time of the memory dump, the following error message is displayed:

```
ACCEPT: WRONG CODE FILE--OK OR RESTART
```

The operator enters OK if DUMPANALYZER should continue using the same MCP code file. If RESTART is entered, the code file is closed and DUMPANALYZER will look for a file on DISK titled MCPCODEFILE. The operator should then use the FA (File Attribute) system command to specify the desired MCP code file.

Running DUMPANALYZER

DUMPANALYZER can be run from a remote terminal, an ODT, or in batch mode. Each command is processed before the following command has been parsed.

Remote Operation

To initialize an interactive DUMPANALYZER session at a remote terminal, enter a RUN command from a CANDE or MARC session. When necessary, file-equate the input dump file to TAPEIN and the MCP that was running to MCPCODEFILE in the RUN statement. The following are examples:

```
RUN *SYSTEM/DUMPANALYZER ON MYPACK;
      FILE TAPEIN(KIND=DISK,TITLE=ER/DUMP)

R *SYSTEM/DUMPANALYZER;FILE TAPEIN(SERIALNO="10046");FILE
    MCPCODEFILE(KIND=DISK,TITLE=SYSTEM/MCP34180)
```

A MARC menu selection for running DUMPANALYZER also exists.

After the RUN statement, DUMPANALYZER displays the following messages:

```
DUMPANALYZER VERSION 37.000.00000
SELECT RUN TIME OPTIONS: PRINTER, REMOTE, DISKFILE,
 DEBUG, MEMONLY
```

The following diagram shows the syntax of the response the user can enter at this point:

```
  ┌←──────────────────────────────────────┐
──┴─┬─ REMOTE ────────────────────────────┬┴────────────────────────────┤
    ├─ PRINTER ───────────────────────────┤
    ├─ DISKFILE ── " ──<file title>── " ──┤
    ├─ DEBUG ─────────────────────────────┤
    └─ MEMONLY ───────────────────────────┘
```

Note that multiple options are separated by blank spaces, not by commas. Further, if more than one of the destination options (PRINTER, REMOTE, and DISKFILE) are included, only the last one is really used. If none of the destination options is included, REMOTE is used by default.

The following text describes the meaning of each option:

(blank line)	Transmitting a blank line has the same effect as the REMOTE option.
REMOTE	Causes the output from all DUMPANALYZER commands to be directed to the terminal.
PRINTER	Causes the output to be directed to a printer backup file.
DISKFILE "<file title>"	Causes output to be directed to the specified disk output file.
DEBUG	Causes diagnostic information related to DUMPANALYZER to be displayed.
MEMONLY	Allows a restricted analysis of memory dumps. A subset of the DUMPANALYZER commands are available in this mode. The available commands are displayed after the message "FUNCTIONS CURRENTLY AVAILABLE ARE:".

After the desired option or options are transmitted, DUMPANALYZER initializes. When DUMPANALYZER is ready to accept commands it displays the following prompt:

```
:READY
```

If DUMPANALYZER is initiated with the task equation *VALUE = 1*, then DUMPANALYZER does not prompt the user for run-time options. Instead, DUMPANALYZER assumes that only the REMOTE option is to be used.

ODT Operation

Three methods can be used to initiate DUMPANALYZER using the ODT. Only limited output is sent to the ODT, such as the HELP command information. Output from most commands is sent to the printer as soon as the session terminates. If the RELX command is in effect, information is sent to the printer during the session.

Method 1

In the first method, enter the following:

```
RUN SYSTEM/DUMPANALYZER; FILE OPTIONS(KIND=ODT)
```

When necessary, file-equate the input dump file to TAPEIN and the MCP that was running to MCPCODEFILE in the RUN statement.

The following message is then displayed:

```
INITIALIZING
---------------------------------------------------
FUNCTIONS CURRENTLY AVAILABLE ARE
```

A list of commands is then displayed, and the system displays the following message:

```
"HELP" FOR THIS LIST, "HELP HELP" FOR MORE INFO
```

DUMPANALYZER continues to initialize and then displays the following message:

```
ENTER REQUESTS
```

DUMPANALYZER commands can now be entered.

Method 2

The second method uses the DA (Dump Analyzer) system command. Refer to the *A Series System Commands Reference Manual* for a complete description of the DA command.

Enter the appropriate form of the DA command and transmit. The following message is then displayed:

```
INITIALIZING
---------------------------------------------------
FUNCTIONS CURRENTLY AVAILABLE ARE
```

A list of commands is then displayed, and the system displays the following message:

```
"HELP" FOR THIS LIST, "HELP HELP" FOR MORE INFO
```

DUMPANALYZER continues to initialize and then displays the following message:

```
ENTER REQUESTS
```

DUMPANALYZER commands can now be entered.

Method 3

If DUMPANALYZER is run on an ODT configured in data comm mode, it should be initiated using MARC. This is done by entering *??MARC*, logging on to MARC, and proceeding with the REMOTE OPERATION instructions.

Batch Operation

Batch runs of DUMPANALYZER can be initiated from Work Flow Language (WFL) jobs.

When DUMPANALYZER is initiated from a WFL job, the STATION task attribute usually defaults to a value of 0. If the STATION value is 0, then output is directed to the printer by default.

The following paragraphs outline two methods that can be used to run DUMPANALYZER from a Work Flow Language (WFL) job.

Method 1

In this method, DUMPANALYZER commands are provided in a local data specification in the WFL job. The following is an example of such a WFL job:

```
BEGIN JOB ANALYSIS;
RUN SYSTEM/DUMPANALYZER;
DATA OPTIONS
SUMMARY
LOCKS
DEADLOCK
IO UINFO NAMES ALL
OPT
MEM
TRACE
BOXINFO
MODE + ALL
STACK 123 ACTIVE SUMMARY
NAMES
AREAS AVAIL CODE LINKS
DC
?
END JOB
```

Method 2

In this method, the WFL job directs DUMPANALYZER to read commands from a disk file. The following WFL job directs DUMPANALYZER to read commands from the disk file titled DPA/INPUT:

```
BEGIN JOB ANALYSIS;
RUN SYSTEM/DUMPANALYZER;
FILE OPTIONS(KIND=DISK,TITLE=DPA/INPUT,DEPENDENTSPECS=TRUE);
FILE TAPEIN(KIND=DISK,TITLE=MEMORY/DUMP);
END JOB
```

Analyzing Program Dumps

DUMPANALYZER can be used to analyze both system dumps and program dumps. This subsection discusses the uses of DUMPANALYZER that are exclusive to program dumps.

DUMPANALYZER can analyze program dumps that have been directed to a disk file. Many of the DUMPANALYZER commands that are used for system dumps can be used for program dumps. The output from these commands is limited to the information available in the program dump file. For example, the SUMMARY command produces a listing of only dumped stacks.

The following DUMPANALYZER commands can be used when a program dump is analyzed:

ARRAYLIMIT	ASDNUMBER
BYE	DEBUG
FIB	GRAPHS
HDR	HEADING
HEAP	HEAPSTACK
HELP	IOCB
KEEP	LIB
LINKS	LOADXREF
MASK	MD
MIX	MODE
MSCW	NAMES
OPT	PATTERN
PIB	PORT
PRINTARRAY	PRINTCODE
PRINTER	PRINTVAL
PROCSTACKS	PROGRAMDUMP
RECESS	RELEASE
RELX	REMOTE
REPEAT	RIGHTJUST
SEARCH	STACK
STACKWINDOW	STOP
SUBPORT	SUMMARY
USE	WHERE
WHO	

DUMPANALYZER still displays identifier names for either the MCP or the user environments when analyzing a program dump. Note that the SAVE command is not available for use with program dumps. Therefore, the relevant code files must be present at the time of the analysis.

Input to DUMPANALYZER

The following subsection, "Basic Constructs," describes the syntactic variables commonly used in the addressing and value schemes specified in various DUMPANALYZER commands. "DUMPANALYZER Commands" in this section provides detailed descriptions of each of the commands that can be used as input to DUMPANALYZER.

Basic Constructs

The addressing and value schemes used in the syntax diagrams of DUMPANALYZER commands commonly employ certain basic syntax constructs.

<number>

```
──┬─<hexadecimal number>───┬──────────────────────────────────────────┤
  ├─ DEC ──<decimal number>─┤
  └─ OCT ──<octal number>───┘
```

The following text describes the meaning of each variable:

<hexadecimal number>	A number in base 16, each of whose digits ranges from 0 to F. In DUMPANALYZER commands, <number> is assumed to be hexadecimal unless it is preceded by a prefix such as DEC (decimal) or OCT (octal), indicating another base.
<decimal number>	A number in base 10, each of whose digits ranges from 0 to 9. In DUMPANALYZER commands, <number> is decimal when it is preceded by the prefix DEC.
<octal number>	A number in base 8, each of whose digits ranges from 0 to 7. In DUMPANALYZER commands, <number> is octal when it is preceded by OCT.

Simple Address

A <simple address> represents a location in memory. One of the common uses of the <simple address> construct is in the MD command, described later in this section.

A <simple address> is made up of an <absolute address> or a <simple location> followed by an optional offset. Three types of <simple location>s exist: stack-related, global, and indirect. The <offset> is a number that indicates the displacement of the <simple address> from the given <simple location> or <absolute address>.

```
<simple address>

──┬─<absolute address>─┬─┬───────────────┬───────────────────────────┤
  ├─<simple location>──┘ ├─ + ─┬─<offset>─┤
  │                      └─ - ─┘          │
  └─<stack ID>─────────────────────────────┘

<absolute address>

──<number>────────────────────────────────────────────────────────────┤

<simple location>

──┬─<stack ID>─┬─<stack offset>───────────────────────────────────────┤
  │            └─ BASE ─┬─<attribute name>──┤
  │                     └─ # ──<offset>─────┤
  ├─ PIB ──<stack number>─┬─<attribute name>─┤
  │                       └─ # ──<offset>────┤
  ├─ G ─┬─<global ID>────────────────────────┤
  │     └─ # ──<offset>──────────────────────┤
  ├─ RV ──<simple value>─────────────────────┤
  └─ VIA ──<ASD number>──────────────────────┘

<stack ID>

──┬─ STK ──<hexadecimal stack number>─┬─┬────────┬────────────────────┤
  └─ @ ───────────────────────────────┘ └─ SD ───┘

<stack offset>

──┬─ LOSR ─┬──────────────────────────────────────────────────────────┤
  ├─ BOSR ─┤
  ├─ SREG ─┤
  └─ FREG ─┘

<offset>

──<number>────────────────────────────────────────────────────────────┤
```

Explanation

<absolute address>

A hexadecimal, decimal, or octal number that specifies an address within a present, online memory module. The prefixes DEC and OCT are required if decimal or octal numbers are specified. The minimum valid address is 0. The maximum valid address is determined by the memory capacity of the system. For example, on a Micro A system with 2 megawords of memory, the command MD 1FFFFF is valid.

<simple location>

Specifies locations in memory that are stack-relative, global identifiers, or indirect.

The following groups of variables represent valid <simple location>s; expansions of some of the variables within those groups are included:

continued

continued

<stack ID>

A <simple location> can consist of a <stack ID>. A <stack ID> specifies a stack by number or uses the last stack explicitly referenced. An expansion of the variable <stack ID> follows:

```
STK <hexadecimal stack number>
STK <hexadecimal stack number> SD
@
@ SD
```

These four statements are all expansions of the variable <stack ID>. STK indicates that a stack is involved. <hexadecimal stack number> specifies the hexadecimal number that identifies the stack. The last stack explicitly referenced is represented by the *at* sign (@). Referencing a stack by its hexadecimal number sets up the @ for subsequent use. The segment dictionary of the stack can be referenced if the stack number or the *at* sign (@) is followed by SD.

<stack ID> <stack offset>

A <simple location> can consist of a <stack ID> (as defined previously), followed by a specific location within the stack, which is called the <stack offset>. An expansion of the token <stack offset> follows:

- LOSR = Limit-of-stack register
- BOSR = Bottom-of-stack register
- SREG = S register
- FREG = F register for environment stacks (that is, stacks that are not segment dictionaries).

<stack ID> BASE <attribute name>

<stack ID> BASE <offset>

PIB <stack number> <attribute name>

PIB <stack number> <offset>

For some other possible <simple location>s, the stack BASE and process information block (PIB) of the stack can also be used; a <stack ID> (with BASE) or <stack number> (with PIB) must be specified. In addition, either an <offset> or an attribute name must be identified. The <attribute name>s are listed in the MCP symbolic; they change with each new release. Since the availability of the cell names depends on the presence of the MCP code file, these cell names are not valid if the task (PIB) and stack index arrays cannot be generated.

G <global ID>

G <offset>

These are <simple location>s that are global IDs (operating system D[0] cells). The G denotes that the location is global. The <offset> indicates a displacement of a given number of words away from D[0].

continued

continued

RV <simple value>	Addresses can also be indirectly specified by using the RV (the reference value), option. For example, if a cell contains an absolute address, an indirect reference word (IRW), or a stuffed indirect reference word (SIRW), the RV option allows the contents of this cell to be used as an indirect address. (An IRW would require that an environment be set up, but an SIRW would not; a set up environment implies that a stack has already been referenced.) An absolute address would be used as is. The actual value that can be used for indirect addressing is a <simple value>, which is defined later in this subsection. Each level of indirection is printed as it occurs so that each chain of addresses can be seen along with the referenced data.
VIA <ASD number>	A <simple location> pointed to by an <ASD number>. The <ASD number> can be any number from 1 through the largest valid actual segment descriptor (ASD) that is specified in ASD1[0].
+ <offset>	
– <offset>	An <offset> is a hexadecimal, decimal, or octal number that specifies the number of words away from a reference point such as a <simple location>, that a particular address is located. If an <offset> is specified in decimal form, DEC should precede it. If an <offset> is specified in octal form, OCT should precede it. The offset either increases (+) or decreases (–) the address.

Examples

In the following examples, blank spaces and special characters function as delimiters. Delimiters are needed whenever two alphanumeric items are juxtaposed.

```
41AC0                      %ABSOLUTE ADDRESS
DEC 47990                  %ABSOLUTE ADDRESS
OCT 170071                 %ABSOLUTE ADDRESS
STK 4A BASE LOCKCOUNT      %BASE ADDRESS (STACK-RELATED)
PIB 2E7 ASDSINUSE          %PIB ADDRESS (STACK-RELATED)
PIB 2E7 # 5F               %PIB ADDRESS (STACK-RELATED)
@ SD LOSR                  %INVARIANT STACK-RELATED ADDRESS
G HLUNIT                   %GLOBAL IDENTIFIER ADDRESS
RV M[47AB]                 %INDIRECT ADDRESS
```

Multiple Addresses

The <multiple addresses> variable provides for the specification of more than one address. The addresses generated by these expressions are treated as sequences; that is, the first address in the sequence is generated and passed to the execution routine that requires it. Then, the next address is generated, and so forth, until the list is exhausted. The addresses are generated in the proper order, with memory addresses increasing, except in the case of stack-related addresses, in which case, the addresses are generated from LOSR to BOSR. These rules hold true even when the specification by the user differs from the proper order.

<multiple addresses>

```
  ┌←──────────────── , ─────────────┐
──┴<simple address>─┬───────────────┴──────────────────────────┤
                    └<until part>──┘
```

<until part>

```
──┬─ FOR ─┬─<simple value>─┬─────────────────────────────────┤
  │       └─ ALL ──────────┤
  └─ TO ──┬─<offset>───────┤
          ├─<stack offset>─┤
          └─ END ──────────┘
```

The following text describes the description of each variable:

<until part>	The <until part> specifies over what range a list of <multiple addresses> should extend, from an initial location.
FOR ALL	Indicates that all the addresses from the initial location to the end of the area referenced by the descriptor that provided the initial location are to be included as part of the <multiple addresses>.
FOR <simple value>	Here the <simple value> indicates the number of consecutive addresses to be included in the <multiple addresses> list. <simple value> is defined later in this subsection.
TO <offset>	Here the <offset> indicates the displacement of the highest address to be included in that part of the <multiple addresses>.
TO <stack offset>	Here the <stack offset> indicates that all addresses up to the indicated location in the stack (BOSR, LOSR, SREG) are to be included as part of the <multiple addresses>.
TO END	Indicates that all the addresses from the initial location to the end of the area referenced by the descriptor that provided the initial location are to be included as part of the <multiple addresses>.

Note: *Only one <absolute address> per statement is allowed. Multiple stack references must reference the same stack.*

Examples

The following are examples of possible <multiple addresses>. Each example uses a different form of the <until part>.

3BAC FOR 20	Specifies the region starting at absolute address 3BAC and extending for 20 words.
3BAC TO 3BCB	Specifies the region extending from absolute address 3BAC to absolute address 3BCB.

continued

continued

STK 32 LOSR TO BOSR	Specifies the region extending from the bottom of stack address (BOSR) for stack 32 to the limit of stack address (LOSR) for the same stack. DUMPANALYZER ignores the fact that LOSR and BOSR appear in reverse order in the address.
STK 11C BASE BDINFO FOR ALL	Specifies the entire region referenced by the BDINFO word in process stack 11C.

Simple Value

A <simple value> is a single scalar value either defined by the user or derived from a value in a memory location or in a stack. A <simple value> is a <word value> followed by an optional <concatenation> value. The <simple value> construct is often used in the PV command.

```
<simple value>

—<word value>—┬───────────────────────────┬──────────────────────┤
              │ ┌←─────────────────────┐  │
              └─┴─ & —<concatenation>──┴──┘

<word value>

—<simple word>─┬──────────────────────┬─────────────────────────┤
               └─ . —<partial word>───┘

<simple word>

──┬─ M — [ —<hexadecimal address>— ] ────────────┬───────────────┤
  ├─ C — ( ─┬─<simple location>──┬─ ) ───────────┤
  │         └─<absolute address>─┘               │
  ├─<number>─────────────────────────────────────┤
  ├─<simple location>──┬─ [ —<simple index list>— ] ─┤
  ├─<absolute address>─┘                         │
  └─ ( —<simple value>— ) ───────────────────────┘

<simple index list>

  ┌←──────── , ───────┐
──┴─<simple value>────┴─────────────────────────────────────────┤

<partial word>

──┬─ [ —<simple value>— : —<simple value>— ] ─┬─────────────────┤
  └─ TAG ─────────────────────────────────────┘
```

```
<concatenation>

——<word value>————————————————————————————————————————→

→—┬— [ —<left bit to>— : —<left bit from>— : —<bit count>— ] —┬—┤
  ├— [ —<left bit to>— : —<bit count>— ] ——————————————————————┤
  └— TAG ——————————————————————————————————————————————————————┘

<left bit to>

——<simple value>——————————————————————————————————————┤

<left bit from>

——<simple value>——————————————————————————————————————┤

<bit count>

——<simple value>——————————————————————————————————————┤
```

The following text describes the meaning of each variable:

<word value>	A <word value> is a <simple word> with an optional <partial word>. The following groups of tokens represent valid <simple word>s; expansions of these tokens are also given.
M [<simple address>]	*M* signifies memory. The contents of the memory location at <hexadecimal address> is one type of <simple word>. This operation is the same as subscripting the MEMORY array.
C (<simple location>)	
C (<absolute address>)	*C* signifies contents. The C option can be used to obtain the contents of any <simple location> or <absolute address>. <simple location> and <absolute address> are defined under <simple address>, one of the preceding items in "Basic Constructs."
<number>	This construct is defined under "Basic Constructs," earlier in this section. ***Note:*** *A <number> is assumed to be hexadecimal unless preceded by DEC or OCT. For example, to indicate bit 47 for 6 in a <partial word>, either of the following constructs could be used:* [2E:6] *or* [DEC 47:6]. *Similar considerations apply to the <concatenation> construct.*
<simple location> [<simple index list>]	
<absolute address> [<simple index list>]	A <simple location> or <absolute address> can be indexed, and the derived value can be used as a <simple word>. This method of forming values is valid only if the word specified (which can be reached through an IRW chain) is an unindexed data descriptor, and the number of indexes specified matches the number of dimensions in the array.
<partial word>	A <partial word> is an optional component of a <word value>. When a partial word is present, the <word value> is the value of a selected group of bits within the <simple word>. <partial word> specifies a particular group of bits within the <simple word>.

continued

continued

<simple value>:<simple value>	Two <simple value>s separated by a colon (:) make up a <partial word>. The first <simple value> indicates the number of the starting bit in a range. The second <simple value> indicates how many bits the range extends over (heading from 47 down to 0).
TAG	When TAG is specified as the <partial word>, then <word value> is the value of the 4-bit TAG in the attached <simple word>.
<concatenation>	The <simple value> consists of a <word value> & <concatenation>. In the context of concatenation, <word value> is a 48-bit, binary word. The <concatenation> variable includes within itself a second <word value>, along with bit specifiers and a bit count; the specifiers and count indicate a substitution of certain bits to be made from the second word value into the first. For further information about bit manipulation, see the *A Series ALGOL Programming Reference Manual, Volume 1: Basic Implementation.*
<word value> [<left bit to>:<left bit from>:<bit count>]	Here, <word value> represents a binary, 48-bit word with a 4-bit tag value. The word value in the <concatenation> is the source word, while the word value that preceded the ampersand (&) is the destination word. The <left bit to> variable defines the highest (ranging from 0 through 47) bit number in the destination word. The <left bit from> variable defines the highest (ranging from 0 through 47) bit location in the source word. The <bit count>, ranging from 1 through 48, specifies the length of the data field to be moved from the source word to the destination word.
<word value> [<left bit to>:<bit count>]	The <word value> variable represents a binary, 48-bit word with a 4-bit tag value. The <left bit to> variable defines the highest (ranging from 0 through 47) bit number in the destination word. In this case, the <left bit from> specification is understood to coincide with the <left bit to> specification. Again, <bit count> specifies the length of the data field to be moved from the source word to the destination word.
<word value> TAG	Here <word value> represents a binary, 48-bit word with the preceding tag value. TAG indicates that the TAG of the source word should be substituted for the TAG in the destination word.

Examples

In the following examples, blank spaces and special characters function as delimiters. Delimiters are needed whenever two alphanumeric items are juxtaposed.

M[47AC]	%MEMORY LOCATION
C (G HLUNIT)	%CONTENTS OF SIMPLE LOCATION
DEC 123456 & 3 TAG	%CONCATENATION
C (STK 53 BOSR).[6:2]	%PARTIAL WORD

DUMPANALYZER Commands

The following text describes the commands that can be used when running DUMPANALYZER. These commands allow the user to select the type of analyses to be done on the dumped data. Multiple commands separated by semicolons (;) can be entered on the same line.

ALLPORTS

The ALLPORTS command displays information about all the port files in the dump. This command is valid only for dumps in which the running BNA version was BNAV2.

```
— ALLPORTS ————————————————————————————————————————————|
```

Because this command in most cases produces many thousands of lines of output, it should be used sparingly.

If the global PORT_ARRAY_DESC value is zero, indicating that either no ports were set up or BNAV1 was in use, an appropriate negative response is returned.

The output from the ALLPORTS command is similar to the output from the PORT command, except that ALLPORTS displays all ports instead of just one. An example of the PORT output is given in the PORT command description later in this section.

AREAS

The AREAS command prints the contents of memory areas. The command allows the user to specify what areas of the dump are to be analyzed and what information is to be displayed.

In general, all memory can be analyzed. However, in split code and data environments on HDU systems, only the data environments are analyzed.

<areas>

```
          ┌←──────────────────────────────────────┐
— AREAS ──┴──┬─ AVAIL ──────────────────────────┬─┴─<output selection>──|
             ├─ DATA ───────────────────────────┤
             ├─ CODE ───────────────────────────┤
             ├─/1\─ SIZE ──<number>─────────────┤
             ├─/1\─ ODDBALL ──<oddball field>───┤
             └─/1\─ RANGE ──<multiple address>──┘
```

<output selection>

```
  ┌←─────────────────────┐
──┴─/3\─┬─ CONTENTS ─┬───┴────────────────────────────────────|
        ├─ LINKS ────┤
        └─ STATS ────┘
```

```
<oddball field>

──┬─ BOXINFOAREA ──────┬──────────────────────────────────────┤
  ├─ BUFFERHEADER ─────┤
  ├─ CODEAREA ─────────┤
  ├─ DCQAREA ──────────┤
  ├─ DIRECTIOBUFF ─────┤
  ├─ DOPEOREVENTAREA ──┤
  ├─ DOPEVECT ─────────┤
  ├─ EVENTARRAY ───────┤
  ├─ FIBMARK ──────────┤
  ├─ ICMMARK ──────────┤
  ├─ INSTACKAREA ──────┤
  ├─ IOCBAREA ─────────┤
  ├─ IOCDAREA ─────────┤
  ├─ MESSAGEAREA ──────┤
  ├─ MESSAGEMOMAREA ───┤
  ├─ NORMALAREA ───────┤
  ├─ PERMSAVEAREA ─────┤
  ├─ PIBMARK ──────────┤
  ├─ PROCINFOAREA ─────┤
  ├─ PROCREFARRAY ─────┤
  ├─ RESERVEEVENTAREA ─┤
  ├─ RESERVEAREA ──────┤
  ├─ SAVECODEAREA ─────┤
  ├─ SEGDOPE ──────────┤
  ├─ SEGSEG ───────────┤
  ├─ SIBMARK ──────────┤
  ├─ SORTAB ───────────┤
  ├─ STACKMARK ────────┤
  ├─ STALISTAREA ──────┤
  ├─ SUBSPACE_ARRAY ───┤
  ├─ TABMARK ──────────┤
  └─ UINFOAREA ────────┘
```

The following text describes the meaning of each variable:

AREAS <output selection>	All memory areas that contain data or code, or that are available are analyzed.
AREAS AVAIL <output selection>	All memory areas that are available are analyzed.
AREAS DATA <output selection>	All memory areas that contain data are analyzed.
AREAS CODE <output selection>	All memory areas that contain code segments are analyzed.
AREAS SIZE <number> <output selection>	All memory areas of the designated size are analyzed.
AREAS ODDBALL <oddball field> <output selection>	All memory areas that contain the item designated in the oddball field are analyzed. The items that can be specified as an oddball field are defined as follows:

BOXINFOAREA	The area is part of the BOXINFO array.
BUFFERHEADER	The area contains information about input/output (I/O) buffers.
CODEAREA	The area contains code.

continued

continued

DCQAREA	The area is a DCQUEUEHEAD area.
DIRECTIOBUFF	The area contains a direct array.
DOPEOREVENTAREA	The area is a bad event.
DOPEVECT	The area contains a dope vector; this dope vector includes mom descriptors.
EVENTARRAY	The area contains an event array.
FIBMARK	The area contains a file information block (FIB).
ICMMARK	The area is a connection block.
INSTACKAREA	The area is an in-stack array.
IOCBAREA	The area contains an I/O control block (IOCB).
IOCDAREA	The area is for I/O control direct (IOCD) use.
MESSAGEAREA	The area is a message.
MESSAGEMOMAREA	The area contains message MOMS.
NORMALAREA	The area is a "normal" area that does not contain any of the structures indicated in the rest of the option list.
PERMSAVEAREA	The area is a permanent save area allocated by GETITGOING.
PIBMARK	The area contains a process information block (PIB) or task variable, or a segment dictionary pseudo-PIB.
PROCINFOAREA	The area is part of the PROCINFO array.
PROCREFARRAY	This area contains a procedure reference array.
RESERVEAREA	The area is for the GETAREA reserve pool.

continued

continued

RESERVEEVENTAREA	The area is space held in reserve for the task control unit (TCU). The RESERVEEVENTAREA is passed to the TCU by the TCP_INSTALL_EVENT_SPACE call when the TCU indicates it has no unused EVENTs during a TCP_ALLOCATE_EVENT call. A small pool of the areas is maintained by RESERVE_EVENT_ AREA_MANAGER. RESERVEEVENTAREAs are currently used only on A 16 systems.
SAVECODEAREA	This area contains save code.
SEGDOPE	The area contains the dope vector for a segmented (that is, "paged") array.
SEGSEG	The area is a page of a segmented array.
SIBMARK	The area contains a DMSII SIB created on Mark 3.5 or later releases.
SORTAB	The area contains a sort table listing absolute addresses.
STACKMARK	The area contains a stack or a segment dictionary.
SUPSPACE_ARRAY	The area is a type of array subset that is treated as a complete unit for MASKSEARCH operations and primitive memory allocation routines.
STALISTAREA	The area contains a station list.
UINFOAREA	The area is part of the UINFO array.

AREAS RANGE <multiple address> <output selection> — The analysis of memory areas is restricted to the specified address range. Areas partially or completely in the desired range are analyzed.

<output selection> — At least one of the following options must be designated:

CONTENTS	The contents of all the selected areas are displayed.

continued

continued

LINKS	The links of all the selected areas are displayed.
STATS	Statistics regarding the memory areas and their associated stacks are collected. These statistics are reported in a memory usage summary.

Example

Figure 4–1 shows example output from the AREAS command.

```
INPUT: AREAS RANGE 2100 TO 5000 LINKS

000020F5  INUSE AREA  DATA: LENGTH=01C20(7200)MOM ADDR=0001DB96 (THIS IS STACK 012) -------------------------(STACK)      ---------
LINKS: 6 E01C24 000012    7 000000 000000    3 000000 000000   ... 3 E01C24 000000

          RESCODE MEMORY
00003D19  INUSE AREA  DATA: LENGTH=04209(16905)MOM ADDR=000028AD (D[0] + 07AD) -----------------------------(PERMSAVEAREA)---------
LINKS: 6 E0420D 0010D5    7 000000 000000    3 000000 000000   ... 3 E0420D 000000
```

Figure 4–1. AREAS Command Output

ARRAYLIMIT

The ARRAYLIMIT command limits the size of each array printed to the number of lines specified by the positive <decimal number>.

```
<arraylimit>

— ARRAYLIMIT —<decimal number>——————————————————————————————————|
```

If the <decimal number> is O, the printing of arrays is suppressed.

For multidimensional arrays, the limit is applied cumulatively for both the dope vector entries and the data entries.

ASDNUMBER

The ASDNUMBER command prints out information about ASDs.

```
<asdnumber>

— ASDNUMBER —┬—<number>——┬———————————————————┬—┬——————————┬—————————————|
             |           ├— UNTIL <number> ——┤ └— EXPAND —┘
             |           └— FOR <number> ————┤
             └— STACK —<number>—┬————————————┤
                                └— VIRGIN ———┘
```

The following text describes the meaning of each variable:

ASDNUMBER <number>	This form of the command prints out ASD1 through ASD4 for the specified ASD number.
ASDNUMBER <number> UNTIL <number>	This form of the command prints out ASD1 through ASD4 for the specified range of ASDs.
ASDNUMBER <number> FOR <number>	This form of the command prints out ASD1 through ASD4 for the number of ASDs specified starting at the designated ASD number.
ASDNUMBER STACK <number>	This form of the command prints a list of the ASDs for the specified stack.
ASDNUMBER STACK <number> VIRGIN	This form of the command prints a list of the ASDs associated with the specified stack that have been referenced but have never been allocated an area in main memory for the array.
EXPAND	When the EXPAND parameter is used, the ASD 1 through ASD 4 information is printed in greater detail.

Example

The following is an example of the output from the ASDNUMBER command:

```
INPUT: ASDNUMBER 33 FOR 2 EXPAND
ASD1[ØØØ33]: [ØØØ4ØØ37] 5 63ØØØØ Ø22418
   ASD1_PRESENTTOPROCF      =  1
   ASD1_UNALTEREDF          =  Ø
   ASD1_NOTSTACKF           =  Ø
   ASD1_PRESENTFORIOF       =  1
   ASD1_NOTPAGEF            =  1
   ASD1_READONLYF           =  Ø
   ASD1_DONTRESIZEF         =  Ø
   ASD1_PAGEDF              =  Ø
   ASD1_ADDRESSF            =  ØØØ22418

ASD2[ØØØ33]: [ØØØ68F92] 3 ØAØØØØ ØØØ192
   ASD2_SPACEUSAGEF      =  ØA  (STACK)
   ASD2_LENGTHF          =  ØØ192

ASD3[ØØØ33]: [ØØØ91EED] 5 EØØØØ3 3Ø3149
  INDEX                 = ØØØ33
  ASDINDEX              = Ø3149

ASD4[ØØØ33]: [ØØØBAE48] Ø ØØØ1D8 ØØ33ØØ
   ASD4_STACKSIZEF          =  ØØØ1D8
   ASD4_OLAYSTATEF          =  Ø (NON OVERLAYABLE SEG)
   ASD4_OWNERSTACKF         =  Ø33
   ASD4_VIRGINASDF          =  Ø
   ASD4_MARKEDFORWSSHERIFFF =  Ø
   ASD4_UNLOCKDISPOSALF     =  Ø
   ASD4_LOCKBITF            =  Ø

TBIT_RCWS[ØØØ33]: Ø ØØØØØØ ØØØØØØ

ASD LOCK STACK              =  Ø14

ASD1[ØØØ34]: [ØØØ4ØØ38] Ø ØØØØØØ ØØØØØØ
   ASD1_PRESENTTOPROCF      =  Ø
   ASD1_UNALTEREDF          =  Ø
   ASD1_NOTSTACKF           =  Ø
   ASD1_PRESENTFORIOF       =  Ø
   ASD1_NOTPAGEF            =  Ø
   ASD1_READONLYF           =  Ø
   ASD1_DONTRESIZEF         =  Ø
   ASD1_PAGEDF              =  Ø
   ASD1_ADDRESSF            =  ØØØØØØØØ

ASD2[ØØØ34]: [ØØØ68F93] 6 ØAØØØØ ØØØE28
   ASD2_SPACEUSAGEF      =  ØA  (STACK)
   ASD2_LENGTHF          =  ØØE28

ASD3[ØØØ34]: [ØØØ91EEE] Ø Ø144ØF BØ11Ø2
ASD4[ØØØ34]: [ØØØBAE49] 6 ØØØFCC ØØ12Ø1
   ASD4_STACKSIZEF          =  ØØØFCC
   ASD4_OLAYSTATEF          =  Ø (NON OVERLAYABLE SEG)
   ASD4_OWNERSTACKF         =  Ø12
```

ASDTABLEBASE

The ASDTABLEBASE command displays the base of the ASD table or specifies a new <simple address> for future computations.

```
<asdtablebase>

— ASDTABLEBASE ─┬──────────────────┬──────────────────────────────┤
                └─<simple address>─┘
```

The following text describes the meaning of each variable:

ASDTABLEBASE	This form of the command displays the base of the ASD table.
ASDTABLEBASE <simple address>	This form of the command sets the base of the ASDTABLE to the specified <simple address> for future computations.

Example

The following is an example of the output from the ASDTABLEBASE command:

```
INPUT: ASDTABLEBASE
THE BASE OF THE ASD TABLE=  00040004
```

ASN

Note: *The ASN command has been deimplemented. This command had meaning only on systems with ASN memory, which is not supported on Mark 3.9 and later system software releases.*

BOXINFO

The BOXINFO command prints the box information array.

```
<boxinfo>

— BOXINFO ─┬───────────────────────┬──────────────────────────────┤
           ├─<BOXINFO cell name>───┤
           └─ # ──<offset>─────────┘
```

The following text describes the meaning of each construct:

BOXINFO	Analyzes and prints the contents of the MCP array BOXINFO. The BOXINFO array contains global information such as the core utilization at the time of the dump.
BOXINFO <BOXINFO cell name>	
BOXINFO # <offset>	The word with the requested cell name or offset is printed.

Example

Figure 4–2 shows example output from the BOXINFO command.

```
INPUT: BOXINFO

BOX INFO DUMP

BOXINFO =      5 800003 103160

00      0 000000 000000     BOXINFOZERO
01      5 800001 103161     MA
02      5 800001 103163     SMA
03      5 800001 103162     PMA
04-07   0 000000 000000     ** UNKNOWN WORD **AVAILCORE          OLAYCORE           SAVECORE
08      0 000000 000F89     SCHEDCORE         = 4025
09      0 000000 000000     NOMEMWORD
0A      0 000000 0002E3     CCOLAY            = 739
0B      0 000000 0109B7     PCOLAY            = 68023
0C      0 000000 000C3F     OCOLAY            = 3135
0D      0 000000 00179C     COOLAY            = 6044
0E      0 000000 0007DA     CLOLAY            = 2010
0F      0 000000 02FEA2     CCWORDS           = 196258
10      0 000000 D9648A     PCWORDS           = 14247050
11      0 000000 03F7E4     OCWORDS           = 260068
12      0 000000 072871     COWORDS           = 469105
13      0 000000 072291     CLWORDS           = 467601
14      0 000000 000002     OLAYCHANNELS
15      0 000000 000000     OLAYCLOCKS
16      0 000000 25D56C     BATCHOLAYDISKSIZE = 2479468
17      0 000000 000000     SUSPENDEES
18      0 000000 800000     MEMMAX
19      0 000000 020000     MODSIZE
1A      2 000000 100140     ETERNALEVENT      (WAITING: STK 014)
1B      2 000000 000000     (SALADINPLACE2)
1C      0 000000 000000     ETERNALHEAD
1D      0 000000 000002     ONEIRPERBOX
1E      5 800000 603165     ** UNKNOWN WORD **
1F-20   0 000000 000000     HLDDINFO          ** UNKNOWN WORD **
21      0 000000 253160     BATCHPBITTIME     =  5.8499328 SECONDS
22      0 000000 001052     BATCHPBITCOUNT    = 4178
23      2 219687 8FFE00     INITPBITTIME      = 453.594314 SECONDS
24      2 0000B6 6607E1     INITPBITCOUNT     = 3060140001
25-26   0 000000 000000     FREEPBITTIME      FREEPBITCOUNT
27      0 000000 000088     BATCHTASKS        = 136
28      0 0000A2 C24FFA     READYTIME         = 6553.54428 SECONDS
29      0 000000 000000     SCHEDPROC
2A      0 000000 000030     PROCMASK
2B      0 000000 00008F     INTRINSICSTKNUM
2C-2D   0 000000 000000     PLISUPPORTSNRS    BASICSUPPORTSNRS
2E      0 000000 000043     GENERALSUPPORTSNR
2F-30   0 000000 000000     CODEIOCWS         CODEIOCWSAVAILABL
```

Figure 4–2. BOXINFO Command Output

BYE

The BYE command terminates DUMPANALYZER. A synonym for BYE is STOP.

```
<bye>

— BYE ——————————————————————————————————————————————————|
```

CAND

The CAND command analyzes a "candidate" for a port-matching structure. The candidate is a subport that has been offered for interprocess communication and has not yet found a matching subport. (Refer to the example under "PORT.") The CAND command is only valid for BNA Version 1.

```
<cand>

— CAND ─┬─<index>────────────┬──────────────────────────|
        └─ AT ─<address>─────┘
```

The following text describes the meaning of each construct:

CAND <index>	The subport with the indicated <index> is analyzed. The index is found in the library info word of the file information block (FIB).
CAND AT <address>	The subport at the given <address> is analyzed.

For more information, see the discussion of the PORT command in this section.

CB

The CB command displays the connection block (CB) at the specified <simple address>. If the <simple address> is valid and the word at <simple address> + 1 is a valid CB identification word, then the memory area is analyzed as a connection block.

```
— CB ─┬──────┬─<simple address>─┬───────────────────────|
      ├─ AT ─┘                  │
      └─ VIA ─<ASD number>──────┘
```

The following text describes the meaning of each construct:

CB <simple address> CB AT <simple address>	The connection block at the specified address is analyzed.
CB VIA <ASD number>	The connection block addressed through the specified ASD entry is analyzed.

An alternative to the CB command is the CB option of the MODE command. The CB option causes connection blocks to be analyzed wherever they occur in the output from other commands.

Example

Figure 4–3 shows example output from the CB command.

```
INPUT: CB AT 269A5C

---- CONNECTION BLOCK ----
0 000000 000003 : STATUS (STATE=CONNECTED)
1 103296 4C0000 : SIRW TO ACTIVATION RECORD MSCW
3 C3C26D C9C4F2 : IDENTIFICATION
0 300002 300002 : CB=2/3,CBC=2/3
0 000000 000002 : ACCESS CODE WORD
0 000000 70000F : TEMPLATE
0 001000 05000B : FORMAT (FIXED=0010,LINKS=0005,AR=000B)
0 000000 000000 : GONE COUNT=0
2 000000 000000 : DISCONNECTION EVENT1
2 000000 000000 : DISCONNECTION EVENT2
0 000012 000012 : MY OWNER STACK = 0012, HIS OWNER STACK = 0012
0 000121 544FA3 : MY REFERENCE
0 000121 537E62 : HIS REFERENCE
5 800000 C32964 : ACTIVATION RECORD MOM
0 000000 000000 : COUNT LOCK (NOT LOCKED)
0 000000 000001 : COUNT = 1
--- LINKS TO PCWS ---
1 103296 4C0007 :  POINTS TO 0008 (0001+0007) IN ASD 32964
1 103296 4C0008 :  POINTS TO 0009 (0001+0008) IN ASD 32964
1 105062 BC000A :  POINTS TO 000B (0001+000A) IN ASD 5062B
1 103296 4C000A :  POINTS TO 000B (0001+000A) IN ASD 32964
1 103296 4C0009 :  POINTS TO 000A (0001+0009) IN ASD 32964
----- ACTIVATION RECORD ANALYSIS -----
(00,000B)  3 E00010 045C43 : *** NOT ALLOWED ***
(00,000A)  7 000493 B0951C : PCW: LL=2, D[0] SEGMENT @ 151C:093B:2, NORML STATE
(00,0009)  7 00002B D09514 : PCW: LL=2, D[0] SEGMENT @ 1514:02BD:0, NORML STATE
(00,0008)  7 0000C3 30950F : PCW: LL=2, D[0] SEGMENT @ 150F:0C33:0, NORML STATE
(00,0007)  7 00011B E0950E : PCW: LL=2, D[0] SEGMENT @ 150E:11BE:0, NORML STATE
(00,0006)  7 000000 909517 : PCW: LL=2, D[0] SEGMENT @ 1517:0009:0, NORML STATE
(00,0005)  0 000000 000001 :
(00,0004)  0 000000 000000 :
(00,0003)  0 000000 000002 :
(00,0002)  0 000000 000001 :
(00,0001)  3 2100C8 00550E : RCW: LL=1, NORMAL STATE
(00,0000)  3 012000 844002 : MSCW: D[08]=0000 IN STACK 012
           5 C00001 544FA3 : CB REFERENCE
```

Figure 4–3. CB Command Output

CODEFILE

The CODEFILE command specifies either the code file or disk family name to be used when DUMPANALYZER requires a code file. This command applies only to code files that are not MCP code files.

By default, DUMPANALYZER searches for code files with the same file name and family name as on the dumping system, but sometimes this gives erroneous results (such as misleading line information). The CODEFILE command can be used to specify a correct code file for DUMPANALYZER to use.

```
<code file>

— CODEFILE ─┬──────────────────────────────────────────────┬──────────┤
            ├─<seg dict stack #>— " —<code file title>— " ─┤
            └─ FAMILY —<family name>───────────────────────┘
```

The following text describes the meaning of each construct:

<seg dict stack #>	A segment dictionary stack number from the dump.
<code file title>	A code file title if it is different from the file title on the dumping system.
<family name>	A family name if it is different from the family name on the dumping system.
CODEFILE	CODEFILE returns the code file family, a list of the stack numbers, and code file names that are specified.
CODEFILE <seg dict stack #> <code file title>	If this option is used, DUMPANALYZER uses the specified code file whenever it needs access to the code file for the specified segment dictionary stack number. This happens when DUMPANALYZER converts return control words (RCWs) to line numbers or when *MODE + IDNAMES* has been specified.
CODEFILE FAMILY <family name>	This option causes DUMPANALYZER to search for required code files on the specified disk family name. This option overrides the family name that the code file had on the dumping system. If both options are used, any family name specified in the <seg dict stack #> option overrides the CODEFILE FAMILY specification.

Normally, when DUMPANALYZER finds a mismatched code file, it asks for confirmation before proceeding. However, if the CODEFILE <seg dict stack #> command is being used for the required code file, this confirmation is not required. In addition, if the confirmation fails and the user specifies a code file using the FA command, DUMPANALYZER behaves as if a CODEFILE <seg dict stack #> command was entered with that title.

Note that when DUMPANALYZER searches for a code file, it searches first without using family substitution and then using family substitution.

CODEINFO (HDU Systems)

Note: *The CODEINFO command has been deimplemented. This command had meaning only on systems with address space number (ASN) memory, which is not supported on the Mark 3.9 system software release and later releases.*

COREMAP

The COREMAP command analyzes how memory is being used. The user can specify which reports and stacks to be analyzed. This command works only for COMPLETE memory dumps; it is rejected for PARTIAL or ALLINUSE memory dumps.

The first time a COREMAP command is entered it creates a disk file called MEMDUMP/COREMAP/<date>/<time>, which contains memory information. This process requires a scan through memory and takes a few minutes for every million words of memory. The *?AX WHERE* command can be used to determine the progress of this scan. The disk file can be used by subsequent runs of DUMPANALYZER to eliminate the need to re-create the screen.

```
<coremap>

— COREMAP ─┬─ HELP ──────────────────────────────────────────────────┤
           ├─ LAYOUT ─────────────────────┤
           ├─ SUMMARY ─┬───────────────────┤
           ├─ USAGE ───┘ ├─ STACK <number> ─┤
           │             └─ ALL ────────────┤
           ├─ DETAIL ─┬──────────────────────┤
           └─ HISTO ──┘ └─ STACK <number> ───┘
```

The following text describes the meaning of each construct:

COREMAP HELP	COREMAP HELP displays a short description of the various subcommands.
COREMAP LAYOUT	COREMAP LAYOUT displays a map of memory showing the amount and location of save, overlay, and available memory. Each line of the chart represents 32K words, with each character denoting 512 words. This option also displays a chart of the number of occurrences of adjacent areas by type.
COREMAP SUMMARY	COREMAP SUMMARY displays the amount of save, overlay, code, and available memory for the system.
COREMAP SUMMARY STACK <number>	Displays save, overlay, and code memory for the specified stack.
COREMAP SUMMARY ALL	Displays save, overlay, and code memory for every stack.
COREMAP USAGE COREMAP USAGE STACK <number> COREMAP USAGE ALL	Displays the number of areas and total words for each kind of memory area, such as array, stack, FIB, dope vector, and so forth, divided into save and overlay memory. Specifying a STACK number analyzes only the memory being used by that stack. Specifying ALL displays all stacks.

continued

continued

The type of memory area in the USAGE reports is the value of SPACEUSAGE with the following changes:

ARRAY	NORMAL ARRAY (0) AND PERM SAVE AREA (22).
IO	IOCB (6), IOCD (18) AND UINFO ARRAY (23).
EVENT	EVENT ARRAY (8) AND EVENTDOPE (26).
DATACOM	DATACOM QUEUE (24) AND STATION LIST (25).
MISC	SORT TABLE (11), SIB (12), ICM (21), PROCINFO (27), BOXINFO (28), AND RESERVEAREA (29).

COREMAP DETAIL

COREMAP DETAIL STACK <number>

Displays the number of areas and total words for each kind of memory area for each size range, such as 1 to 10 words, 10 to 20 words, and so forth, in a logarithmic progression. Specifying a STACK number analyzes only the memory being used by that stack.

The type of memory area in the DETAIL reports is the value of SPACEUSAGE except for some changes. See the preceding explanation of the COREMAP USAGE command for more information.

COREMAP HISTO

COREMAP HISTO STACK <number>

Displays a histogram of the number of areas in each size range of save, overlay, code, and available memory. Specifying a STACK number analyzes only the memory being used by that stack.

DC

The DC command prints a full data communications analysis.

<dc>

```
— DC ─┬─────────────────────────────────────────────────────────┤
      └─ NSP ─┬───────┘
              └─ MSG ─┘
```

The NSP option causes an analysis of the NSP tables to be printed. This analysis includes NSP line vectors, NSP line tables, and NSP station tables in addition to the DCC table analysis.

The MSG option causes messages in nontanked data comm queues in the DCALGOL queue to be analyzed.

Example

Figure 4–4 shows example output from the DC command.

```
INPUT: DC NSP MSG

                                                  DATA COMMUNICATIONS ANALYSIS
NSP DATACOM CONFIGURATION:
    MAXIMUM LSN                                 = 712
    CONFIGURED  : RELATIVE NSP NUMBER(S) = 0, 1, 2, 3, 4
    INITIALIZED: RELATIVE NSP NUMBER(S) = 0, 1, 3, 4
    NSP0107/00  : UNIT TAKEN, MAXLINES=15, DCC SNR=06E
    NSP0110/01  : UNIT TAKEN, MAXLINES=15, DCC SNR=06D
    NSP0109/03  : UNIT TAKEN, MAXLINES=31, DCC SNR=06C
    NSP0111/04  : UNIT TAKEN, MAXLINES=47, DCC SNR=06B

MCS INFORMATION:
    MCS[1]: SYSTEM/COMS
          STACK=083, PRIMARY QUEUE=0017
    MCS[2]: SYSTEM/CANDE
          STACK=07A, PRIMARY QUEUE=000F
    MCS[3]: SYSTEM/RJE
          STACK=DDF, PRIMARY QUEUE=022C
    MCS[4]: SYSTEM/X25
          NOT RUNNING, NO PRIMARY QUEUE
    MCS[5]: SYSTEM/DIAGNOSTICMCS
          DIAGNOSTICMCS, NOT RUNNING, NO PRIMARY QUEUE
    MCS[6]: RJE/BSC/HASP
          NOT RUNNING, NO PRIMARY QUEUE
    MCS[7]: SYSTEM/BNAMCS
          STACK=DB4, PRIMARY QUEUE=0029

DCC STATION TABLE
LSN 2: ----- NO LINE ASSIGNMENT -----
          0 500140 026050  (WRAPAROUND, ENABLED, READY, NOT/ATTACHED, MCS=5, MYUSE=IO, WIDTH=80)
          0 6F020A 640001  (CONTROLCHAR=4"6F", REMOTETYPE=2, RETRY=10, FREQUENCY=100, NIF INDEX=1)
          0 000000 000000  (QUEUES: PRIMARY=NONE; CURRENT=NONE; FILE=NONE; PSEUDOMCS=0)
          0 000000 000000  (DLS=NONE)
LSN 3: ----- NO LINE ASSIGNMENT -----
          0 541040 036050  (WRAPAROUND, ENABLED, READY, ATTACHED, MCS=1, MYUSE=IO, WIDTH=80)
          0 6F020A 640002  (CONTROLCHAR=4"6F", REMOTETYPE=2, RETRY=10, FREQUENCY=100, NIF INDEX=2)
          0 840000 017017  (MCS PARTICIPATES, ALL RESULTS)
                           (QUEUES: PRIMARY=0017; CURRENT=0017; FILE=NONE; PSEUDOMCS=0)
          0 000000 000000  (DLS=NONE)
LSN 4: ----- NO LINE ASSIGNMENT -----
          0 541040 046050  (WRAPAROUND, ENABLED, READY, ATTACHED, MCS=1, MYUSE=IO, WIDTH=80)
          0 6F020A 640003  (CONTROLCHAR=4"6F", REMOTETYPE=2, RETRY=10, FREQUENCY=100, NIF INDEX=3)
          0 840000 017017  (MCS PARTICIPATES, ALL RESULTS)
                           (QUEUES: PRIMARY=0017; CURRENT=0017; FILE=NONE; PSEUDOMCS=0)
          0 000000 000000  (DLS=NONE)
LSN 5: ----- NO LINE ASSIGNMENT -----
          0 541040 056050  (WRAPAROUND, ENABLED, READY, ATTACHED, MCS=1, MYUSE=IO, WIDTH=80)
          0 6F030A 640004  (CONTROLCHAR=4"6F", REMOTETYPE=3, RETRY=10, FREQUENCY=100, NIF INDEX=4)
          0 840000 017017  (MCS PARTICIPATES, ALL RESULTS)
                           (QUEUES: PRIMARY=0017; CURRENT=0017; FILE=NONE; PSEUDOMCS=0)
          0 000000 000000  (DLS=NONE)
```

Figure 4–4. DC Command Output

DCTRACE

The DCTRACE command displays or prints a summary of the last 250 requests or results of the specified network support processor (NSP). These requests or results are listed in the chronological order in which they were processed by the DCCONTROL stack of the specified NSP; thus, requests and results are interspersed in the summary. The summary includes such items as request type, associated logical station number (LSN), line number, station number (DLS) and associated error values.

The DCTRACE option of the ID (Initialize Data Comm) system command determines whether the system audits NSP traffic. If the DCTRACE option was not ON at the time of the dump, the DCTRACE command in DUMPANALYZER returns the message "NSP DOES NOT HAVE A TRACE TABLE".

```
<dctrace>

— DCTRACE —<relative NSP number>————————————————————————|
```

Example

Figure 4–5 shows example output from the DCTRACE command.

```
INPUT: DCTRACE 0
                        DC TRACE TABLE FOR STACK 068 "NSP107/00".

15:41:35.43358  REQUEST 37:   MAKE STATION READY REQUEST FOR LSN 46
                              LENGTH 7
15:41:35.53529  RESULT  67:   ACK RESULT IN RESPONSE TO REQUEST NUMBER 37
                              LENGTH 10
15:41:35.95994  REQUEST 38:   SET EXTERNAL REQUEST FOR LSN 46
                              LENGTH 14
15:41:35.96039  REQUEST 38:   CHANGE STATION EDITOR REQUEST FOR LSN 46
                              LENGTH 11
15:41:36.06589  RESULT  68:   ACK RESULT IN RESPONSE TO REQUEST NUMBER 38
                              LENGTH 10
15:41:36.06597  RESULT  69:   ACK RESULT IN RESPONSE TO REQUEST NUMBER 38
                              LENGTH 10
15:41:36.33219  REQUEST 39:   OUTPUT REQUEST FOR LSN 46, TEXT LENGTH 16
                              LENGTH 52
15:41:36.90996  REQUEST 40:   OUTPUT REQUEST FOR LSN 472, TEXT LENGTH 16
                              LENGTH 52
15:41:37.01290  RESULT  70:   REJECTED REQUEST RESULT NON-PRESENT STRUCTURE IN RESPONSE TO REQUEST NUMBER 40
                              LENGTH 10
15:41:37.39546  RESULT  71:   ERROR RESULT FROM LSN 46
                              LENGTH 36
15:41:37.39588  RESULT  72:   STATION NOT READY RESULT FROM LSN 46
                              LENGTH 12
15:41:44.10962  REQUEST 41:   MAKE STATION READY REQUEST FOR LSN 46
                              LENGTH 7
15:41:44.11035  REQUEST 42:   MAKE STATION READY REQUEST FOR LSN 46
                              LENGTH 7
15:41:44.21132  RESULT  73:   ACK RESULT IN RESPONSE TO REQUEST NUMBER 41
                              LENGTH 10
15:41:44.21170  RESULT  74:   ACK RESULT IN RESPONSE TO REQUEST NUMBER 42
                              LENGTH 10
15:41:44.31115  RESULT  75:   ERROR RESULT FROM LSN 46
                              LENGTH 36
15:41:44.31251  RESULT  76:   STATION NOT READY RESULT FROM LSN 46
                              LENGTH 12
15:41:44.31356  REQUEST 43:   MAKE STATION READY REQUEST FOR LSN 46
                              LENGTH 7
15:41:44.41101  RESULT  77:   ACK RESULT IN RESPONSE TO REQUEST NUMBER 43
                              LENGTH 10
```

Figure 4-5. DCTRACE Command Output

DEADLOCK

The DEADLOCK command prints information regarding stacks that hold locks or are waiting for a lock. The following items are analyzed:

- Global locks
- Header locks
- Directory locks
- Userdisk locks
- Operator reply waits

If a stack is waiting on any of these items, the stack number is printed. In addition, any of the locks that the stack holds are listed.

All EVENTs and EVENT ARRAYs declared in the MCP outer block are considered, but “hard” locks are not reported.

A stack is reported as waiting for a unit or path only when the stack is actually waiting, and not when the stack simply has I/O operations requested.

```
<deadlock>

              ┌←─────────────┐
— DEADLOCK ─┴─/1\─┬─ CRITICAL ─┤──────────────────────────────────────┤
                  └─<number>───┘
```

The following text describes the meaning of each construct:

DEADLOCK	All stacks that either hold locks or are waiting for a lock are listed by stack number, along with the names of the events or locks.
DEADLOCK CRITICAL	CRITICAL is specified to indicate that only stacks that hold locks and are also waiting on locks are to be listed. CRITICAL deadlocks might indicate an MCP problem.
DEADLOCK <number>	<number> specifies a particular stack. When a <number> is given, only locks for the stack identified by <number> are listed.

Example

The following is a sample of the information that is printed regarding stacks that either hold locks or are waiting for locks:

```
INPUT: DEADLOCK

LOCK AND WAITING STACK ANALYSIS

STACK Ø33  WAITING FOR EVENT G Ø32C ANABOLEVENT
STACK Ø37  WAITING FOR EVENT G ØB93 NET_EVENTSW[ØØØ5]
           WAITING FOR EVENT G ØB93 NET_EVENTSW[ØØØ1]
           WAITING FOR EVENT G Ø9AC PSE[ØØØF]
           WAITING FOR EVENT G Ø9AC PSE[ØØØF]
STACK Ø38  WAITING FOR EVENT G Ø9AC PSE[ØØØ6]
           WAITING FOR EVENT G Ø9AC PSE[ØØØ5]
           WAITING FOR EVENT G Ø9AC PSE[ØØØ6]
           WAITING FOR EVENT G Ø9AC PSE[ØØØ5]
STACK Ø4C  WAITING FOR EVENT G 1B98 SLOGMONITOREVENT
```

DEBUG

The DEBUG command dynamically sets, disables, or interrogates the DEBUG option. If DEBUG is set, a program dump is produced to aid in debugging DUMPANALYZER.

<debug>

```
— DEBUG ─┬─────┬──────────────────────────────────────────┤
         ├─ + ─┤
         ├─ - ─┤
         └─ ? ─┘
```

The following text describes the meaning of each option:

DEBUG	Sets the DEBUG option. This is a synonym for for *DEBUG +*.
DEBUG +	Sets the DEBUG option.
DEBUG –	Disables the DEBUG option.
DEBUG ?	Shows whether or not the DEBUG option is set.

Example

The following DEBUG command sets the DEBUG option:

```
DEBUG +

      DEBUG SET
```

DESCANAL

The DESCANAL command performs a descriptor analysis for all descriptors and in-use areas in the dump.

```
<descanal>

— DESCANAL ─┬──────────────┬──────────────────────────────────────┤
            └─ ORPHANED ──┘
```

The following text describes the meaning of each option:

DESCANAL	Displays the location of all in-use areas of the dump, all descriptors containing an address in the area, and miscellaneous descriptors that cannot be associated with a specific area.
DESCANAL ORPHANED	Displays areas for which no descriptor could be found and all miscellaneous descriptors.

Example

Figure 4–6 shows example output from the DESCANAL command.

```
INPUT: DESCANAL
ADDRESS 00000000 NOT PART OF A DUMPED MEMORY AREA
SORTING      1 DESCRIPTORS

------------------------------------------------  DESCRIPTOR ANALYSIS  -----------------------------------------------
IN-USE AREAS =     0,  DESCRIPTORS PRESENT =      1

           AREA      LENGTH  LOCATION  DESCRIPTOR   LOCATION  DESCRIPTOR   LOCATION  DESCRIPTOR   LOCATION  DESCRIPTOR
--------------------  BEGIN MISC DESCRIPTOR DUMP --------------------

----------------------------------------  END MISC. DESCRIPTOR DUMP  ----------------------------------------
#PERM SAVE =     1  #TEMP SAVE =     0, #OVERLAY =     0
```

Figure 4-6. DESCANAL Command Output

DISKFILE

The DISKFILE command routes the subsequent output to the user-specified disk output file.

```
<disk file>

— DISKFILE ─┬──────────────────────┬──────────────────────────┤
            ├─ ? ──────────────────┤
            └─ " ─<file title>─ " ─┘
```

The following text describes the meaning of each construct:

DISKFILE	The DISKFILE command routes the subsequent output to the disk output file previously specified by the user. If there is no output file available, an error message is displayed. This command switches the output mode from remote or printer to the disk file provided that a disk output file is available.
DISKFILE "<file title>"	The "<file title>" option of the DISKFILE command causes DUMPANALYZER to use the given file title as the disk output file and to route the subsequent output to it if no disk output file exists. If there is a disk output file available, an error message is displayed. The RELEASE DISKFILE command should be used in this case to release and save the current disk output file before assigning another disk file. See the RELEASE command for information on releasing and saving a disk output file.
DISKFILE ?	The question mark (?) option of the DISKFILE command can be used to find the name of the current disk output file.

For more information, see the discussion of the following commands in this section:

- RELEASE
- REPEAT
- SHOW

FIB

The FIB command analyzes a file information block (FIB) that starts at an arbitrary address and prints the results.

Text contained in buffers is printed unconditionally.

```
<fib>

— FIB ─┬──────┬─<simple address>─┬──────────────────────────┤
       ├─ AT ─┘                  │
       └─ VIA ─<ASD number>──────┘
```

The following text describes the meaning of each construct:

FIB <simple address>	
FIB AT <simple address>	The FIB starting at <simple address> is analyzed and the results are printed.

continued

continued

FIB VIA <ASD number>	The FIB located at <ASD number> is analyzed and the results are printed.

Example

Figure 4–7 shows example output from the FIB command.

```
INPUT: FIB VIA 33

 0(00) 3 000002 B00006  SELECTOR
 1(01) 3 033000 040000  FIBMSCW
 2(02) 0 000000 000000  FIBLOCK
 3(03) 0 000000 000000  RECORDSTATUS
 4(04) 6 800000 000000  FILESTATUS  (LEVEL=8,ALGOL, BUFFERS=0, NOTOPENED STATE)
 5(05) 0 000000 000000  TANKDATA1   (FILETYPE=0,UNIT=WORDS)
 6(06) 7 033000 000000  TANKDATA2   (PHYSICAL: BLOCKSIZE=816, MINRECSIZE=0, MAXRECSIZE=0)
 7(07) 0 000000 000000  DISKBLOCK   (SEGPERBLK=0, RECPERBLK=0)
 8(08) 0 000000 000000  PAGESPEC
 9(09) 0 000000 000000  IOINFO      (KIND=UNASSIGNED)
10(0A) 5 000000 400000  LEB  (LEB BLOCK SHOWN BELOW IN HEX)
11(0B) 0 000000 000000  IOAREA
12(0C) 0 000000 004608  BUFFDESC
13(0D) 0 000000 000000  SIOAREA
14(0E) 0 000000 000000  FMTBUFFDESC
15(0F) 0 000000 000000  FMTLOCK
16(10) 0 000000 000000  SELFDESC
17(11) 5 270001 2418E8  PWRITES
18(12) 3 00033B 1C8011  PREADS
19(13) 3 000000 002000  PWRITEN
20(14) 3 012000 844002  PREADN
21(15) 3 020290 0890E4  PSEEK
22(16) 0 000000 000000  PINITIATE
23(17) 0 000000 000006  PSEARCH
24(18) 0 000027 606500  PLOCKER
25(19) 0 000000 000000  PRELEASE
26(1A) 0 000000 000000  PMOVEOUT
27(1B) 5 C00000 50765A  PMOVEIN
28(1C) 5 800000 30470B  PWAIT
29(1D) 5 C00005 B05383  PFLOAT
30(1E) 5 C00004 F1CBA6  PCWCONTROL
31(1F) 6 800030 000800  RECOFF
32(20) 3 012000 84400C  RECORDCOUNT
33(21) 3 010201 6050E8  BLOCKCOUNT
34(22) 1 012000 84032C  LOWER
35(23) 0 000000 000001  UPPER
36(24) 0 000000 000000  MINRECSZ
37(25) 3 00033B 1C8005  RECSIZE
38(26) 3 000812 D8501E  I
39(27) 0 000000 000000  T
40(28) 0 000846 7FB37E  AEXP
41(29) 0 000000 00074A  DHEADER
42(2A) 0 000000 000000  FIBEOF       (EOFU=0, EOFV=0)
43(2B) 0 000000 000005  ACTNUM
44(2C) 0 000000 000012  SBLOCKING
45(2D) 0 3E6858 074A30  SIOINFO
46(2E) 0 000000 000005  SOFFSET
47(2F) 0 000000 000033  CURRENTBLOCK
48(30) 0 000893 F0101E  FILEEVENT1 (LOCKED/CONTENDED BY STK 008 @ 13F0:101E:0 (78747500) CLOSE1STREEL OF LMGUTS)
49(31) 0 000812 D0101E  FILEEVENT2
50(32) 0 000000 0016E6  OUTPUTTRANSLATION
51(33) 0 000000 000000  INPUTTRANSLATION
52(34) 3 012000 844008  USEROUTINES
53(35) 3 010200 F85A44  FLOPPYMISC   (NORMALIOLENGTH=4128)
54(36) 1 033002 540001  FIBLOCKSNR
55(37) 0 000000 000000  IOCB
56(38) 0 000000 000000  TRANSACTIONCOUNT
57(39) 0 000812 D0101E  LIBRARYINFO
58(3A) 1 012000 840000  RMM PHYSICALIOCOUNT (READ = 73728, WRITE = 8650752)
59(3B) 3 012000 844007  FILIOTIME
60(3C) 3 010409 B85022  MONITORWORD
61(3D) 1 033002 040000  BUFLINKS
62(3E) 1 012000 840000  IOMOM
```

Figure 4–7. FIB Command Output

FINDIOCB (EMS Systems)

The FINDIOCB command is used to find I/O control blocks (IOCBs) in a memory dump, and is valid only for EMS systems. The user can set various options to narrow the scope of the search and cause DUMPANALYZER to display only the IOCBs that satisfy those options.

<findiocb>

```
                 ┌←─────────────────────────────────┐
— FINDIOCB ──────┴┬──────────────────────────────┬──┴──────────────→
                  ├─/1\─ VIA ASD ────────────────┤
                  ├─<number>─┬───────────────────┤
                  │          ├─ - ──<number>─────┤
                  │          └─ TO ──<number>────┤
                  └─/1\─ RANGE ──<multiple address>┘

→─┬──────────────────────┬──────────────────────────────────────┤
  └─ : ── KEY ──<number>─┘
```

Note: *The <number> option refers to a word in the IOCB and must be a hex value from 0 up to the last IOCB word.*

The following text describes the meaning of each construct:

FINDIOCB	All of system memory is searched for the pattern "IOCB" in bits [47:16] and a 0 in the Tag. Each match is then analyzed and displayed.
FINDIOCB VIA ASD	The ASD table is searched for in-use memory areas with an ODDBALL field of 6 (IOCB). This method is faster than searching all of memory, but might miss some IOCB areas that were returned to the system shortly before the dump.
FINDIOCB <number>	Only the word with the specified number of each IOCB found is analyzed and displayed.
FINDIOCB <number>–<number>	
FINDIOCB <number> TO <number>	IOCB words in the range <number> through <number> are analyzed and displayed.
FINDIOCB RANGE <multiple address>	System memory within the specified address range is searched for IOCBs.
FINDIOCB :KEY <number>	IOCBs that are found where word <number> masked by the DUMPANALYZER MASK value matches the DUMPANALYZER PATTERN value. These IOCBs are then analyzed and displayed. (See the discussions of the MASK and PATTERN commands for more details.)

Examples

The following command displays an analysis of words 0, 3, A, B, C, D, E, and F in each IOCB that is found:

```
FINDIOCB Ø 3 A-F
```

The following command displays an analysis of words 3 and 5 of all IOCBs found from absolute address 90AAA through 92AAA. IOCBs are found by means of the ASD table.

```
FINDI 3 5 RANGE 90AAA to 92AAA VIA ASD
```

The following command examines all IOCBs found from absolute address 9A000 through 9B000. If word 2 of an IOCB masked with the system MASK value matches the system PATTERN value, then words 1 through 4 and A through E of that IOCB are analyzed and displayed. IOCBs are found by means of the ASD table.

```
FINDI VIA ASD 2 A TO E 1-4  R 9A000 to 9B000 :K 2
```

The following is an example of output from the FINDIOCB command:

```
INPUT: FINDI 3 RANGE 15000 TO 20000 VIA ASD
*********************************************
IOCB AT  0001E727   (ASD# =0214E)

WORD # 03         SELF PTR: 5 C00002 00214E
*********************************************
IOCB AT  0001EDC4   (ASD# =0247B)

WORD # 03         SELF PTR: 5 C00002 00247B

***********************************
2   VALID IOCBS WERE FOUND.
***********************************
```

FINDSTACKS

The FINDSTACKS command produces a list of all stacks containing an environment for a specified global procedure name in the MCP D0 stack.

```
<findstacks>
— FINDSTACKS —<global procedure name>——————————————————————|
```

Example

The following is an example of the output from the FINDSTACKS command:

```
INPUT: FINDSTACKS INTERLOOPER

STACKS WHICH CONTAIN AN ENVIRONMENT FOR INTERLOOPER:
  02B  02C  02D  02E
```

GC (EMS Systems)

The GC command displays the configuration of the system at the time of the dump. The resources of the system and information about the I/O subsystem are given,

including all DLPs base by base for all bases and DLPs configured at the time of the dump. The GC command is valid only for dumps from EMS systems.

```
<gc>

— GC ─┬──────────────────┬─┬─────┬─ OVERRIDE ─┬──────────────────┤
      └─<simple address>─┘ └─ : ─┘            
```

The following text describes the meaning of each construct:

GC and GC <simple address>	Memory starting at the address of the descriptor located at <simple address> is analyzed as a configuration file. If no <simple address> is given, the address where the CONFIGURATION global descriptor is located is used by default.
GC OVERRIDE	
GC <simple address> OVERRIDE	
GC <simple address> : OVERRIDE	If word 1 of the memory area analyzed does not contain the EBCDIC word CONFIG (indicating a valid configuration), the error message "Configuration location must be specified" is given unless the OVERRIDE option is selected. The OVERRIDE option causes the memory area to be analyzed as if it were a configuration file even if word 1 lacks the correct marker.

Example 1

```
INPUT: MD G CONFIGURATION

    8Ø24F(ØØØØØ)  5 8ØØØ3Ø C8B275
```

Entering GC or GC 8024F causes the configuration to be analyzed.

Example 2

The following is an example of the GC command display and an explanation of some of its terminology:

```
INPUT: GC
     ***** GROUP CONFIGURATION *****
GROUP ID: DEFAULT
PROCESSORS:
          PROC ID. 9;
          PROC ID. 10;
OPERATIONS:
          DL BACKUP   PRINTS;
          DL LOG      SCRATCH48;
          DL USERDATA 39;
          DL JOBS     CRONUS;
          DL CATALOG  PS;
          DL OVERLAY  504 SCRATCH49;
          DL SORT     CRONUS;
          DL DPFILES  SCRATCH48;
          DL IPFILES  CRONUS;
          HN          SFA6AB;
PERIPHERALS ALLOWED TO GROUP:
   2,3,17-29,32,44-69,75,80-181,183,184,187-235,237,240-255;
I/O:
   BASE 0/1/0
      IOP 1      % PORT 0  LEMPORT 0
         ADDRESS 0 DLPID   1   ODT2
          ADDRESS 1 DLPID   243 NSP5
          ADDRESS 3 DLPID   76  MT6
          ADDRESS 4 DLPID   112 NSP5
          ADDRESS 5 DLPID   44  SCSI1
          ADDRESS 6 DLPID   20  TP5
          ADDRESS 7 DLPID   28  MT1;
    BASE 0/2/0
       IOP 2      % PORT 0  LEMPORT 0
          ADDRESS 1 DLPID   157 ICP1
          ADDRESS 2 DLPID   236 IPFIPS1
          ADDRESS 5 DLPID   68  SCSI1;
```

The following text describes the meaning of elements that commonly appear in the GC output:

PROC ID.	A unique number identifying a host processor
PATH	A description of the unique path from a host to the DLP of a unit. The path consists of an I/O processor (IOP) and a line expansion module (LEM) port.
IOP	The relative I/O processor (IOP) that the path traverses. (There can be up to eight ports per processor.)
LEMPORT	The relative line expansion module (LEM) port that the path traverses.

continued

continued

BASE	A description of the base in which the DLP of the unit resides.
DEPENDENT HOST	If a path to a unit or units is through an outboard host (such as an NSP), a description of the outboard host.

GRAPHS

The GRAPHS command causes MCP stack graphs to be printed one level deep.

```
<graphs>
— GRAPHS ─┬──────────┬──────────────────────────────────────┤
          └─<number>─┘
```

The following text describes the meaning of each construct:

GRAPHS	All MCP stack graphs are printed one level deep.
GRAPHS <number>	The graphs for the stack identified by <number> are printed.

Example

The following is an example of the output from the GRAPHS command:

```
INPUT: GRAPHS 36

GRAPHS . . . . . STACK Ø36      = *SYSTEM/DRCSUPPORT.
    PASSIVELIBGRAPH . . . . Ø12 = *SYSTEM/MCP/39Ø19F ON DISK. (1)
    PROGRAMDUMPGRAPH  . . . Ø39 = *SYSTEM/DRCSUPPORT ON DISK. (1)
                            Ø12 = *SYSTEM/MCP/39Ø19F ON DISK. (1)
```

HARDINFO (RMM Systems)

The HARDINFO (Hardware Information) command displays various system state registers and data. HARDINFO is valid only when analyzing a standalone tape dump from an RMM system.

```
<hardinfo>
— HARDINFO ─────────────────────────────────────────────┤
```

Example

Figure 4–8 shows example output from the HARDINFO command.

```
INPUT: HARDINFO
IOP 1 IS DISTINGUISHED
TCP 1 IS DISTINGUISHED
ITC 4 IS DISTINGUISHED
PROCESSOR 4
  IN STACK 037 F=00000002376C S=000000023774 LL=1 @ 1021:0C47:4 CONTROL STATE
  FAIL INFORMATION

    HALT REASON IS 0: UNC HALTED
  SWITCH SETTINGS:
    ERROR HALT IS ON
    OPERATOR DEPENDENT HALT IS OFF
    CONDITIONAL HALT IS OFF
    INSTRUCTION HALT IS OFF

  TCP CPMS STRUCTURE
    CPM 4
      CPM STATE          000000000003
      (RESERVED)         000000000000
      CURRENT STACK      000000000037
      NEXT STACK         000000000035
      ACTIVE TIME        0000435EBE86
      IDLE TIME          FB0000022D52
      (RESERVED)         FF0000000000
      CPMLINK/PRIORITY   0000000A1900
      PREFERRED STACK    00000000002B
      HELD STACK         000000000000
      CPM REQUESTTIME    00004362E980
      CPM SAVE TIME      000000000000
      REJECTION TIME     00004221F2BD
      SELECTED STACK     000000000035
      MIC CNTRL WORD     0A0000000010
      CONTINUE COMMAND   8F20E1000000
MSM 1
  PRESENT MSU BOARDS: 0 1 2 8 9
  ENABLED MSU BOARDS: 0 1 2 8 9
  TO BE SAVED MSU BOARDS: NONE
  MSM FAIL REGISTER INFORMATION
    SCHED A:    0 000000 000400  0 000000 0001A4    NO ERRORS

    SIM 0:      0 000000 0000E0  0 000000 1E5883    NO ERRORS

    SIM 1:      0 000000 0000E0  0 000000 345881    NO ERRORS

CONSOLE ERROR LOG ANALYSIS
(ANALYZE SUMLOG FOR MORE DETAILS)
     LOAD REPORT : FORCE DUMP
```

Figure 4–8. HARDINFO Command Output

HDR

The HDR command analyzes the disk file header stack or a particular header and prints out that analysis.

<hdr>

```
— HDR ─┬──────────────────────────────────────────┬──────────────────────────┤
       ├─ SUMMARY ────────────────────────────────┤
       ├─ ROWS ───────────────────────────────────┤
       ├─<number>─────────────────────────────────┤
       ├─ " ─<file title>─ " ─┐  ┌─ ROWS ─┤
       └─ JOB <number> ───────┘  └─ RAW ──┘
```

The following text describes the meaning of each construct:

HDR	The HDR form of the command prints an analysis of all the file headers that existed in memory when the dump was performed. Row address words are not printed.
HDR SUMMARY	The HDR SUMMARY form of the command prints a list containing the location and names of all the headers that existed in memory when the dump was performed.
HDR ROWS	The HDR ROWS form of the command produces a printout with all the information that the HDR form produces plus an analysis of the row address words.
HDR <number>	The HDR <number> form of the command prints the same information as HDR, but only for the header specified by <number>.
HDR <number> ROWS	The HDR <number> ROWS form of the command prints the same information as HDR ROWS, but only for the header specified by <number>.
HDR <number> RAW	The HDR <number> RAW form of the command produces a printout in hex format of the entire header specified by <number>.
HDR "<file title>"	The <file title> form of the command causes the header stack to be searched and the requested header printed if it is located.
HDR JOB <number>	The JOB <number> form of the command causes the header stack to be searched and the requested header is printed if it is located.

Example

Figure 4–9 shows example output from the HDR command.

```
INPUT: HDR "(PHOEBE)EDITOR/OPTIONS ON ASD"
HDR [02D4]  AT 5 800002 516851  (LENGTH = 37) NAME : (PHOEBE)EDITOR/OPTIONS ON ASD
HDRO[02D4]  =  0 001D04 A015E6  RECORD SIZE = 37, LOCATION = 5606, BASE UNIT = 29
            0000  0 3F3F04 A015E6  BLOCKLENGTH = 37, LOCATION = 5606
            0001  0 0010C0 C12000  OPENCOUNT = 1, FILEKIND = 192 (DATA), PERMANENT, WRITTEN ON, FIXED SIZE = 18
                                   AVAIL SPACE = 0
            0002  0 000000 000000  EXTMODE = SINGLE, UNITS = WORDS, FILETYPE = 0
            0003  0 001E00 00001E  BLOCKSIZE = 30, MINRECSIZE = 0, MAXRECSIZE = 30
            0004  0 4E4760 2A3215  TIMESTAMP (DATE = 90039, TIME = 17:12:33.6682)
            0005  0 63000A 00000A  VERSION = 6, SECURITYTYPE = PRIVATE, SECURITYUSE = I/O, MODE = SHARED
                                   ROWS = 10, ROWSIZE = 10
            0006  0 000001 000000  SAVEFACTOR = 0, CYCLE = 1, VERSION = 0, FILEORGANIZATION = NOTRESTRICTED
            0007  0 080382 500000  OPT ATTRIB WORDS = 3, TIMESTAMP SYNC'D, EOF ROW SCRUBBED, FLATHDRLENGTH = 37
            0008  0 01901F 01D001  TITLEINDEX = 31, TITLESIZE = 25, BASEUNIT = 29, LASTUNIT = 1
            0009  0 000000 000002  DISK EOFU = 0, DISK EOFV = 2
            0010  0 4E475F DD02D2  CREATION TIMESTAMP (DATE = 90039, TIME = 17:09:19.4269)
            0011  0 4E475F DD02D2  ALTER    TIMESTAMP (DATE = 90039, TIME = 17:09:19.4269)
            0012  0 4E475F DD02D2  ACCESS   TIMESTAMP (DATE = 90039, TIME = 17:09:19.4269)
            0013  0 000000 000000  MULTIUSE WORD
            0014  0 000000 1A02D4  ORIGHDRVERSION = 0, CONTENDORS = 0, COREINDEX = 2D4
            0015  0 800000 008400  FILESTRUCTURE = ALIGNED180 (BY DEFAULT), SECTORSIZE = 180
            0016  0 001000 000000  DRCINFO FAMILYLISTINDEX = 0(INVALID), DRCUSERCOUNT = 1
            0017  0 000000 000000  NOT USED

            NUMBER OF ALLOCATED ROWS = 1

            OPTIONAL ATTRIBUTE WORDS:

            0000  0 800000 000000  AVAILABLE

            0001  0 800000 000000  AVAILABLE

            0002  0 000E00 0007FF  ATTNO = 14 (** UNKNOWN **), TYPE = FIELD
                                   VALUE : 000007FF (DEC) : 2047
```

Figure 4-9. HDR Command Output

HEADING

The HEADING command causes the heading (first page) of a dump to be printed at the terminal or line printer.

The information provided in the header includes the title of the MCP, the cause of the memory dump, and the processor that initiated the dump.

For an enhanced memory dump, a title displaying "<machine type> <dump type> MEMORY DUMP" is shown. In this title, <machine type> is the name of the machines, and <dump type> can be COMPLETE, ALLINUSE, or PARTIAL.

```
<heading>

— HEADING ————————————————————————————————————————————|
```

Example

The following is an example of the information returned by the HEADING command:

```
INPUT: HEADING

DUMPANALYZER Version: 39.42.280,  SDASUPPORT Version: 39.40.44
 *****************************************
 *       A15  COMPLETE Memory Dump       *
 *         (System Serial: #    8)       *
 *        A15  MCP: MARK 39. 23.3353     *
 *         3/23/90           20:59       *
 *  This is an analyzed dump from DISK:  *
 *     *MEMORY/SAVEDUMP/9002082010 ON MCP*
 *****************************************
Clock:              0003870E89EE (10:06:01 Since HALT/LOAD)
Actual Clock:       0007555BCFA4 (20:59:52)
Hostname:           MPA15C
Dumping MCP:        *SYSTEM/MCP/39023A ON DISK. (MCP/AS)
                    Creation Timestamp: 03/22/90 22:43:08
Intrinsic Name:     (NOT SPECIFIED)

Cause of NONFATAL Dump: FAULTED MCP CODE
Memdump called @ 1109:0111:1 (21074000) PROCESSKILL
Memory dump performed by:   Processor  5 in Stack 7DB  S=>001EE589
                            Processor  4 in Stack 0F5  S=>001C2F03
```

The following are the meanings of selected items in the HEADING command output:

Item	Description
Memdump called @	This item specifies the segment and relative address where memdump was called.
Memory dump performed by	This item specifies the processor that initiated the dump, the stack number in hexadecimal, and the S register setting. Any other active processors are then printed with their stack numbers and S register settings. When the memory dump procedure is invoked, it "seizes" all processors with a processor-to-processor interrupt. This information indicates the position of the processors at the time they were seized.

HEAP

The HEAP command displays information about the heaps in use by a Pascal process. Before the HEAP command is used, the HEAPSTACK command must be used to specify the target process stack.

```
— HEAP ─┬────────────────────────────────────────────────────────┬─┤
        └─┬───────────────────┬─<heap pointer>─┬─────────────────┤
          └─ S ─<heap number>─┘                └─ FOR ─┬─<length>─┤
                                                       └─ SIZE ───┘
```

<heap pointer>

```
—<Pascal pointer>─┬──────────────────┬──────────────────────────┤
                  └─ + ─<skip value>─┘
```

The <heap number>, <length>, and <skip value> constructs are each <hexadecimal number>s.

The following are the meanings of each of the elements of the HEAP command:

HEAP	If entered without parameters, this command displays information about all the heaps in use by the process stack.
S <heap number>	Specifies a separate heap that the <heap pointer> is applicable to. If the S <heap number> clause is not used, or if the <heap number> specified is 0, the nonseparate heap is displayed.
<heap pointer>	Causes the display to begin at the specified point in the heap. For separate heaps, the <heap pointer> is a row index into the dope vector for the separate heap. For the nonseparate heap, the <heap pointer> is a word offset into the heap array. If the <skip value> is included, it specifies an offset, in words, from the <Pascal pointer>. The display starts at the specified offset. The <skip value> cannot skip past the end of a separate heap entry or beyond the end of a nonseparate array row.

continued

continued

FOR

Specifies the number of words to display. If no FOR clause is included, a single word is displayed.

If the FOR <length> clause is used, then <length> specifies the number of words to be displayed. The actual number of words displayed is the minimum of <length> and the number of words remaining in the row after the <heap pointer>.

If the FOR SIZE clause is used, the structure of the heap determines the number of words displayed. For separate heaps, the size is the row size of the specified heap, minus any specified <skip value>. For the nonseparate heap, the size is the number of words in the heap page, minus any specified <skip value>.

Examples

The following is an example of the output from a simple HEAP command for a process stack that is using only the nonseparate heap:

```
INPUT: HEAP
NON SEPARATE HEAP ROWS 0 - 19 IN USE, TOP OF HEAP IS 196F1B
SYSTEM PAGE SIZE IS 100 (256) WORDS
```

The remaining HEAP examples show the progressive analysis of a single process stack that uses multiple separate heaps. The following is the output from a simple HEAP command for this process:

```
INPUT: HEAP
A (10) SEPARATE HEAPS IN USE
# 1 @ 22, TOP: 80, FREE HDR: 7E, MAX SIZE: FFFFF, 100 OF AB WORDS
# 2 @ 23, TOP: 0, FREE HDR: -1, MAX SIZE: FFFFF,  ABSENT
# 3 @ 24, TOP: 53, FREE HDR: -1, MAX SIZE: FFFFF, 100 OF 9 WORDS
# 4 @ 25, TOP: 89, FREE HDR: 7A, MAX SIZE: FFFFF, 100 OF 6 WORDS
# 5 @ 26, TOP: 2, FREE HDR: 1, MAX SIZE: FFFFF, 100 OF 2AAE WORDS
# 6 @ 27, TOP: 30, FREE HDR: 2D, MAX SIZE: FFFFF, 100 OF 12 WORDS
# 7 @ 28, TOP: 6F, FREE HDR: 3D, MAX SIZE: FFFFF, 100 OF C WORDS
# 8 @ 29, TOP: 0, FREE HDR: -1, MAX SIZE: FFFFF,  ABSENT
# 9 @ 2A, TOP: 5A, FREE HDR: -1, MAX SIZE: FFFFF, 100 OF 8A WORDS
# A @ 2B, TOP: 2, FREE HDR: 1, MAX SIZE: FFFFF, 100 OF 6 WORDS
SYSTEM PAGE SIZE IS 100 (256) WORDS
```

The following are the meanings of various elements in the previous example:

# xx	Heap number, 1 through nn
@ yy	Offset of descriptor from bottom of stack (BOSR)
TOP:	Current top of heap marker
FREE:	Head of free list within heap. A value of -1 means the free list is empty.
MAX SIZE:	Maximum number of entries allowed in heap

continued

continued

ABSENT:	Heap not yet used
zz of aa WORDS	zz rows (entries) of size aa words each

The following command attempts to display row 100 in separate heap 0:

```
INPUT: HEAP S Ø 1ØØ
NON SEPARATE HEAP IS NOT AVAILABLE
```

Because separate heap 0 does not exist, DUMPANALYZER attempts to display word 100 of the nonseparate heap. Because there is no nonseparate heap for this process, DUMPANALYZER returns the error shown.

Figure 4–10 shows the output from several commands that analyze separate heaps associated with the same process stack.

```
INPUT: MD STK 93E BOSR + 2B
000D5C73(00000000)  5 800010 0104A4

:READY

INPUT: MD VIA 104A4 FOR 100
000B32C4(00000000)  5 800000 610CCE  5 800000 6139E3  5 000000 600000  5 000000 600000  5 000000 600000  5 000000 600000
000B32CA(00000006)  5 000000 600000  THRU  000B33C3(000000FF)

:READY

INPUT: MD VIA 10CCE FOR 6
0022A538(00000000)  3 4EFC63 468848  3 000807 30E3EF  3 400000 000001  3 4EFC63 46D482  3 000803 90E3F0  0 C4C9E2 D7E9C4

:READY

INPUT: HEAP S A 0 FOR SIZE

0022A538(00000000)  3 4EFC63 468848  3 000807 30E3EF  3 400000 000001  3 4EFC63 46D482  3 000803 90E3F0  0 C4C9E2 D7E9C4
:READY

INPUT: MD VIA 139E3 FOR 6
000FD356(00000000)  3 4EFC63 796699  3 000807 30E3EF  3 000000 000000  3 4EFC63 79A585  3 000803 90E3F0  0 C4C9E2 D7E9C4

:READY

INPUT: HEAP S A 1 FOR SIZE

000FD356(00000000)  3 4EFC63 796699  3 000807 30E3EF  3 000000 000000  3 4EFC63 79A585  3 000803 90E3F0  0 C4C9E2 D7E9C4
:READY

INPUT: HEAP S 3 1 FOR SIZE

001EF936(00000000)  3 4EFC63 10C1FB  3 000808 91238C  0 35C5E6 7DC440  0 001300 011C04  0 0CF0F8 F0F0F0  0 C2F0F0 F0F2F9
001EF93C(00000006)  0 F2C5E6 7DC440  0 D5C5E6 7DC440  0 D5C5E6 7DC440
:READY

INPUT: HEAP S 3 1+2 FOR SIZE

001EF938(00000000)  0 35C5E6 7DC440  0 001300 011C04  0 0CF0F8 F0F0F0  0 C2F0F0 F0F2F9  0 F2C5E6 7DC440  0 D5C5E6 7DC440
001EF93E(00000006)  0 D5C5E6 7DC440
:READY

INPUT: HEAP S 3 1+2 FOR 3

001EF938(00000000)  0 35C5E6 7DC440  0 001300 011C04  0 0CF0F8 F0F0F0
:READY
```

Figure 4–10. Heap Analysis Examples

The following are notes about the commands shown in Figure 4–10:

MD STK 93E BOSR + 2B

Displays the stack cell containing the descriptor for heap A. The heap is identified by its offset of 2B; this offset appears in the output for the simple HEAP command shown previously. The simple HEAP command also shows that heap A can hold FFFFF entries, and that currently space for 100 rows of size 6 words has been allocated. The top of heap value is 2, indicating the next nonallocated row that can be used. The free header is 1, which means that the row at index 1 has been disposed and is in the free chain.

MD VIA 104A4 FOR 100

Displays the contents of the dope vector of the heap. The address of the dope vector, 104A4, was taken from the output of the previous command. It can be seen that only rows 0 and 1 have been used.

MD VIA 10CCE FOR 6

Displays the contents of row 0 of separate heap A. The address of the row descriptor, 10CCE, was derived from the first word of the dope vector for heap A.

Because the process was executing a Pascal program compiled with the HEAPDEBUG option set, several words of trace information appear at the start of the row; these are the tag 3 words. The following are their meanings, in order:

- The first word is the timestamp of the NEW call that last returned the row for program use.
- The second word is the return control word (RCW) where the NEW call was made.
- The third word is the free list link. Note the free list link of –1.
- The fourth word is the dispose timestamp.
- The fifth word is the return control word (RCW) where the DISPOSE call was made.

The third, fourth, and fifth trace words appear only for rows that are disposed.

HEAP S A 0 FOR SIZE

Displays row 0 of separate heap A. This HEAP command displays the same information as the previous MD command, but is easier to use. That is, the HEAP command can be used without examining the heap descriptor and dope vector first.

MD VIA 139E3 FOR 6

Displays the contents of row 1 of separate heap A. The address of the row descriptor, 139E3, was derived from the second word of the dope vector for heap A. Like row 0, row 1 is disposed and includes five words of trace information in the order described for row 0. Note the free list link of 0.

HEAP S A 1 FOR SIZE

Displays row 1 of separate heap A. This is the same area as that displayed by the previous MD command.

continued

continued

HEAP S 3 1 FOR SIZE	Displays the row at index 1 in heap number 3. This is a nondisposed area. The amount of memory displayed is the size of the row; that is, the size of the record type that is in this heap. As discussed previously, the row contains trace information in the tag 3 words. Because the row is not disposed, only the first two tag 3 words appear: the timestamp of the last NEW call and the RCW for the NEW call.
HEAP S 3 1+2 FOR SIZE	Displays the row at index 1 in heap number 3. The first two words, storing the trace information, are omitted. The amount of memory displayed is equal to the size of the row minus two, for the words that are omitted.
HEAP S 3 1+2 FOR 3	Displays the same information as the previous command, except that the output is limited to 3 words.

HEAPSTACK

The HEAPSTACK command specifies or displays the process stack and the heap offset used by the HEAP command.

The heap offset is measured relative to the mark stack control word (MSCW) at lex level 2 in the process stack. The default heap offset is 2, but the actual heap offset can vary depending on whether the program contains modules and depending on the version of the Pascal compiler used.

```
— HEAPSTACK ─┬───────────────────────────┬──────────────────────→
             └─<hexadecimal stack number>┘

→┬────────────────────────────────┬──────────────────────────────┤
 └─ OFFSET ─<hexadecimal offset>──┘
```

Explanation

The following text describes the meaning of each construct:

HEAPSTACK

Displays the hexadecimal stack number of the process stack currently targeted for all HEAP commands, and the hexadecimal offset used to locate the heap descriptor.

HEAPSTACK <hexadecimal stack number>

Specifies the process stack to be used as the target for all HEAP commands. The hexadecimal offset is reset to its default value of 2.

HEAPSTACK OFFSET <hexadecimal offset>

Specifies the offset of the heap descriptor relative to the MSCW at lex level 2.

HEAPSTACK <hexadecimal stack number> OFFSET <hexadecimal offset>

Specifies the process stack and offset to be used for subsequent HEAP commands.

The HEAPSTACK command displays an error message if the specified process stack is of the wrong type or in the wrong state for use in a HEAP command.

Examples

The following example specifies stack 1DE as the target and causes the default offset of 2 to be used. The response indicates that DUMPANALYZER found a nonseparate heap at the default offset.

```
INPUT: HEAPSTACK 1DE
HEAPSTACK = 1DE, Compiled by level 37.335 Pascal, BOSR = 7B83B
Separate Heaps are not in use
Non Separate Heap row Ø in use (Default heap size),Top Of Heap is 6F
System Page Size is 1ØØ (256) words
```

In the following examples, the user first specifies stack 1E0 as the target and causes the default offset of 2 to be used. The response indicates that there is no heap descriptor at that offset, and suggests that an offset of 2A be used instead. The user reenters the HEAPSTACK command, specifying an offset of 2A. The response indicates that there is a heap descriptor at that location.

```
INPUT: HEAPSTACK 1EØ
HEAPSTACK = 1EØ, Compiled by level 37.335 Pascal, BOSR = B9A8Ø
Non Separate Heap Seg Dope not present at (2,2)
Failure loading heap information.
Please specify the offset of the Non Separate Heap Array.
Possible offset(s):
2A

INPUT: HEAPSTACK OFFSET 2A
Separate Heaps are not in use
Non Separate Heap row Ø in use (Default heap size),Top Of Heap is 6F
System Page Size is 1ØØ (256) words
```

Following is an example of a simple HEAPSTACK command. The response shows the stack number and offset that are currently in use.

```
INPUT: HEAPSTACK
Heapstack = 1EØ, BOSR = B9A8Ø
Heapstack Offset = 2A
```

In the following example, the HEAPSTACK command specifies both the stack number and the offset. The response indicates that DUMPANALYZER found the heap descriptor at the specified offset.

```
INPUT: HEAPSTACK 510 OFFSET 2A
HEAPSTACK = 510, Compiled by level 39.80 Pascal, BOSR = 6E8B
3 (3) Separate heaps in use
Non Separate Heap row 0 in use(Default heap size),Top Of Heap is 6F
System Page Size is 100 (256) words
```

HELP

The HELP command provides information about DUMPANALYZER commands. When DUMPANALYZER is run from the ODT, the response to HELP is displayed on the ODT.

```
<help>

— HELP ———————————————————————————————————————————————————————————————|
         |—<dumpanalyzer command>—|
         └—<meta-item>————————————┘
```

The following text describes the meaning of each construct:

HELP	A list of available DUMPANALYZER commands is given.
HELP <dumpanalyzer command>	A railroad syntax diagram and a brief explanation of the commands is given.
HELP <meta-item>	A further explanation of most metalinguistic items (for example, <number>) is provided. The broken brackets are required for meta-items.

IO

The IO command invokes input/output (I/O) analysis of all peripherals.

```
<io>

— IO ——————————————————————————————————————┬—————————┬—————————┬—|
      |                                    | └ SUMMARY ┘ └ RESULTQ ┘
      |  ┌←————————————————————————————┐   |                     |
      |——┴—/1\—<unit spec.>———————————┤   |                     |
      |    └—/1\— UINFO ——┬——————————┬—┘   |                     |
      |                   |— NAMES ——|                            |
      |                   └— ALL ————┘                            |
      └—<active spec.>————————————————————————————————————————————┘

<unit spec>

— UNIT ——┬——————————————<decimal number>——┬———————————————————|
         |— INTERNAL —|                   |
         |— LOGICAL ——┘                   |
         |—<unit type>————————————————————|
         |— CATALOG ——————————————————————|
         |— OFFLINEUNIT ——————————————————|
         |— VOLUNIT ——————————————————————|
         └— TAPEUNIT —————————————————————┘
```

```
<active spec>

— ACTIVE ─┬────────────────────────────────────────────────────────┤
          └─ UNIT ─┬─────────────┬─<decimal number>─┘
                   ├─ INTERNAL ─┤
                   └─ LOGICAL ──┘

<unit type>

──┬─ MT ─┬─────────────────────────────────────────────────────────┤
  ├─ PK ─┤
  ├─ DK ─┤
  ├─ CD ─┤
  ├─ HC ─┤
  ├─ DC ─┤
  ├─ DP ─┤
  ├─ NP ─┤
  ├─ VC ─┤
  ├─ SC ─┤
  ├─ LP ─┤
  ├─ CP ─┤
  ├─ CR ─┤
  ├─ IP ─┤
  ├─ HY ─┤
  └─ SP ─┘

<AX options>

— ? — AX ─┬─ WHERE ─┬─────────────────────────────────────────────┤
          ├─ STOP ──┤
          └─ SKIP ──┘
```

The following text describes the meaning of each construct:

IO

Each I/O peripheral is analyzed and result queues are displayed.

IO ACTIVE

On EMS systems, the IO ACTIVE form searches for active IOCBs; this search can be asynchronously monitored and controlled by entering the AX (Accept) message command after the IO ACTIVE form has been initiated.

The WHERE option displays the location of the search, the number of the physical unit now being searched, and the number of active IOCBs found so far.

The STOP option cancels the search and prompts the user with the ":READY" message; no information from the search is saved.

The SKIP option stops the search of the current unit and skips to the next unit (if any). The search occurs from the high end of memory toward 0; thus, memory indices printed in response to ?AX WHERE decrease.

The IO ACTIVE form of the command is equivalent to the simple IO form of the command, and has no effect when analyzing a dump created by an HDU system.

continued

continued

IO UNIT <decimal number>	The IO UNIT <decimal number> form of the command causes the analysis of the designated I/O peripheral. Each I/O peripheral has a unique physical unit number associated with it. It is that number that is used as the decimal number in this form of the command. The IO UNIT form of the command can also indicate whether the unit number used is an internal unit number or a logical unit number. If neither INTERNAL nor LOGICAL is used, the <decimal number> is assumed to be an external unit number. An example of this form of the command is *IO UNIT INTERNAL 27*.
IO UNIT <unit type>	This form of the command causes the analysis of all peripherals of the designated I/O peripheral unit type.
IO UNIT CATALOG	
IO UNIT OFFLINEUNIT	
IO UNIT VOLUNIT	
IO UNIT TAPEUNIT	These forms of the command provide specific table information concerning catalog units. To give significant information, these forms of the command must be followed by a UINFO option.
IO UNIT VSUNIT	This form of the command provides specific table information concerning the dummy unit used for the InfoGuard Volume Directory.
IO UINFO	
IO UINFO NAMES	
IO UINFO ALL	If the UINFO option is specified by itself, a unit information analysis is provided. A basic analysis of each I/O peripheral, identical to that provided by the IO form of the IO command, is given; in addition, a UINFO entry for each peripheral is given. The UINFO option lists the contents of each UINFO entry as an uninterpreted array. When the UINFO NAMES option is used, it provides a columnar listing of the entries and the purpose of each word. The mass-storage lists and other arrays attached to the UINFO entry are not printed. These lists can be obtained by using the UINFO ALL option. The UINFO ALL option applies only to pack and disk.
IO RESULTQ	The IO RESULTQ form of the command provides the user with result queues that are otherwise suppressed. This option is only valid on EMS systems. An example of this option is "IO UNIT MT RESULTQ".
IO SUMMARY	The IO SUMMARY form of the command provides a skeleton analysis of each specified I/O peripheral unit. The unit table and I/O queue expansion are suppressed from the output. An example of the use of this option is IO UNIT MT RESULTQ SUMMARY.

Examples

Figures 4–11 and 4–12 show example output from the IO command.

```
INPUT: IO UNIT 113

(UNIT @ 000FA1C8, UNITMAP @ 000F8FE6, GIVETAKERCW @ 000FA26C,
UNITCONTROL @ 000FA4A9, UINFO @ 000FA176, UNITIOERR @ 0001FF49,
PATHCNTRL @ 000FA003, COMMANDQUEUES @ 000FC062, HORIZONTALQUEUES @ 000FBFFD,
RESULTQUEUES @ 0001FFEF, UNITSTATUS @ 0001FEA5, UNITMISC @ 0001FEF7)

********** MT113  LOGICAL UNIT 56 **********
0 3C8000 019000  UNIT[038] MT113 (9-TRK / PE / 1600 BPI)
                                   LABEL READ,
                                   LABELLED *SYSTEM/FILE000 (REEL 1)
0 000000 800000  UNITIOERR[038]
                                   SUBTYPE=4(MTGCRV)
0 1E0000 029047  UNITCONTROL[038]
                                   FIPS COMPATIBLE GCR MAG TAPE, REL UNIT=00, DYNAMIC UNIT,
                                   HAS STATUS, STATUS RUNNING, PATH NODE INDEX=00047
0 090000 071012  UNITMAP[038]
                                   UM NUMBER OCCUPIED, UMLOGF=56, UMPHYSF=113
0 04E871 D8536C  GIVETAKERCW[038]
                                   UNIT GIVEN @ 136C:071D:4 (51641600) READALABEL BY STACK 04E
0 E00001 F20002  UNITSTATUS[038]
                                   UNIT EXISTS, PHYSICALLY READY, LOGICALLY READY.
                                   AN IOP SEES THIS UNIT AS READY.
                                   PATHS SEEING UNIT READY: 1.

    >>>>> UNIT QUEUE @ 000FBD2F
          UQ CONTROL WORD: 0 10CC00 000154  INACTIVE COUNT=00, ACTIVE COUNT=00, ACTIVE LIMIT=01,
                                            DYNAMIC PATH SEL., SUSPENDED, HORIZ. Q PRESENT
          HEAD IOCB LINK:  0 000000 000000
          TAIL IOCB LINK:  0 000000 000000
          HQ HEAD PTR:     5 E00000 1021F1
          DU QUEUE LINK:   0 000000 000000
          LAST IOCB:       5 C00001 E02282
          SPARE IOCB DESC: 0 000000 000000
          I/O OPT WORD #1: 0 000000 000000
          I/O OPT WORD #2: 0 000000 000000
          PATH WORD:       5 C00002 A021F2
          UNIT PATH INFO:  0 000000 000002  READY PHYSICAL PATH TO DLP INDEXES: 1.
          SOFTWARE WORD:   0 010086 000008  STATUS MONITORING:  INITIATED BY DOSTATUSIO
                                            MISC. INFORMATION:  LAST SUSPENDED BY DOSTATUSIO
                                            SUSPENDED BY:       DOSTATUSIO
          DLP INDEX #1:    5 800001 E02282
          DLP INDEX #2:    0 000000 000000
          DLP INDEX #3:    0 000000 000000
          DLP INDEX #4:    0 000000 000000
          DLP INDEX #5:    0 000000 000000
          DLP INDEX #6:    0 000000 000000
          DLP INDEX #7:    0 000000 000000
          DLP INDEX #8:    0 000000 000000
          CHECK HUNG I/O:  0 000000 000000

       <****** PATH GROUP TABLE FOR UNIT QUEUE @ 000FBD2F ******>
          PGT LOCK    : 0 10C300 000000
          PGT STATE   : 0 028000 010021 DYN PATH SEL DLPS: 1, PGT VERSION=1, CONNECT
          IOP INFO1 : 0 000000 002200 DLPAW INDEX=22
          IOP INFO2 : 0 000000 001A02 DLPAW INDEX=1A, DLP MASK CONTAINS DLP INDEXES:1.
          DLP INFO1   : 0 000000 000400 DLP ACTIVE COUNT=00
            ( UNIT QUEUE IS EMPTY )
```

Figure 4–11. IO Command Output

```
INPUT: IO UNIT LOGICAL 51 UINFO NAMES

(PHYSICALUNITINFO   ASD INDEX = 03178; PHYSICALUNITINFO1  ASD INDEX = 03179; PHYSICALUNITINFO2  ASD INDEX = 0317A;
 PHYSICALUNITINFO3  ASD INDEX = 0317B; PHYSICALUNITINFO4  ASD INDEX = 0317C; PHYSICALUNITINFO5  ASD INDEX = 0317D;
 PHYSICALUNITINFO6  ASD INDEX = 0317E; PHYSICALUNITINFO7  ASD INDEX = 0317F; PHYSICALUNITINFO8  ASD INDEX = 03180;
 PHYSICALUNITINFO9  ASD INDEX = 03181; LOGICALUNITINFO    ASD INDEX = 03186; PHYSICALUNITQUEUES ASD INDEX = 03182;
 HAQDOPEVECTOR      ASD INDEX = 03175; RQDOPEVECTOR       ASD INDEX = 03177; ECQDOPEVECTOR      ASD INDEX = 03187;
 ACTIVEHCBHEAD      ASD INDEX = 040AC; PHYSICALUNITLOCKWORDS ASD INDEX = 03176)

********** LP 5  INTERNAL UNIT 4  LOGICAL UNIT 51 **********
0 1D0080 01C000  UNIT[033] LP5 (TRAIN PRINTER)  TRAINID=16, TRAIN NAME=EBCDIC96, LPKIND=TRANSLATE TABLE, MODE=8-BIT
                              SCRATCH, AUTOBACKUP
          UINFO [033] 0 000000 000000
0 000000 000000  UNITIOERR[033]
0 001300 000007  PHYSICALUNITINFO[0004] CURRENT HDU = 0, DLP-RELATIVE NUMBER = 0, TRAIN PRINTER,
0 427010 A08000  PHYSICALUNITINFO1[0004] QUEUE COUNT  = 1, IN PARTITION, CURRENT STATUS, CHECK STATUS,
                                        TEST WAIT OP, ON LINE, MONITORING STATUS, READY, HAS STATUS, INHIBIT TEST WAIT COUNT = 0,
                                        RING WALK COUNT = 0,
0 006700 650004  PHYSICALUNITINFO2[0004] UNIT PORT = 103, UNIT DLP = 101,
0 00007F FF0065  PHYSICALUNITINFO3[0004] PHYSICAL STOP COUNT = 0, LAST UNIT ON EXCHANGE, FIRST DLP SERVING = 101,
0 003500 005000  PHYSICALUNITINFO4[0004] CTP STATUS = 0, NORMAL TIMEOUT INTERVAL (80),
0 003300 000006  PHYSICALUNITINFO5[0004] MAJOR TYPE IS UNIT,
0 002800 040006  PHYSICALUNITINFO5[0033]
0 00007F FF0003  PHYSICALUNITINFO6[0004] LAST DEVICE IN CLASS 4, NEXT DEVICE BY TYPE = 3,
0 000000 000005  PHYSICALUNITINFO7[0004]
0 000000 000000  PHYSICALUNITINFO8[0004]
0 000000 000000  PHYSICALUNITLOCKWORDS[0004]
5 800000 403250  PHYSICALUNITQUEUES[0004] 5 C00000 803244  0 000000 000000  0 000000 000000  0 000000 000000

5 C00000 803244  COMMAND QUEUE 0:  QUEUE IS RUNNING, STOP COUNT = 0, IOS OUTSTANDING = 0, SOFT QUEUE LENGTH = 0,
              0 10CC00 000104  CQ CONTROL WORD:  ACTIVE COUNT = 0, ACTIVE LIMIT = 1, DLP PRESENT,
               0 000000 000000  HEAD IOCB LINK
               0 000000 000000  TAIL IOCB LINK
               5 C00000 8031A3  DLP QUEUE HEAD POINTER
               0 000000 000000


IOCB'S IN RQ FOR CQ 0:

                         * HCB ANALYSIS *                (HCB ASD INDEX = 4'032E6'), (HCB @ 4'00120728')
                         CONTROL WORD:  0 10CB00 00C409 DLP COMMAND, INTERRUPT PROCESSOR ON FINISH, TESTOP, CHARACTER MODE,
                                                        FORCE TAGS TO SINGLE, QUEUE AT HEAD, IGNORE COUNT ERROR, CQ IMMEDIATE
                         DLP COMMAND:     2200
                         HDU RESULT:    0 000000 000000
                         DLP RESULT:      0041
                         C-Q POINTER:   5 C00000 803244
                         R-Q POINTER:   5 C00000 8031A7
                         MISCELLANEOUS: TEST WAIT EPILOG, PHYSICAL UNIT NUMBER = 4'0004', ACTIVE, RETRY COUNT = 0,
                                        SUCCESSOR = 5 C00001 8032ED, MASK = 844110 100000, I/O TIME > 1 SECOND,
                                        PATH = 006700 650004
                         RAW HCB:       0 10CB00 00C409 0 000000 000000 5 C00000 803244 5 C00001 8032E6 5 C00001 8032ED
                                        5 C00000 8031A3 0 000002 00000C 5 E40001 3032E6 5 E40001 5032E6 0 844110 100000
                                        5 C00000 8031A7 0 030000 12073B 0 000000 000000 0 040002 0032E6 0 000013 D2DA03
                                        0 000000 000000 0 000000 000000 0 900000 004800 5 C00001 8032E5 0 220000 000000
                                        0 000000 000000 0 004100 000000 0 000000 000000 0 006700 650004
0 000000 003030  LOGICALUNITINFO[033]
                         NOT HELD (RESERVATION = 3, NOW SERVING = 3)

********** DLP1003  INTERNAL DEVICE 101  **********
0 7FFF10 004133  PHYSICALUNITINFO[0065] DLP WEIGHT = 1, IN PARTITION, B9246-20 2000 LPM TRAIN PRINTER,
                                        SLOT IN BASE = 3,
0 000400 637FFF  PHYSICALUNITINFO1[0065] FIRST UNIT SERVED = 4, CONTAINING BASE = 99, LAST DLP ON EXCHANGE,
0 006600 200003  PHYSICALUNITINFO2[0065] NEXT DLP IN BASE = 102, DLP ADDRESS = HDU 0 @ HDP 0 @ MLI 0 @ LEM PORT 0 @ DLP 3 HRF = 2,
```

Figure 4–12. IO Command Output With NAMES Option

IOCB

The IOCB command prints the I/O control block (IOCB) specified by <simple address> or <ASD number>. If the specified area is not an IOCB, then DUMPANALYZER displays an error message.

```
<iocb>

— IOCB ─┬────────┬─<simple address>─┬──────────────────────────────┤
        ├─ AT ───┘                  │
        └─ VIA ──<ASD number>───────┘
```

The following text describes the meaning of each construct:

IOCB <simple address>	
IOCB AT <simple address>	The IOCB command analyzes both IOCBs and hardware control blocks (HCBs) for HDU systems, and only IOCBs for EMS systems. The word located at the <simple address> is a descriptor that can refer to either an IOCB or an HCB. If the descriptor refers to an IOCB and an HCB has been allocated for that IOCB then both structures are analyzed. Word three of the structure pointed to by the descriptor is checked to determine if the the descriptor is for an HCB or an IOCB. If the tag of the word is 5 (indicating that it is a data descriptor), then the word is assumed to be the HCB Self Pointer and the descriptor is assumed to point to an HCB; otherwise, the descriptor is assumed to point to an IOCB.
IOCB VIA <ASD number>	The IOCB located at <ASD number> is analyzed and printed.

Examples

Figure 4–13 shows example output from the IOCB command.

```
INPUT: IOCB VIA 377E

* SOFTWARE IOCB ANALYSIS *      (IOCB ASD INDEX = 4'0377E'), (IOCB @ 4'0002C9F8')
HCB CW:         0 10CB00 046420 DLP COMMAND, READ, CHARACTER MODE, FORCE TAGS TO SINGLE,
                                CONTINUE COUNT, IGNORE COUNT ERROR
IOCW:           0 1303C0 000000 READ, 8-BIT, MEMORY PROTECT
LOGICAL RD:     0 000000 000101 0 CHARACTERS TRANSFERRED, EOT, EXCEPTION
LAST RESULT:    END OF TAPE
DLP COMMAND:      8220
HDU RESULT:     0 000000 000000
DLP RESULT:       0001 0280
AREA DESC:      5 E40000 10118A (ASD INDEX = 4'0118A'), BASE ADDRESS = 4'0002CA43', WORD INDEX = 1,
                                BYTE INDEX = 0
MISCELLANEOUS:  IN PROCESS, ACTIVE, REQUEST LENGTH = 234 CHARACTERS, ERROR MASK = 4'08001',
                REQUESTOR = 10, OWNER STACK = 4'012', INITIATING STACK = 4'054',
                BUFFER DESCRIPTOR = 5 C4000F 00118A, FIB DESCRIPTOR = 0 000000 000000,
                EVENT REFERENCE = 5 E10005 503779, LOGICAL UNIT NUMBER = 4'0030',
                PHYSICAL UNIT NUMBER = 4'0007', RELATIVE CQ = 0, LAST PATH = 006700 690007,
                PORT = 4'67', DLP = 4'69', TIME CELL = 0, RETRY COUNT = 0, LOGGED RDS COUNT = 0
RAW IOCB:       0 10CB00 046420 0 000000 000000 0 1303C0 000000 0 000000 000101 0 200000 000101
                0 000000 000000 0 000002 00000C 5 E40001 40377E 5 E40001 60377E 0 000100 010000
                0 000000 000005 5 E40000 10118A 0 000000 0000EA 5 C4000F 00118A 0 000000 000000
                5 E10005 503779 0 000000 000000 0 006700 690007 0 012054 288001 0 003000 000006
                0 822000 000000 0 000000 000000 0 000102 800000 0 000000 000000 5 C00001 8032C7
                0 000000 000000 0 000000 300000
RAW HCB:        0 10CB00 046420 0 000000 000000 5 C00000 803247 5 C00001 8032C7 0 000000 000000
                5 C00000 8031A5 0 000002 00000C 5 E40001 40377E 5 E40001 60377E 0 000100 010000
                5 C00000 8031A7 5 E40000 10118A 0 000000 0000EA 0 040002 800020 0 000000 0018A3
                0 000000 000000 5 C00001 80377E 0 900000 007000 5 C00001 8032BD 0 2A0000 000000
                0 000000 000000 0 000000 000000 0 000000 000000 0 006700 690007
```

Figure 4-13. IOCB Command Output

IOTABLE

The IOTABLE command gives I/O information pertaining to the specified IOP. This information includes expansion of the IOCB queue pointed to by the I/O table. This command is valid only for dumps executed on Micro A, A 1, A 2, A 3, A 4, A 5, A 6, and A 10 systems.

```
<iotable>

— IOTABLE —< IOP number >————————————————————————————|
```

The following text describes the meaning of the variable:

<IOP number>	An integer between 0 and 7.

Example

Figure 4–13 shows example output from the IOTABLE command.

```
IOTABLE 2

##### I/O TABLE FOR IOP# 2 @ 00020347   #####
         I/O TABLE CW      :0 10C000 000000
         IOCB QUE HEAD PTR:5 C00000 3202F8
                                            ( IOCB QUEUE )
                                            IOCB QUE CW  :0 10C100 000000
                                            IOCB QUE HEAD:0 000000 000000
                                            IOCB QUE TAIL:0 000000 000000


         IOP QUE CW       :0 10C200 000000
         IOP QUE HEAD     :0 000000 000000
         IOP QUE TAIL     :0 000000 000000
         EMP DEST. SET    :0 000000 000400 EMP#10.
         IOP DEST. SET    :0 000000 000006 IOP #1, IOP#2.
         SCRATCH AREA     :5 E00001 C1D551
```

KEEP

The KEEP command causes the last command entered to be saved for future use. This command is stored in a 10-deep, first-in-first-out queue. Each entry has a number between 0 and 9, inclusive, which is used if the entry is recalled by the USE command. (Refer to the USE command.)

```
<keep>

— KEEP ─┬──────────┬──────────────────────────────────|
        └─<number>─┘
```

The following text describes the meaning of each construct:

KEEP	The last command entered is saved at the next available number between 0 and 9. The command is retrieved by the USE command.

continued

continued

KEEP <number>	Assigns a command to be stored at the specified number between 0 and 9.

For more information, see the discussion of the USE command in this section.

Example

The following is an example of the response to a KEEP command:

```
INPUT: KEEP
SAVED: #2
```

LIB

The LIB command analyzes and prints library information.

<lib>

```
— LIB ─┬─ AT ──<simple address>─┬──────────────────────────────┤
       ├─ VIA ──<ASD number>────┤
       ├─ MAP ─┬────────────────┤
       │       └─<number>───────┤
       └─ SL ───────────────────┘
```

The following text describes the meaning of each construct:

LIB AT <simple address>	
LIB VIA <ASD number>	Analyzes the contents of the indicated location of memory as a library structure.
LIB MAP	Displays the contents of the library map. Empty slots in the map are not displayed.
LIB MAP <number>	Displays the contents of a linked list chain in the library map, starting at the given index.
LIB SL	Displays the system library function definitions.

Examples

The following is an example of the response to the *LIB AT <simple address>* command:

```
INPUT: LIB AT 45A8
----  HEADER  ----
    LEVEL = 3, LOCKED.
    STACK INFORMATION:  IMP AT ØØ1D IN STK Ø9Ø = (*,ØØØ9).
----  USEINFO  ----
    LINKED TO STACKS: EXP AT ØØ19 IN STK ØA1.
----  AREAS  ----
    FREE        ØØ31
    USEINFO     ØØ2D
    STACKREF    ØØØ6
    IMPORTS     ØØØB
    EXPORTS     ØØØØ
    TYPES       ØØØE
    NAMES       ØØ12
    ATTRIBS     ØØ1A
----  IMPORT OBJECTS  ----
  ([V] = BY VALUE, [R] = BY REFERENCE, [N] = BY NAME)
    ALSOLINKED IS A INTEGER FUNCTION (1 PARAMETER);
        INTEGERIV];
        INDEX = 12, OBJECT = (*,ØØØB).
----  ATTRIBUTES  ----
    VALUE = -5 H 4ØØ ØØØØ ØØØØ5, INTNAME = L,
    TITLE = LIBRARY/MULTIPLELINKERROR.
----
```

The following is an example of the response to the *LIB MAP* command:

```
INPUT: LIB MAP 6

            MAP[6]  = STK Ø45, HDR ØØ17, NEXT = 5
            MAP[5]  = STK Ø44, HDR ØØ19, NEXT = 4
            MAP[4]  = STK Ø3F, HDR ØØ1B, NEXT = 3
            MAP[3]  = STK Ø3E, HDR ØØ1D, NEXT = 2
            MAP[2]  = STK Ø3A, HDR ØØ2Ø, NEXT = 1
            MAP[1]  = STK Ø36, HDR ØØ24
```

The following is an example of the response to the *LIB SL* command:

```
INPUT: LIB SL
SL BNASUPPORT        = *36/SYSTEM/BNASUPPORT
   ONLY 1, LINKCLASS 1
SL COMSSUPPORT       = *SYSTEM/COMS
   LINKCLASS 1, TRUSTED, HDR ØØ57
SL DATACOMSUPPORT    = *36/SYSTEM/DATACOMSUPPORT
   ONLY 1, LINKCLASS 1, HDR ØØ5A
SL GENERALSUPPORT    = *36/SYSTEM/GENERALSUPPORT
   HDR ØØ55
SL MCPSUPPORT        = ">> CURRENT MCP <<"
   MCPLIB, STK Ø1Ø, HDR FFFF
```

LINKCHECK

The LINKCHECK command examines each memory link in the in-use and available areas of memory. If an error is found, an error message is displayed and DUMPANALYZER recovers and continues. Note that LINKCHECK is valid only for complete memory dumps. Because the memory links are validated before the dumping process is initiated and the memory is not in any special order, LINKCHECK is disallowed for ALLINUSE and PARTIAL memory dumps.

```
<linkcheck>
— LINKCHECK ——————————————————————————————————————————————|

<AX options>
— ? — AX ─┬─ WHERE ─┬─────────────────────────────────────|
          ├─ STOP ──┤
          ├─ +T ────┤
          └─ -T ────┘
```

The following text describes the meaning of each option:

?AX WHERE	Asks where the LINKCHECK is.
?AX STOP	Stops the LINKCHECK and reprompts the user with a ":READY" message.
?AX –T	By default, LINKCHECK prints the contents of areas where link corruption occurred. *?AX –T* suppresses this printing.
?AX +T	Reenables the printing of areas where link corruption occurred.

Example

The following is an example of LINKCHECK output indicating that link corruption was found:

```
LINKCHECK

TAG ERR & 27621-2762Ø  :  Ø  ØØØØØØØØØØØØ
```

LINKS

The LINKS command prints the addresses and contents of each link of the memory area that contains a specified <simple address>. The analysis also includes the following memory area attributes, which are encoded in the links:

- Whether the area is available, save, csave (currently saved), or olay (overlayable)
- If in-use, whether it is code, data or read-only
- The area size
- The area MOM address
- Whether or not the MOM address for the area is in the MCP D[0] stack and, if the name of that cell is available, the D[0] MOM address name
- The area stack number
- If present, the link C RCW
- The memory priority of the area
- If the area is available, the RCW trace of FORGETSPACE

```
<links>

— LINKS ─┬─<simple address>─────┬─┬──────────┬─────────────────────────┤
         └─ VIA ─<ASD number>───┘ └─ EXPAND ─┘
```

The EXPAND option expands the fields of the associated ASD.

Examples

An example of the LINKS command output for an in-use area is given in Figure 4–14.

An example of the LINKS command output for an available area is given in Figure 4–15. Note that the RCW trace information appears only for dumps created on a system running a DIAGNOSTICS MCP.

```
INPUT: LINKS 4000 EXPAND
00003D19  INUSE AREA  DATA: LENGTH=04209(16905)MOM ADDR=000028AD (D[0] + 07AD)
ASD1[010D5]: [000410D9] 5 670000 003D1C
   ASD1_PRESENTTOPROCF          =  1
   ASD1_UNALTEREDF              =  0
   ASD1_NOTSTACKF               =  1
   ASD1_PRESENTFORIOF           =  1
   ASD1_NOTPAGEF                =  1
   ASD1_READONLYF               =  0
   ASD1_DONTRESIZEF             =  0
   ASD1_PAGEDF                  =  0
   ASD1_ADDRESSF                =  00003D1C

ASD2[010D5]: [0006A034] 3 160000 004209
   ASD2_SPACEUSAGEF          =  16  (PERMSAVEAREA)
   ASD2_LENGTHF              =  04209

ASD3[010D5]: [00092F8F] 1 012000 8407AD
  STACK NUMBER              = 012
  DISPLACEMENT              = 0008
  OFFSET                    = 07AD

ASD4[010D5]: [000BBEEA] 0 000000 001200
   ASD4_LINKASDINDEXF           =  00000
   ASD4_OLAYSTATEF              =  0 (NON OVERLAYABLE SEG)
   ASD4_OWNERSTACKF             =  012
   ASD4_VIRGINASDF              =  0
   ASD4_MARKEDFORWSSHERIFFF     =  0
   ASD4_UNLOCKDISPOSALF         =  0
   ASD4_LOCKBITF                =  0

LINK A : [00003D19] 6 E0420D 0010D5
  D[0] MOM ADDRESS NAME= RESCODE MEMORY
  SPACE USAGE               = (PERMSAVEAREA)
  INUSE_SEGSTATEF           =  SAVE AREA
  INUSE_ASDINDEXF           =  010D5
  INUSE_SIZEF               =  00420D

LINK C : [00003D1B] 3 000000 000000

LINK Z : [00007F25] 3 E0420D 000000
  INUSE_SIZEF               =  00420D
```

Figure 4–14. LINKS Command Output: In-Use Area

```
INPUT: LINKS 7E42DE

007E42D9  AVAILABLE AREA  : LENGTH = 00000016(22)
LINKS: 1 480000 000016   0 000000 D67275   0 000000 000000   0 000000 000000   0 000000 000000   ... 1 400000 000016
     RCW TRACE OF FORGETSPACE---
        STACK: 894
        @ 1005:037F:0 (04899700) FORGETSPACE
        @ 100F:00A9:4 (04700750) AMNESIA
        @ 1106:0ACB:1 (20881800) MUTATE
        @ 0248:003E:1
        @ 0247:0198:1
        @ 1207:0103:4 (24565550) BLOCKEXIT
        @ 0247:016D:4
        @ 10EA:06DB:1 (16562000) BOJEOJ
```

Figure 4–15. LINKS Command Output: Available Area

LOADXREF

The LOADXREF command associates a set of cross-reference information files with a segment dictionary stack or with the MCP. The cross-reference information files are used to display identifier name and compiler class along with stack variables. The LOADXREF command depends on the presence of an XREFSUPPORT system library. XREFSUPPORT must have been established as a support library with the SL (Support Library) system command.

If the code file was compiled without the LINEINFO option set, the identifier information cannot be displayed.

If LOADXREF has been requested for a stack and the IDNAMES option has been requested in the MODE command, the BINDINFO information is used to identify stack variables instead of the cross-reference information files.

If BINDINFO information does not exist for an environment, then the cross-reference information files are used. The LOADXREF command cannot be used to get identifier names for environments that are bound into the MCP.

```
<loadxref>

— LOADXREF —┬— MCP ——————┬— " —<code file name>— " —┬———————————┬———|
            └—<number>———┘                           └— : — C ———┘
```

The following text describes the meaning of each construct:

LOADXREF MCP "<code file name>"	The cross-reference information files, XREFFILES/<code file name>/DECS, XREFFILES/<code file name>/REFS, and XREFFILES/<code file name>/LOCS are associated with the MCP code file and then used to display identifier name and compiler class along with stack variables.
LOADXREF <number> "<code file name>"	The cross-reference information files, XREFFILES/<code file name>/DECS, XREFFILES/<code file name>/REFS, and XREFFILES/<code file name>/LOCS are associated with the specified stack if it is a segment dictionary stack, or with the segment dictionary stack for the specified stack. Cross-reference information files for a stack whose segment dictionary stack is the MCP stack can be loaded only by using the *LOADXREF MCP* form of the command.
: C	The XREFFILES/<code file name>/LOCS file is needed by the LOADXREF command to process cross-reference information files more efficiently. Use this option to request the creation of a LOCS file in the user's directory if a valid LOCS file is not found. A LOCS file can also be created by running the INTERACTIVEXREF utility with the SW1 task attribute set to true. For further information about INTERACTIVEXREF, refer to the *A Series System Software Utilities Operations Reference Manual*

For more information, see the discussion of the MODE command in this section.

LOCKS

Locks or events are defined as being in one of two states: normal or abnormal. An abnormal event is one that is procured or has a stack waiting on it. All EVENTs and EVENT ARRAYs declared in the MCP outer block are considered, but only abnormal ones are listed. Hard locks are not reported.

<locks>

```
— LOCKS ————————————————————————————————————————————|
```

Note: *The MCP global events and event arrays are located using BINDINFO data in the MCP code file. If the analyzed code file does not match the dumping MCP code file, the event analysis might be incorrect.*

Example

Figure 4–16 shows example output for the LOCKS command.

```
INPUT: LOCKS
MCP EVENTS (PROCURED OR WITH WAITERS)

 G AB8    { 000660 000008  2 000A03 D01A18  UPDATE FLOG LOCK
          {UNAVAILABLE, PROCURED BY STK 066 @ 1A18:003D:5 (85899872) UPDATE_FLOG )
 G 4F8    { 0002F0 000009  2 000806 1012F3  DCINITIALLOCK
          {UNAVAILABLE, HAPPENED, PROCURED BY STK 02F @ 12F3:0061:4 (49253274) BUILD_DC_TABLES )
 G 56A    { 000370 000009  2 000400 801AAB  HEADERLOCKS[0006]
          {UNAVAILABLE, HAPPENED, PROCURED BY STK 037 @ 1AAB:0008:2 (94560600)  )
          { 000480 000009  2 000A09 401331  HEADERLOCKS[005D]
          {UNAVAILABLE, HAPPENED, PROCURED BY STK 04B @ 1331:0094:5 (37991900) RELEASEHEADER )
 G 9AB    { 000000 400380  } 000000 000000  PSE[0005]
          {WAITING: STK 038}
          { 000000 300380  } 000000 000000  PSE[0006]
          {WAITING: STK 038}
 G 4C3    { 0002F0 000008  } 000206 4012F3  DCPREFIXLOCK
          {UNAVAILABLE, PROCURED BY STK 02F @ 12F3:0064:1 (49253280) BUILD_DC_TABLES )
 G 8B7    { 000000 1004C0  } 000000 000000  SLOGMONITOREVENT
          {WAITING: STK 04C}
 G EE8    { 000000 100680  } 000000 000000  AHW_ATTENTION
          {WAITING: STK 068}
 G 6AB    { 000310 000008  } 00005C E011CF  CTRLWARELOCK
          {UNAVAILABLE, PROCURED BY STK 031 @ 11CF:05CE:0 (32195290) PATHRES )
 G 1AA    { 000000 100760  } 000000 000000  OLAYSPACEWANTED
          {WAITING: STK 076}
 G 32D    { 000000 000330  } 000000 000000  ANABOLEVENT
          {WAITING: STK 033}
 G 9AB    { 000000 400380  } 000000 000000  LISTS_E[0005]
          {WAITING: STK 038}
          { 000000 300380  } 000000 000000  LISTS_E[0006]
          {WAITING: STK 038}
```

Figure 4–16. LOCKS Command Output

MASK

The MASK command is used to set or examine the mask register. The MASK command is used in conjunction with the PATTERN and SEARCH commands to search a dump for all words that contain a particular pattern of bits. The mask register modifies the pattern in the pattern register by masking certain bits in the pattern. The SEARCH command searches for all words which match that part of the pattern which is not masked. (Refer to the PATTERN and SEARCH commands for further information.)

```
<mask>

— MASK ─┬────────────────┬────────────────────────────────────────┤
        ├─<simple value>─┤
        ├─ ABSENTDESC ───┤
        ├─ ABSENTCOPY ───┤
        └─ MOMDESC ──────┘
```

The following text describes the meaning of each construct:

MASK

MASK specified by itself displays the contents of the mask register. The default for the MASK register is 7 FFFFFF FFFFFF.

MASK <simple value>

Places a mask that can be expressed as a given <simple value> into the mask register. A mask is a distribution of 1s and 0s in a 48-bit word and its 4-bit tag. The mask indicates which bits are significant and which are irrelevant within the current pattern set by the PATTERN command. All bits in a mask that have a value of 1 are significant digits in the pattern. All bits in a mask that have a value of 0 are disregarded in the pattern. The mask most recently specified is stored in the mask register.

The mask can include the 4-bit tag for a word. To specify a TAG, the concatenation form of simple value should be used; input would specify 12 hexadecimal digits *& <number> TAG*. The tag value should be no larger than 15; however, if it is larger than 15, the tag value is placed in the tag using modulus 16. If no tag is specified, the tag is assumed to be 0.

Although the mask represents 48 binary bits and a tag value of 4 binary bits, it is displayed and is most often set in hexadecimal. Refer to Figure 4–17 for a diagram of a word that contains a mask value, with its hexadecimal equivalent stated below it.

	bit#													
1	47	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	1	1	1	3
1	46	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	1	1	1	2
1	45	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	1	1	1	1	1
1	44	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	1	1	1	1	Ø
F		Ø	Ø	Ø	Ø	Ø	Ø	Ø	3	F	F	F	F	

Figure 4–17. Mask Word Example

To indicate that the last 18 bits of the pattern word and its tag are significant, the corresponding mask command would be MASK 3FFFF & F TAG. (It is understood that there are seven leading zeros.) The mask register would display its contents as F 000000 03FFFF.

The default for the MASK register is F FFFFFF FFFFFF; in this case, all bits including the four tag bits (represented by the leading F) are set to one, and therefore are significant in the pattern register.

MASK ABSENTCOPY
MASK ABSENTDESC
MASK MOMDESC

A mask value appropriate for finding descriptors is placed in the mask register. The same mask value is generated, regardless of whether the ABSENTCOPY, ABSENTDESC, or MOMDESC option is specified. If the dump was created on a system running E-mode level Beta, the mask value is F C00000 000000. If the dump was created on a system running E-mode level Gamma, the mask value is C 000000 000000. As of this publication, only A 16 systems run E-mode level Gamma. The Micro A, A 1, A 2, A 3, A 4, A 5, A 6, A 9, A 10, A 12, A 15, and A 17 run E-mode level Beta.

Examples

```
INPUT: MASK
MASK: F FFFFFF FFFFFFF

INPUT: MASK ABSENTCOPY
MASK:  F CØØØØØ ØØØØØØ

INPUT: MASK MOMDESC
MASK:  F CØØØØØ ØØØØØØ

INPUT: MASK 123456789ABC
MASK:  Ø 123456 789ABC

INPUT: MASK 1324 & 5 TAG
MASK:  5 ØØØØØØ ØØ1324
```

For more information, see the discussion of the PATTERN and SEARCH commands in this section.

MD

The MD (Memory Dump) command dumps the contents of a group of addresses in memory with no analysis.

```
<md>
— MD —<multiple addresses>———————————————————————————————————————|
```

Example

A small subset of the raw dump analysis, which is output from the MD command follows:

```
INPUT: MD C119D to C1200

000C119D(00000000)  0 C00040 190C6D  7 700120 080C7F          ...
000C11A4(00000007)  0 000000 000000  0 300023 000096          ...
000C11AB(0000000E)  0 400000 00CF62  0 000000 000074          ...
000C11B2(00000015)  0 000000 000000  0 000000 000000          ...
000C11B9(0000001C)  0 000000 000000  THRU 000C11DB(0000003E) ...
000C11DC(0000003F)  1 4000A0 000000  6 C0001C 180C5A          ...
000C11E3(00000046)  3 B20695 B4AE42  3 4E601A AFB208          ...
...
```

The word THRU in the output identifies a sequence of repeated words. For example, in the block of zero operands, 000C11B9 and 000C11DB are the addresses of the first and last zero operands, in hex. The hex offsets in parentheses such as (0000001C) and (0000003E) represent offsets that are relative to the beginning address requested. Each word is broken into tag, upper half, and lower half.

MDCODE (HDU Systems)

Note *The MDCODE command has been deimplemented. This command had meaning only on systems with ASN memory, which is not supported on the Mark 3.9 system software release and later releases.*

MEM

The MEM (memory) command lists the memory modules present and the range of addresses contained in each module on the system when a dump was created.

```
<mem>
— MEM ———————————————————————————————————————————————————————————|
```

Example

The following is an example of output from the MEM command:

```
INPUT: MEM


MEMORY MODULES IN DUMP (MODULE = 128K WORDS)
        Ø (ØØØØØØØØ) -  63 (ØØ7FFFFF)
TOTAL MODS READY: 64
```

MESSAGES

The MESSAGES command finds, analyzes, and prints the DATA COMM messages in the system. This is useful for determining

- If any stations or MCSs have loaded the system with messages
- The amount of DATA COMM traffic to various stations.
- If an MCS is recalling messages correctly

Note that this command works only for COMPLETE or ALLINUSE memory dumps; it is rejected for PARTIAL memory dumps.

Due to the nature of DUMPANALYZER and the way it accesses memory, a message analysis sometimes takes quite a while, especially if there are a lot of messages or large ASD table sizes. To check on the status of the command, enter the following command:

```
?AX WHERE
```

DUMPANALYZER then displays information regarding the searching process.

```
<messages>

— MESSAGES ─┬────────────────────────────────────────────────┤
            ├─ EXPAND ─────────────────┤
            ├─ ALL ────────────────────┤
            └─ LSN ── <decimal number> ┘
```

The following text describes the meaning of each construct:

MESSAGES	MESSAGES finds and counts all data comm messages. For each station, it displays the number of queued messages along with the name of the controlling MCS. Note that if the MCP data comm tables were not in memory when the dump was created, some of this information might not be available.
MESSAGES EXPAND	Along with the summary previously described, all DATA COMM messages are printed with the header decoded and the message text displayed in alpha characters.
MESSAGES LSN <decimal number>	The messages queued for the specified station are printed.

continued

continued

MESSAGES ALL	All message areas are printed. Data comm messages are printed as previously described, and other message areas are printed in Hex and alpha characters. This report is used for detailed analysis of memory area usage.

Example

The following is an example of the output from the MESSAGES command:

```
INPUT: MESSAGES
S U M M A R Y   O F   M E S S A G E   A R E A S
756 TOTAL AREAS (4736 WORDS)
745 NON-DATACOMM AREAS (4472 WORDS)
11 DATACOMM MESSAGES (264 WORDS)
1 MESSAGES FOR UNKNOWN LSN'S.  (13 WORDS)
1 MESSAGES FOR LSN 3 (8 WORDS)  MCS = 1 SYSTEM/COMS ON 39
3 MESSAGES FOR LSN 155 (2Ø1 WORDS)  MCS = 1 SYSTEM/COMS ON 39
2 MESSAGES FOR LSN 169 (14 WORDS)  MCS = Ø
2 MESSAGES FOR LSN 171 (14 WORDS)  MCS = Ø
1 MESSAGES FOR LSN 172 (7 WORDS)  MCS = Ø
1 MESSAGES FOR LSN 173 (7 WORDS)  MCS = Ø
```

MIX

The MIX command displays the contents of the process stack or segment dictionary stack associated with the supplied mix number. The results are the same as for the STACK command.

```
<mix>

— MIX —<decimal number>─┬──────┬──────────────────────────┤
                        └─ SD ─┘
```

The following text describes the meaning of each construct:

MIX <decimal number>	The contents of the task stack associated with the supplied mix number (<decimal number>) are displayed.
MIX <decimal number> SD	The contents of the segment dictionary stack associated with the supplied mix number are displayed.

For more information, see the discussion of the STACK command in this section.

MODE

The MODE command allows the user to control the mode in which words are expanded in the MIX and STACK commands. It is also used to control how <simple value>s are expanded in the PV command. (Refer to the PV command.) Some of the mode options might not be available depending on whether the area was included in the memory dump. For example, if the contents of an array have not been dumped, the command MODE + ARRAY cannot display the contents of the array.

```
<mode>

— MODE ─┬─ ? ──────────────────┬───────────────────────────────────────┤
        ├─ + ─┬─<mode option>──┘
        └─ - ─┘

<mode option>

──┬─ ALL ──────────────────────────────────────────────────────────────┤
  ├─ ARRAY ───┤
  ├─ BCL ─────┤
  ├─ CB ──────┤
  ├─ CODE ────┤
  ├─ DEC ─────┤
  ├─ EBC ─────┤
  ├─ FIB ─────┤
  ├─ FILE ────┤
  ├─ IDNAMES ─┤
  ├─ IOCB ────┤
  ├─ LIB ─────┤
  ├─ LOCK ────┤
  ├─ OCT ─────┤
  ├─ PCW ─────┤
  ├─ PIB ─────┤
  └─ TAB ─────┘
```

The following text describes the meaning of each construct:

MODE?	The current mode or modes are displayed.
MODE+ <mode option>	The specified mode is added.
MODE– <mode option>	The specified mode is deleted.
ALL	The ALL modifier is inclusive of all of the following modes.
ARRAY	If ARRAY mode is used, mom descriptors are expanded subject to the limit established by the ARRAYLIMIT command. If the array is multidimensional, each dimension is expanded as it is encountered. Information concerning the current level is printed in the left margin when multidimensional arrays are expanded.
BCL	If BCL mode is used, all operands (Tag 0 words) are expanded to show both the hexadecimal representation and the Common Language (BCL) representation of the operand.
CB	If CB mode is used, descriptors referencing connection blocks are expanded.

continued

continued

CODE	If the CODE option is used, DUMPANALYZER expands and lists code areas and read-only data areas. When analyzing a stack, the code for return control words (RCWs) is expanded into mnemonics for the operators and names of items at the D[0] level. Op codes for all systems are recognized. An op code that is not defined for any of the systems is displayed in the form m*hh, where hh is the code syllable in hexadecimal and m is a letter indicating the context in which the syllable was encountered: • P: Primary (normal context) • V: Variant (right after code 95) • E: Edit (right after EXSD, EXSU, EXPU)
DEC	If DEC mode is used, all operands (Tag 0 words) are expanded to show both the hexadecimal representation and the decimal representation of the operand.
EBC	If EBCDIC mode is used, all operands (Tag 0 words) are expanded to show both the hexadecimal representation and the EBCDIC representation of the operand.
FIB	Expansion of FIBs in the stack is controlled by the FIB option. Buffers are dumped and analyzed.
FILE	FIB and FILE are synonyms.
IDNAMES	The IDNAMES mode is used to display identifier name and compiler class information along with stack variables. When the IDNAMES mode is used, the code file corresponding to a return control word (RCW) is checked to determine if BINDINFO information is present. Complete BINDINFO information must be present to get a display. Currently, only ALGOL, FORTRAN, COBOL, COBOL74, and PL/I support complete BINDINFO. To create BINDINFO information in COBOL or COBOL74, an external procedure declaration must be made. ALGOL, FORTRAN, and PL/I generate complete BINDINFO unless the program was compiled with the compiler control option NOBINDINFO set. FORTRAN77 generates partial BINDINFO information when the compiler control option BINDINFO is set. IDNAMES mode cannot be used for MCP environments; instead, the LOADXREF command must be used. There is no mechanism for getting identifier names for environments that are bound into the MCP.

continued

continued

If the code file cannot be found, an RSVP of NO FILE is displayed. The following responses are possible:

- OF: This response cancels the IDNAMES option and the following message is displayed: CODEFILE NOT AVAILABLE. The line numbers and identifier names are not shown.
- FA: This response changes the file attribute of the code file.

  If the code file that is found has a different creation time or date than the code file that was running at the time the dump was created, the message WRONG CODE FILE is displayed and the user must answer with one of the following:

  - OK: This response causes the code file to be used anyway.
  - RESTART: This response causes CODEFILE to be used as the code file. If CODEFILE cannot be found, the "NO FILE" RSVP described above is displayed.

When a correct environment is not found, no identifier name or class information can be displayed. Two conditions can produce an incorrect environment. Those conditions are

- When ALGOL was compiled with the BEGINSEGMENT/ENDSEGMENT compiler control options set.
- When the use of entry points created packed segments in PL/I.

IOCB

If IOCB mode is used, descriptors in the stack referencing an IOCB are expanded.

LIB

If the LIB option is used, library structures are expanded and analyzed. This applies to both library templates and library directories. (Refer to the LIB command.)

LOCK

If the LOCK option is used, the specified value is analyzed to determine whether it is a ***hard lock*** (tag of 0) or a ***soft lock*** (tag of 2).

OCT

If OCT mode is used, all operands (Tag 0 words) are expanded to show both the hexadecimal representation and the octal representation of the operand.

PIB

If PIB mode is used, arrays that have an oddball field of PIB in their memory links are displayed as PIBs.

PCW

Program Control Word. A word is analyzed as if it were a program control word. The expansion of PCWs in stacks is optional; normally, PCWs are not expanded, but they are expanded if PCW is set.

continued

continued

TAB	If TAB mode is used, arrays that have an oddball field of TAB in their memory links are displayed as TABS.

For more information, see the discussions of the following commands in this section:

- PV
- LIB
- MODE

Example

The following is an example of the response to a MODE command:

```
INPUT: MODE + ALL
MODES: SET: EBC BCL DEC OCT PCW LOCK CODE FIB ARRAY LIB IOCB PIB TAB
            IDNAMES CB
       RESET:
```

MSCW

The MSCW command helps to analyze stacks that have a corrupt mark stack control word (MSCW) or are in a state in which DUMPANALYZER cannot analyze them correctly.

<mscw>

```
— MSCW — FOR —<number>—┬— AT —<number>—┬——————————————————————|
                       └— ? ————————————┘
```

The following text describes the meaning of each construct:

MSCW FOR <number> AT <number>	The first <number> is the number of the specified stack. The second <number> is an offset, which must be the location of a valid MSCW in the stack. All MSCWs below this MSCW must also be valid. The MSCW indicated by <number> is marked off in the stack, so that when the stack is examined the following message appears in the MSCW's place: "MSCW ASSUMED AT THIS POINT BY USER SPECIFICATION" An MSCW is reset by setting the second <number> to zero; that is, the offset is zero.
MSCW FOR <number> ?	The setting of the MSCW for the specified stack is interrogated.

Example

The following is an example of an MSCW command and the reply:

```
INPUT: MSCW FOR 67 AT 51
STACK 067 MARKED MSCW AT 00051
```

NAMES

The NAMES command prints the entire list of MCP names and addresses. SORT intrinsics are invoked to sort the names alphabetically. While the sort is executing, a HI (Cause EXCEPTIONEVENT) system command causes DUMPANALYZER to display the message "SORTING MCP NAMES".

<names>

```
— NAMES ——————————————————————————————————————————|
```

Example

Figure 4–18 shows example output for the NAMES command.

```
INPUT: NAMES

                     IDENTIFIER-ORDERED MCP STACK ADDRESSES FOR *SYSTEM/MCP/39021H ON DISK

PROCEDURE-ADDRESSES ARE SHOWN AS PCW/SEGMENT

00C5      A_SERIES_IO_ALIAS         00C5      A_SERIES_IO_CELL          00C6      A_SERIES_IO2_ALIAS        00C6      A_SERIES_IO2_CELL
    /0731 AAMLSDUMMY                0501      AAUDITRELEASE_SUFFIX      0EC7/12F1 ABNORMALTERMINATIONMESSER
    /17C9 ABORT_SELECTION               /189A ABORTALLOPENS             0B83/1AA9 ABORTJOBLOG               0A9A      ABQLOCK
0319/1112 ACCEPT                    0262      ACCEPT_ANY_PROCS          031A/1113 ACCEPTAMSG                0E98/12B6 ACCEPTME
0D34      ACKNOWLEDGEDMSGLOCK       0D33      ACKNOWLEDGEDMSGQUEUE      0D31/112D ACKNOWLEDGEPRIMITIVEMESSAGES
    /112D ACKNOWLEDGEPRIMITIVEMESSAGES                                      /1021 ACQUIRE                   /101E ACQUIRE
    /1083 ACQUIREUNITS                  /107C ACQUIREUNITS              00E9/1083 ACQUIREUNITS              009C/101E ACQUIRE
00FF      ACQUIREUNITWORD           027F/101F ACTIVATE_OBJECT               /101F ACTIVATE_OBJECT               /1021 ACTIVATE_OBJECT
    /1A51 ACTIVECHECK               1C20      ACTIVEKF                  19C7/19C6 ADD_DIRECTORY_ENTRY       06CD/1599 ADD_DSS_TO_RESULT
1912      ADD_PROVIDER_GROUP_LOCAL_PCW                                  1912      ADD_PROVIDER_GROUP_PCW        /17DD ADD_PROVIDER_GUTS
1912      ADD_PROVIDER_PCW          06CE/14A1 ADD_RESULT                06A0/148E ADDALOGICALUNIT           02D9/10D6 ADDGRAPHEDGE
095C/189A ADDHOST_JACKET                /189A ADDHOST                       /189A ADDHOST_JACKET              /16AE ADDHOST_JACKET
0216/1012 ADDSTACKSEARCHEDGEGUTS    07EA/164D ADDSTANDBYHLUNIT              /173B ADJUST_PROV_LIST_ELEMENT_SIZES
05F8      ADLOCK                    0F42      ADM_STARTER               0F41      ADMINISTRATORS            02A9      AEVPIB
0EE8      AHW_ATTENTION             0EE6      AHW_COUNT                 0EE4      AHW_ENTRY                 0EEA      AHW_HANDLERRUNNING
0EEB/1330 AHW_HANDLER               0EE5      AHW_HEAD                  0EE7      AHW_LOCK                  0133/1097 AITTNSERT
0F43      ALA                       02AF      ALARMCLOCKTIME            0F44      ALE                       0134/100A ALLOCATE_ARRAY
0135/1007 ALLOCATE_CODE_PARAMETERIZED_STRUCTURE                             /17EC ALLOCATE_FLAGS
    /17E9 ALLOCATE_IOCHARS_STRUCTURE                                    0136/100A ALLOCATE_MCP_ARRAY        0137/100A ALLOCATE_MCP_STRUCTURE

                     ADDRESS-ORDERED MCP-STACK IDENTIFIERS FOR *SYSTEM/MCP/39021H ON DISK

/ DENOTES CODE SEGMENT

/0001 WEXTFUNC                 0002 STACK                     0003 ZAPHI                     0004 WORK
/0005 ZAPHI                    0007 ARRAYDEC                  0008 GAR_INFO_CELL             0009 LINEINFODESC
 000A GAR_NAME_CELL            000B GAR_CODE_OFFSET_CELL      000C DABMEM                    0014 MEMDUMP_PROC
 001C PIO                      004F GETTTGOING                0050 KERNEL_AR_REF             0077 DCQUEHANDLER
 0078 OMRREQUESTORMASK         0079 BLOCKEXIT                 007A GOTOSOLVER                007B MYJOB
 007C FREEZELIBRARY            007D PROGRAMDUMP               007E TIMEINTRINSIC             007F CLOSE
 0080 POTL                     0081 POTM                      0082 POTH                      0083 ATTRIBUTEGRABBER
 0084 LONGSETBITS              0085 TRUTHSETS                 0086 ATTRIBUTEHANDLER          0087 USERIOERROR
 0088 LOADCONTROL              0089 SET_GET_LIBRARYSTATUS     008A SORT                      008B RESIZEANDDEALLOCATE
 008C CANCELLIBRARY            008D OPENP                     008E DELIVERY                  008F DESCRIPTORSIZE
 0090 LINKLIBRARY              0091 EBCTOHEX                  0092 UNRAVEL                   0093 MUTATE
 0094 MYSELFER                 0095 CONTINUER                 0096 CLOSEP                    0097 FORKCONTROLCARD
 0098 FIXHANDLER               0099 DIRECTOR                  009A CAUSEP                    009B SETORRESET
 009C ACQUIRE                  009D RELINQUISH                009E COMBINEPPBS               009F FORKHANDLER
 00A0 EBCTOASC                 00A1 ASCTOHEX                  00A2 ASCTOEBC                  00A3 HEXTOEBCDIC
 00A4 HEXTOASCII               00A5 READLOCKTIMEOUT           00A6 CLOCKOFFPCW               00A7 STATCLOCKON
 00A8 STATCLOCKRESUME          00A9 STATCLOCKSUSPEND          00AA GETSTRINGAREA             00AB GETSTRINGPOOLSIZE
 00AC RESETSTRINGPOOLSIZE      00AD ARRAYSEARCHP              00AE GETLIBATTRIBUTES          00AF SETLIBATTRIBUTES
 00B0 HIGHESTPNUM              00B1 HAPPENEDP                 00B2 AVAILABLEP                00B3 MULTIWAIT
 00B4 SIMPLEWAIT               00B5 DELINKLIBRARY             00B6 DESC_HIDING               00B7 ILOK_OK
 00B8 ILOK_STATUS              00B9 ILOK_BREAK                00BA ILOK_ARROGATE             00BB ILOK_LOCKING
 00BC ILOK_UNLOCKING           00BD FA_JACKET                 00BE MCPHANDLEERROR            00BF FILEATTRIBUTEHANDLER
 00C0 FILEATTRIBUTEGRABBER     00C1 BADUNLOCKDUMP             00C2 JUSTEXITANDUPDATE         00C3 PURGE_SEB_INCLUDING_MY_AR
```

Figure 4–18. NAMES Command Output

NSP

The NSP command is similar to the DC command except that it analyzes only the NSP tables, not the DCC tables.

```
<nsp>

— NSP ─┬───────┬──────────────────────────────────────────┤
       └─ MSG ─┘
```

The MSG option causes messages in nontanked data comm queues in the DCALGOL queue to be analyzed.

For more information, see the discussion of the DC command in this section.

Example

Figure 4–19 shows example output for the NSP command.

```
INPUT: NSP

                                                    DATA COMMUNICATIONS ANALYSIS
NSP DATACOM CONFIGURATION:
    MAXIMUM LSN                                 = 712
    CONFIGURED : RELATIVE NSP NUMBER(S) = 0, 1, 2, 3, 4
    INITIALIZED: RELATIVE NSP NUMBER(S) = 0, 1, 3, 4
    NSP0107/00 : UNIT TAKEN, MAXLINES=15, DCC SNR=06E
    NSP0110/01 : UNIT TAKEN, MAXLINES=15, DCC SNR=06D
    NSP0109/03 : UNIT TAKEN, MAXLINES=31, DCC SNR=06C
    NSP0111/04 : UNIT TAKEN, MAXLINES=47, DCC SNR=06B

NSP TABLE ANALYSIS FOR NSP 0
=== ===== ======== === === =

NSPSTATUS 0:0 006B0F C0006E  MAXLINES=15, UNIT=107, UNIT TAKEN, DCC SNR = 06E
  NSPMASTER 0: 5 800001 0043B5     (DATA DESC TO LINE VECTOR)
  LINE 0 DESC  : 0 000D00 1002FF
  LINE 1 DESC  : 0 000D00 100302
  LINE 2 DESC  : 0 000000 200000
  LINE 3 DESC  : 0 000000 200000
  LINE 4 DESC  : 0 000000 200000
  LINE 5 DESC  : 0 000000 200000
  LINE 6 DESC  : 0 000000 200000
  LINE 7 DESC  : 0 000000 200000
  LINE 8 DESC  : 0 000000 200000
  LINE 9 DESC  : 0 000000 200000
  LINE 10 DESC : 0 000000 200000
  LINE 11 DESC : 0 000000 200000
  LINE 12 DESC : 0 000000 200000
  LINE 13 DESC : 0 000000 200000
  LINE 14 DESC : 0 000000 200000
  LINE 15 DESC : 0 000000 200000

NSP TABLE ANALYSIS FOR NSP 1
=== ===== ======== === === =

NSPSTATUS 1:0 006E0F C0006D  MAXLINES=15, UNIT=110, UNIT TAKEN, DCC SNR = 06D
  NSPMASTER 1: 5 800001 00438E     (DATA DESC TO LINE VECTOR)
  LINE 0 DESC  : 0 001000 000305
  LINE 1 DESC  : 0 000A00 100316
  LINE 2 DESC  : 0 000800 000327
  LINE 3 DESC  : 0 001000 000338
  LINE 4 DESC  : 0 000000 200000
  LINE 5 DESC  : 0 000000 200000
  LINE 6 DESC  : 0 000000 200000
  LINE 7 DESC  : 0 000000 200000
  LINE 8 DESC  : 0 000000 200000
  LINE 9 DESC  : 0 000000 200000
  LINE 10 DESC : 0 000000 200000
  LINE 11 DESC : 0 000000 200000
  LINE 12 DESC : 0 000000 200000
```

Figure 4–19. NSP Command Output

OLAYINFO

The OLAYINFO command analyzes overlay file allocation. This command is used when overlay file corruption is suspected.

<olayinfo>

```
— OLAYINFO ─┬───────────────────────┬──────────────────────┤
            │  ┌←──────────────┐    │
            └──┴─┬─ OVERLAY ─┬─┴────┘
                 ├─ FILE ────┤
                 └─ AREA ────┘
```

The following text explains the meaning of each construct:

OLAYINFO	DUMPANALYZER finds all ASD table entries that are in overlay disk, or are in the process of reading or writing from the overlay disk. DUMPANALYZER also checks for overlapping and compares the bit vectors.
OLAYINFO OVERLAY	If OVERLAY is specified, all data of the whole overlay structure is displayed. This includes the size of the areas, number of overlayable segments, number of pack names used, number of areas in use, number of files, and so forth.
OLAYINFO FILE	If FILE is specified, all data for each of the files is displayed separately. This includes the number of sectors per record, records per area, allocated records, in-use records, and so forth.
OLAYINFO AREA	If AREA is specified, all data for each area or disk row is displayed. This includes the unit number base address, number of records in use, and so forth.

Example

The following is an example of the output from the OLAYINFO command:

```
INPUT: OLAYINFO
CURRENT_OVERLAY             Ø ØØØØØØ ØØØØØ7  ( 7 )
OVERLAY_MANAGER (7)         5 8ØØØØ1 9Ø3E15
. FOR 2 SECTORS THERE ARE 1486 SEGMENTS, OVERLAY ALLOCATED 1488
. FOR 5 SECTORS THERE ARE 22 SEGMENTS, OVERLAY ALLOCATED 23
. SUM OF ALL SEGMENTS WITH PROBLEMS 2Ø95, SUM OF ALLOCATED 2Ø98
OTHER OVERLAY_MANAGER (6)   5 ØØØØØ1 9ØØØØ1
```

OPT

The OPT command lists the compile-time and run-time option settings and module alternatives selected when the dump was created.

<opt>

```
— OPT ──────────────────────────────────────────────────────┤
  ‾
```

Example

Figure 4–20 shows an example of the response to the OPT command.

```
INPUT: OPT
OPTIONS: SET: TERMINATE, LPBDONLY, AUTORM, AUTORECOVERY, AUTODC, CPBDONLY, CRUNCH, NOFETCH, RESOURCECHECK, NOSUMMARY,
              LOGPOSITIONING, SERIALNUMBER, MIRRORING, NETRECOVERY, LOGIOERRORSNEW
       RESET: OPEN, NOCHECK, DIAGNOSTICS, CDONLY, DUPSUPERVISOR, DUPINTRINSICS, TRANSWARNINGS, NODUMP, BACKUPBYJOBNR,
              PDTODISK, DIRDEBUG, CATALOGING, OKTIMEANDDATE, , ARCHIVING, , LOCKTRACE, IORANGECHECK, , KEYEDIOII, DIAGNOSTICDUMP,
              AUDIT, FILESATURATION, EOTSTATISTICS, PATHBALANCING, ISCDEBUG, IODIAGNOSTICS, PORTDEBUG, USECATDEFAULT,
              CATTEST, MCPTEST

COMPILETIME OPTIONS SET
     ASDDEBUG
     DIAGNOSTICS
     EXPERIMENTAL
     LINEINFO
     LOCKTRACE
     READLOCK
     READLOCKTIMEOUT
     RESTART
     TRACE
     CATALOGLEVEL = 0

MODULE ALTERNATIVES SELECTED: CPIOHDU SOFTPROCSWITCHING BLOCKEXIT_BMS PORTS_BNAV1  INITFILE_UIO UPIO CONFIGCONTROL70
```

Figure 4–20. OPT Command Output

PATHCNTRL (Selected EMS Systems)

The PATHCNTRL command provides an analysis of the PATHCNTRL table used on EMS systems. The PATHCNTRL table contains information on the access pathways to I/O peripherals. Each entry in the table consists of a PATHNODE node and one or more related PATHENTRY nodes. This command displays information for both types of nodes in an entry.

The PATHCNTRL command is valid only for dumps generated on EMS systems that use A Series I/O rather than pre-A Series I/O.

```
<pathcntrl>

— PATHCNTRL ─┬──────────────────────────────────┬──────────────────┤
             └─ UNIT ─┬───────┬─<decimal number>┘
                      └─ LOG ─┘
```

The following text describes the meaning of each construct:

PATHCNTRL	Every entry in the PATHCNTRL table is analyzed. All units referencing a given PATHNODE are displayed, and the PATHNODE and each associated PATHENTRY are analyzed.
PATHCNTRL UNIT <decimal number>	The PATHCNTRL table entry is analyzed for the specified peripheral unit. The <decimal number> corresponds to the physical unit number of the peripheral. Physical unit numbers are the numbers used to identify peripherals in various system commands such as PER (Peripheral Status) and RY (Ready). For example, the command *READY MT 29* readies the tape drive whose physical unit number is 29.
PATHCNTRL UNIT LOG <decimal number>	The PATHCNTRL table entry is analyzed for the specified peripheral unit. The <decimal number> corresponds to the logical unit number of the peripheral. Logical unit numbers are used internally by the MCP and appear only in memory dumps.

Example

Figure 4–21 shows an example of the first page of the response to the PATHCNTRL command.

```
INPUT: PATHCNTRL

ANALYZING ENTIRE PATHCNTRL TABLE @ 00019831; 160 WORDS IN TABLE

  ** THIS PATH NODE STARTS AT PATHCNTRL[1] **

          PK44 LOGICAL UNIT 6,
          PK45 LOGICAL UNIT 7,
          PK46 LOGICAL UNIT 8,
          PK47 LOGICAL UNIT 9,
          PK48 LOGICAL UNIT 10,
          PK49 LOGICAL UNIT 11,
          PK50 LOGICAL UNIT 12,
          PK51 LOGICAL UNIT 13

     THESE 8 UNITS ACCESSED VIA:

       PATHNODE: 10034010505A  VERSION=1, DLP TYPE=STORAGE MODULE DEVICE (PACK), PATH SIZE=5, NODE SIZE=5, FW DIGITS=10
 PATHENTRYMASK: 200000000002  PATHS ENTRY MASK=0002 => 1 PATH: 1
 PATHAVAILMASK: 300000000002  PATHS AVAILABLE MASK=0002 => NONE UNAVAILABLE
  PATHNODEINFO: 4002C0108000  BASE PHYS UNIT=44, 1 UQ PER UNIT, 8 UNITS ON THIS DLP
 PATHNODEQINDX: A00000030000  UNIT Q INDEX=0003

  ** PATH ENTRY 1, STARTS AT PATHCNTRL[6] **

PATHENTRYSTART: 501001100005  REL UNIT=0, DLP=1, IOP=1, IOP PORT=0, LEM=0, REL DLP=5, ONLINE, H Q PRESENT
 PATHENTRYINFO: 604000000000  CONTROLLING DLP=0,
PATHENTRYINFO2: 700000000001  CLEARSTATE=UNUSED, SNR=000, H Q INDEX=001, H Q DOPE VECTOR INDEX=00, H Q SLOT INDEX=1
   PATHENTRYFW: 801080104030  DLP FIRMWARE LEVEL IS: 0108010403
PATHENTRYQINDX: 900000000000  ** FOR NON A SERIES I/O ONLY **

  ** THIS PATH NODE STARTS AT PATHCNTRL[11] **

          SC128 LOGICAL UNIT 14,
          SC129 LOGICAL UNIT 15,
          SC130 LOGICAL UNIT 16,
          SC131 LOGICAL UNIT 17,
          SC132 LOGICAL UNIT 18,
          SC133 LOGICAL UNIT 19,
          SC134 LOGICAL UNIT 20,
          SC135 LOGICAL UNIT 21,
          SC136 LOGICAL UNIT 22,
          SC137 LOGICAL UNIT 23,
          SC138 LOGICAL UNIT 24,
          SC139 LOGICAL UNIT 25,
          SC140 LOGICAL UNIT 26,
          SC141 LOGICAL UNIT 27,
          SC142 LOGICAL UNIT 28,
          SC143 LOGICAL UNIT 29

     THESE 16 UNITS ACCESSED VIA:

       PATHNODE: 1000F0105050  VERSION=1, DLP TYPE=UIP OPERATOR DISPLAY, PATH SIZE=5, NODE SIZE=5, FW DIGITS=0
 PATHENTRYMASK: 200000000002  PATHS ENTRY MASK=0002 => 1 PATH: 1
 PATHAVAILMASK: 300000000002  PATHS AVAILABLE MASK=0002 => NONE UNAVAILABLE
  PATHNODEINFO: 400800210000  BASE PHYS UNIT=128, 2 UQ'S PER UNIT, 16 UNITS ON THIS DLP
 PATHNODEQINDX: A000000B0000  UNIT Q INDEX=000B
```

Figure 4–21. PATHCNTRL Command Output

PATTERN

The PATTERN command is used to set or examine the pattern register. The pattern register stores a distribution of 1s and 0s in a 48-bit word and its 4-bit tag. The pattern assigned by the PATTERN command is stored until changed by another PATTERN command.

The PATTERN command and the MASK command are used to provide information for use by the SEARCH command. The SEARCH command searches the dump for all words that match the pattern register, bit for bit. The MASK command must be used to mask off some of the bits in the pattern as not significant to the search. (Refer to the MASK and SEARCH commands for further information.)

```
<pattern>

— PATTERN ─┬──────────────────┬─────────────────────────────────────┤
           └─<simple value>───┘
```

The following text describes the meaning of each construct:

PATTERN	If no <simple value> follows PATTERN, the contents of the PATTERN register are displayed.
PATTERN <simple value>	The pattern register is loaded with <simple value>, which can then be used as a nonvolatile search pattern.

The pattern includes the 4-bit tag for a word. To specify a TAG, the concatenation form of simple value is used; input would specify 12 hexadecimal digits *& <number> TAG*. The tag value should be no larger than 15; however, if it is larger than 15, the tag value is placed in the tag using modulus 16. If no tag is specified, the tag is assumed to be 0.

Although the pattern represents 48 binary bits and a tag value of four binary bits, the pattern is displayed and is most often set in hexadecimal. Figure 4–22 shows a diagram of a pattern word, with its hexadecimal equivalent stated below it:

	bit#													
1	47	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	1	1	1	3
1	46	Ø	Ø	Ø	Ø	1	1	Ø	Ø	1	1	1	1	2
1	45	Ø	Ø	Ø	Ø	Ø	Ø	1	1	1	1	1	1	1
1	44	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	1	1	1	Ø
7		Ø	Ø	Ø	Ø	4	4	2	2	F	F	F	F	

Figure 4–22. Pattern Word Example

To put the pattern of bits shown in the word above into the pattern register, the corresponding pattern command would be PATTERN 4422FFFF & F TAG (it is understood that there are four leading zeros). The pattern register would display its contents as F 000044 22FFFF.

The default for the PATTERN register is displayed as 0 000000 000000.

Examples

```
INPUT:   PATTERN
PATTERN: 0 000000 000000

INPUT: PATTERN 23000EC05B72
PATTERN: 0 23000E C05B72

INPUT:   PATTERN 123456789ABC & DEC 10 TAG
PATTERN: A 123456 789ABC
```

For more information, see the discussion of the MASK and SEARCH commands in this section.

PC or PRINTCODE

The PC or PRINTCODE command translates code words into their mnemonic equivalents and prints the results.

```
<pc>

──┬─ PC ─────────┬─<simple address>─┬────────────────┬──────────────────────→
  └─ PRINTCODE ──┘                  └─ : ──<PWI>─────┘

→─┬──────────────────────────┬─┬──────────────────────┬───────────────────────┤
  └─ FOR ──<simple value>────┘ └─ LL ──<lex level>────┘
```

Translation of code begins on or before the designated <simple address>, which is the segment base. Translation of code ceases when both of the following conditions are true:

- The length specified is achieved.
- The instruction word is complete.

The <PWI> construct is a number that specifies a location relative to the segment base. The default value for <PWI> is 0.

If no *FOR <simple value>* construct is specified, the default length is three words.

The <lex level>, a number, indicates the running environment level. If it is not specified, a lex level of 3 is assumed. If the lex level of the running environment is not 3, the lex level must be specified to ensure proper interpretation of operators that contain address couples, such as NAMC and VALC.

Non-tag-three words are translated, but DUMPANALYZER displays a warning indicating that the word being analyzed is probably not code.

All display headings (such as, in the example, the lines from CODE ANALYSIS: to the underline) are suppressed from remote output but appear in the printer output.

Example

```
INPUT: PC 14325:124 FOR 3

CODE ANALYSIS:
              SEGMENT BASE: 14325
              WORD OFFSET : ØØ124
              LENGTH      : ØØØØ3
              LEX LEVEL   : 3

PWI:PSI MNEMONICS          HEX CODE
-------------------------------------------------
Ø125:Ø  RSNR               9581
Ø125:2  INSR 46:15         9C2EØF
Ø125:5  NAMC ØØ,Ø1BB       41BB  MEMLOCK

Ø126:1  RDLK               95BA
Ø126:3  DUPL               B7
Ø126:4  BRFL ØDØ8:5        AØADØ8
```

For more information, see "Simple Address" and "Multiple Address" in this section.

PIB

The PIB command prints the contents of a process information block (PIB), which is a task variable associated with a process stack. If a word within the PIB is specified, only that word is printed.

<pib>

```
— PIB ─┬─<number>──────────────┬─┬──────────────────┬──────────────────┤
       ├─ AT ─<simple address>─┤ ├─<PIB cell name>──┤
       └─ VIA ─<ASD number>────┘ └─ # ─<offset>─────┘
```

The following text describes the meaning of each construct:

PIB <number>	The PIB associated with the stack specified by <number> is printed.
PIB AT <simple address>	The PIB located at memory address <simple address> is printed.
PIB VIA <ASD number>	The PIB located at <ASD number> is printed.

When <PIB cell name> or # <*offset*> is used, only the specified word within the PIB is printed. For more information about tasks, refer to the TAB command.

Example

Figure 4–23 shows an example of the first page of the response to the PIB command.

```
INPUT: PIB 34

SPIBVECTOR[034] = 5 800005 803273

00      3 012000 844000     PIBMSCW
01      0 000000 000000     CODEHEADERS
02      0 000000 000012     CODELINKS          SEG DICT = 012
03      0 000000 000000     COMPILERINFO
04      0 400000 004E18     COREINTEGRAL       = -19992
05      0 000000 0001DC     COREINUSE          = 476
06      0 000000 0001DC     COREINUSESAVED     = 476
07      0 000000 0001DC     MAXSAVEMEMORYUSED  = 476
08-0C   0 000000 000000     ASDSINUSE          MAXASDSINUSE      MAXSTACKCHECK      GRAPHHEADWORD      OLAYCNTL
0D-11   0 000000 000000     OLAYSACTIVE        SEARCHINFO        SUBSPACEID         D1TIMESTARTED      RUNNINGCOUNT
12      7 000079 E89008     WS_PALACE          PCW:  LL=2, D[0] SEGMENT @ 1008:079E:0, CNTRL STATE (05374780) WS_JUDGE_JACKET
13      1 000327 3C0012     WS-PALACEREF       SIRW: OFFSET=0012 (0000+0012) SEGMENT = 03273
14-15   0 000000 000000     DBSINFO            YOURNAME
16      0 000340 000000     BDINFO
17      0 000000 000000     EVENTCAR
18      0 800000 000000     BEDWORD
19-1D   0 000000 000000     EVENTCOUNT         BLOCK             ALARMTIMES         ASSUMEIOTIME       ASSUMEPROCESSTIME
1E      0 4E3845 797D8C     BOTTIMESTAMP       = 01/24/90 12:25:59
1F-20   0 000000 000000     CPINFO             DCKEYPIBINFO
21      1 012000 84026C     ENTRYPOINT         SIRW: OFFSET=0274 (0008+026C) STACK = 012   (PCW) READYQINSERT
22      0 000000 000000     RESTARTRCW
23      B C32000 000040     EXCEPTIONEVENT     EVENT 00040 NOT HAPPENED
24-26   0 000000 000000     (EXCEPTIONEVENT2)  EXTERNALFAMILYLIN FREEPBITS
27      0 000000 000063     HEADERACCESS
28-2B   0 000000 000000     INSTRETRYINFO      INTERCEPTEDRCW    INTERCESSION       JOBDECKINFO
2C      0 034000 000000     JOBINFO
2D-31   0 000000 000000     JOBMSGS            LIBRARYSTATUS     PIBLOCKS47         PORTLIB PIB        PROCESSFAMILYLINK
32-36   0 000000 000000     REMPRINTMASK       REMPUNCHMASK      REPLY              REPLYEVENTX        (REPLYEVENT2)
37-39   0 000000 000000     ROLLOUTINFO        SOFTINTQ          TASKINFO
3A      5 800003 A03276     PIBIOCB
3B      0 000000 000002     TASKTYPE
3C-40   0 000000 000000     WAITSTART          DRCPIBINFO        PRTABLE            ACTIONBITS         ACTIONQHEAD
41      0 000000 000000     RECENTPROC
42      7 273100 189021     PALACE             PCW:  LL=2, D[0] SEGMENT @ 1021:1001:0, CNTRL STATE (14650720) UPDATESTACKPROCS
43      1 000327 3C0042     PALACEREF          SIRW: OFFSET=0042 (0000+0042) SEGMENT = 03273
44      7 2730F8 B89021     HOVEL              PCW:  LL=2, D[0] SEGMENT @ 1021:0F8B:0, CNTRL STATE (14650720) UPDATESTACKPROCS
45      1 000327 3C0044     HOVELREF           SIRW: OFFSET=0044 (0000+0044) SEGMENT = 03273
46      0 000000 002000     USAGE              = NOTINHERITMCSSTATUS
47      0 000000 000000     DMSBED
48      0 000000 000005     READYON            @ 0:00:00 AFTER HALTLOAD = 202.308679 SECONDS PRIOR TO DUMP
49      0 000000 000000     PROCESSOR
4A      0 400000 564C7A     CLOCKONTIME        @ 0:00:13 AFTER HALTLOAD = 188.735074 SECONDS PRIOR TO DUMP
4B-4C   0 000000 000000     LETGEORGEDOIT      READYTIME
4D      0 400000 000455     PROCESSTIME        = -0.0026616 SECONDS
4E-52   0 000000 000000     IOTIME             VOLCOUNT          OFFSPRINGACCOUNTS FILEACCOUNTS       VALIDITYBITS
53      0 007FFF FFFFFF     MAXPROCESSTIME     = 549755813887
54      0 007FFF FFFFFF     MAXIOTIME          = 549755813887
55      0 000000 000000     MISCINFO
56      B C32000 000041     PIBEVENT           EVENT 00041 NOT HAPPENED
57-58   0 000000 000000     (PIBEVENT2)        MLINFO
59      5 800004 F03274     TASKING TAB
- - - TAB

00      0 400000 564DFA     TIMESTARTED        @ 0:00:13 AFTER HALTLOAD = 188.734152 SECONDS PRIOR TO DUMP
01-02   0 000000 000000     DISKINTPERM        DISKINTTEMP
03      0 000000 000003     INITPBITKOUNT      = 3
04      0 000000 000455     INITPBITTYME       =  0.0026616 SECONDS
05-09   0 000000 000000     IOCOUNT1           IOCOUNT2          HISTORY            IOTIMETAB          MYSTACKHISTORY
0A-0E   0 000000 000000     OTHERPBITCOUNT     OTHERPBITTIME     PROCESSTIME        STOPPOINT          CARDSPUNCHED
0F-13   0 000000 000000     DISKUSED           HSINFO            ITINERARY          LINESPRINTED       SURROGATEINFO
14-18   0 000000 000000     TASKPARAMS         TEMPFILEMBYTES    WFLJOBINFO         ACCESSCODE         BACKUPFAMILY
```

Figure 4–23. PIB Command Output

PORT

The PORT command prints an analysis of a port file. A system response of PORT OVERLAYED implies that the port file was not in memory when the dump was created.

```
<port>

— PORT ─┬─<port index>─────────┬──────────────────────────────────┤
        ├─ AT ──<address>──────┤
        └─ VIA ──<ASD number>──┘
```

The following text describes the meaning of each construct:

PORT <port index>

The <port index> value is an index into the PSS array, which is a two dimensional, global data structure maintained by the Lists Module of the MCP. The user can determine the location of the PSS array by using the command *MD G PSS*.

The *PORT <port index>* command actually selects PSS [<index>,*] structures, only some of which are ports. DUMPANALYZER verifies that the <port index> points to a valid PORT structure by checking word 0, field [47:32] of the element selected (PSS [<index>,0].[47:32]). If this field contains "PORT" and field [15:16] of the same word is equal to <port index>, then the element selected is a valid PORT and is analyzed. If not, a "BAD PAB" message is displayed.

The <port index> is stored in the LIBRARYINFO word of a FIB using the PORT.

PORT AT <address>

The port at the given absolute memory address is analyzed. If the memory address specified does not contain a port data structure, then DUMPANALYZER displays the message "BAD PABNO".

PORT VIA <ASD number>

The port at the address referenced by the supplied ASD number is analyzed. If the address referenced by the supplied ASD number does not contain a port data structure, then DUMPANALYZER displays the message "BAD PABNO".

Example

The following is an example of the first page of the response to the PORT command. The dump was created on a system running BNA Version 2.

```
INPUT: PORT 54

PORT ATTRIBUTE BLOCK DESCRIPTOR AT 5 800001 D0265C
00(00) 0 D7D6D9 E30054 MYPORTADDRESS          =84(54)
01(01) 0 003200 000000 MAXSUBFILES            =50(32)
                       LASTSUBPORT            =0(0)
                       SERVICE                =0(0)
02(02) 0 23D004 000000 MAXRECSIZE             =9168(23D0)
                       PREFERREDCHARSET       =4(4)
                       REMOTERRF              =0(0)
                       TRANSLATEATT           =0(0)
                       DEF_DIALOG_PR_ATT      =0(0)
```

```
03(03) 0 000000 000109 FIRSTCHANGEDSUBFILE     =0(0)
                       HOSTSERVICESDIALOGNO    =0(0)
                       OLDYOURUSERCODE         =TRUE
                       WAITFOROPEN             =FALSE
                       SOMEERROR               =FALSE
                       ATTACHED_TO_PLM         =FALSE
                       BNAV2_PLM_PORT          =FALSE
                       TTP_PORT                =1(1)
                       SYNC_PORT               =0(0)
                       SERVICE_ATTRIBUTE       =FALSE
04(04) 0 000000 3205A0 OPENEDSUBPORTS          =0(0)
                       CLOSEDSUBPORTS          =50(32)
                       PORTOWNERSTACK          =90(5A)
                       PROGRAM AGENT           =FALSE
                       NETEX PORT              =FALSE
                       RETIRING                =FALSE
                       IOWNYOURLOCK            =FALSE
05(05) 0 000500 000180 PORT_EVENT_NUM          =5(5)
                       READABLESUBPORTS        =0(0)
                       USING PROCESS TYPE      =1(1)
                       WORD_ORIENTED           =TRUE
06(06) 0 010300 000000 TTP_IX                  =1(1)
                       SECURITYTYPEATT         =3(3)
                       FRAMESIZEATT            =0(0)
                       INTERFACE MODEL         =0(0)
07(07) 0 000001 000002 PORTSTATEEVENT          =1(1)
                       5 670000 0002C6
                       0 000000 000000  EVENT ANALYSIS:
                       MESSAGEEVENT            =2(2)
                       0 000000 000000
                       0 000000 000000  EVENT ANALYSIS:
08(08) 0 000000 000003 INTERNALEVENT           =0(0)
                       5 610000 00055B
                       3 000000 00008D  EVENT ANALYSIS:
                       PORTLOCK                =3(3)
                       0 000000 000000
                       0 000000 000000  EVENT ANALYSIS:
09(09) 0 000000 000000 PORTLOCKWORD            =0(0)
10(0A) 0 000000 000000 PORTGRABWORD            =0(0)
11(0B) 0 001900 1A001B MESSAGE FORWARDING LISTS
       LIST NUMBER: 27. NO OF ELEMENTS : 0.
```

PRINTARRAY

The PRINTARRAY command prints an entire array or part of an array, depending on the setting of the ARRAYLIMIT command. To print a present array, use either version of the PRINTARRAY command. To print an overlayed array from a program dump file, use the VIA attribute of the command.

```
<printarray>

— PRINTARRAY ─┬─────┬─<simple address>─┬──────────────────────────────┤
              ├─ AT ─┘                  │
              └─ VIA ─<ASD index>───────┘
```

The following text describes the meaning of each construct:

PRINTARRAY <simple address>	
PRINTARRAY AT <simple address>	Either of these formats will print all or part of an array at the given memory location.
PRINTARRAY VIA <ASD index>	This format prints all or part of an array at the stack location indicated by <ASD index>.

An array can be printed by three different routes:

- The ARRAY attribute of the MODE command can be set to TRUE; arrays are printed in the normal course of printing a stack.
- The PRINTVALUE command can be used to print present arrays.
- The PRINTARRAY command can be used to print both present and overlayed arrays.

Note: *Overlayed arrays are available only when a program dump to a disk file is being analyzed. The arrays are not dumped during memory dumps.*

For more information, see the discussion of the following commands in this section:

- ARRAYLIMIT
- MODE
- PV

PRINTER

The PRINTER command routes output to the line printer instead of to the terminal or a disk file. The standard DUMPANALYZER header page is printed as a preamble. Each command that causes the information to be dumped is also printed.

```
<printer>

— PRINTER ─────────────────────────────────────────────────────────┤
```

Printer output from DUMPANALYZER includes both upper case and lower case letters. If DUMPANALYZER output is directed to a printer that does not support lower case letters, then a transform function such as the UPPERCASE standard transform function should be used to translate lower case letters to upper case. For information about print transforms, refer to the *A Series Print System (PrintS/ReprintS) Administration, Operations, and Programming Guide.*

PRINTVAL

This command is a synonym for the PV (or PRINTVAL) command. Refer to the PV command.

PROC (EMS Systems)

Note *The PROC command has been deimplemented. This command had meaning only on systems with ASN memory, which is not supported on the Mark 3.9 system software release and later releases.*

PROCS

The PROCS command causes all processors that were running on the system (both E-mode and IOP) to be displayed. This command is valid only for dumps executed on Micro A, A 1, A 2, A 3, A 4, A 5, A 6, and A 10 systems.

```
<procs>
— PROCS ————————————————————————————————————————————|
```

Example

```
INPUT: PROCS
PROCESSORS CURRENTLY ON THE SYSTEM ARE:
EMP  #9  IN STACK ØA6  S=>ØØØ8E7B5
```

PROCSTACKS

The PROCSTACKS command displays the contents of each stack that has a processor currently on it. The stack that requested the memory dump is also displayed. The contents of the stacks are formatted and interpreted before they are displayed.

The STACK ACTIVE form of the STACK command provides more flexibility than the PROCSTACKS command.

```
<procstacks>
— PROCSTACKS ———————————————————————————————————————|
```

For more information, see the discussion of the STACK command in this section.

PROGRAMDUMP

The PROGRAMDUMP command prints the contents of stacks that have been dumped. The function of this command is to produce a listing similar to that of the PROGRAMDUMP command in ALGOL and other language compilers, and therefore is intended to be used with program dumps that were directed to disk. The content of the output is determined by the options specified at dump time and by the parameters of the PROGRAMDUMP command.

<programdump>

```
                    ┌←───────────────────────┐
— PROGRAMDUMP ──┴─┬─ ALL ──────────────────┤──────────────────────────┤
                    ├─ ARRAY ────────────────┤
                    ├─ ARRAYS ───────────────┤
                    ├─ BASE ─────────────────┤
                    ├─ CODE ─────────────────┤
                    ├─ DBS ──────────────────┤
                    ├─ FILE ─────────────────┤
                    ├─ FILES ────────────────┤
                    ├─ LIBRARIES ────────────┤
                    └─ PRIVATELIBRARIES ─────┘
```

The following text describes the meaning of each option:

PROGRAMDUMP	The stack for the dumping task is printed.
PROGRAMDUMP ALL	This option is equivalent to specifying all available options for the PROGRAMDUMP command.
PROGRAMDUMP ARRAY	
PROGRAMDUMP ARRAYS	All arrays in selected stacks are printed.
PROGRAMDUMP BASE	The base of the stack and the process information block (PIB) for all selected stacks are analyzed and printed.
PROGRAMDUMP CODE	The segment dictionary stack for the dumping task is printed.
PROGRAMDUMP DBS	All database stacks in the dump file are printed.
PROGRAMDUMP FILE	
PROGRAMDUMP FILES	All file information blocks (FIBs) in the selected stacks are analyzed and printed. The buffers associated with each file are also printed. Note that this option does not print disk file headers. The HDR command can be used to analyze disk file headers.
PROGRAMDUMP LIBRARIES	All library stacks in the dump file are printed.
PROGRAMDUMP PRIVATELIBRARIES	All private library stacks in the dump file are printed.

PV

The PV or PRINTVAL command displays the specified <simple value> or the specified word in the current stackwindow in several possible forms.

<pv>

```
                                            ┌←─────────────────────┐
──┬─ PV ────────┬─┬─<simple value>──────────┬┴┬─<mode option>──────┬┴─────────┤
  └─ PRINTVAL ──┘ └─ SW ─┬──────────────────┤ ├─ TIME6 ────────────┤
                         └─ # ─<offset>─────┘ ├─ TIME11 ───────────┤
                                              ├─ TIME14 ───────────┤
                                              ├─ MODE ─────────────┤
                                              ├─ RCW ──────────────┤
                                              ├─ EBC ─┬────────────┤
                                              │       └─ ARRAY ────┤
                                              └─ DESC ─────────────┘
```

The following text describes the meaning of each construct:

PV <simple value>	The specified <simple value> is displayed in hexadecimal form.
PV SW	
PV SW # <offset>	This syntax is valid only if the STACKWINDOW command has previously been used. The word that the current stackwindow centers around or the word in the current stackwindow with the specified offset is expanded.
<mode option>	The specified word is displayed in the desired <mode option>. (Refer to the MODE command.)
TIME6	In the TIME6 mode, the specified word is analyzed as if it were a TIME(6) format word. This word has the following format: 0 & (JULIANDATE-70000) [47:16] & (TIME(11) DIV 16) [31:32]
TIME11	In the TIME11 mode, the specified word is analyzed as if it were a TIME(11) format word. This word stores the time of day, expressed in multiples of 2.4 microseconds.
TIME14	In the TIME14 mode, the specified word is analyzed as if it were a TIME(14) format word. This word stores the time elapsed since the last halt/load in multiples of 2.4 microseconds. DUMPANALYZER appends the following message to the analysis: "SINCE LAST HALT/LOAD".
MODE	Displays the specified word in the modes that are currently set. (Refer to the MODE command.) In the FIB mode, <simple value> is expanded as a FIB. <simple value> must be a present descriptor. If the length field is incorrect, a warning message indicating that the value is probably not a FIB is displayed. In the IOCB mode, the IOCB word in the base of the stack and any descriptor can be analyzed as an IOCB.

continued

continued

On HDU systems, the PV command in the IOCB mode analyzes both IOCBs and HCBs. The word whose contents are <simple value> is a descriptor that might refer to either an IOCB or an HCB. If the descriptor refers to an IOCB and an HCB has been allocated for that IOCB then both structures are analyzed. Word three of the structure pointed to by the descriptor is checked to determine if the descriptor is for an HCB or an IOCB. If the tag of the word is 5 (indicating that it is a data descriptor), then the word is assumed to be the HCB Self Pointer and the descriptor is assumed to point to an HCB; otherwise, the descriptor is assumed to point to an IOCB.

If the MCP option READLOCK is set, each hard lock is shown with sequence number and procedure name in the LOCK mode.

In the construct "PV M[a] . . .", if M[a] and M[a + 1] are both Tag-2 words, they are analyzed together as a double operand. This construct is effective in DEC, OCT, BCL, EBC, or LOCK modes.

RCW	In the RCW mode, the specified word is analyzed as if it were an RCW.
EBC ARRAY	In the EBC ARRAY mode, an array is displayed in both hexadecimal and EBCDIC formats.
DESC	In the DESC mode, the specified word is analyzed as if it were a data descriptor.

For more information, see the discussion of the MODE and STACKWINDOW commands in this section.

Example

The following are examples of the output from the RCW option of the PV command:

```
INPUT: PV ØØØ81DF85Ø1E RCW
Ø ØØØ81D F85Ø1E RCW @ 1Ø1E:Ø1DF:4 (126324ØØ) MULTIWAIT

INPUT: PV 8ØØØØØ2Ø323D 7 5 TAG EBC ARRAY
5 8ØØØØØ 2Ø323D
        Ø(ØØØØ) Ø C1C2C3 C4C5C6 Ø C7C8C9 D1D2D3 ABCDEF GHIJKL
```

QUEUE

The QUEUE command displays DCALGOL queues.

<queue>

```
— QUEUE ─┬────────────┬─┬──────────────┬──────────────────────────┤
         ├─<number>───┤ └─ MSG ─┬──────┤
         └─ BUSY ─────┘         └─ ALL ┘
```

The following text describes the meaning of each construct:

QUEUE	If no <number> is specified, displays all DCALGOL queues.
QUEUE <number>	Displays the indicated queue.
BUSY	Displays all queues that have messages.
MSG	Displays all data comm messages in the selected queues.
ALL	Displays all messages in the selected queues, including any messages that are not data comm messages.

Example

Figure 4–24 shows an example of the response to the QUEUE command. The following are explanations of some items in the example:

- Each entry in the QUEUE NUMBER column identifies the start of a queue stack, which is a nonrunning stack of descriptors referencing the actual queue.
- If the QMSGINFO entry in a queue stack ends with a series of asterisks (**********), then there are messages to be removed from the queue. The MSGCOUNT item in the entry indicates the number of messages to be removed.
- The QTIBDESC entry is the queue task information block.

```
INPUT: QUEUE

DUMP OF DCALGOL QUEUE STACK(FFE) : 5 800002 100FFE
QUEUE
NUMBER
------

000D     5 800000 D03694   (QUEUE ADDR = 0002C05F)
                             0000  0 0008B5 80116C   QLINKAGE  (QTAIL=08B58, QHEAD=0116C)
                             0001  0 000370 0D0001   QINFO  (ACTIVATING SNR=037, USERS=1)
                             0002  0 000009 00000C   QSIZE  (TOTAL SIZE=9, MEMORY SIZE=12)
                             0003  0 000000 031000   QMSGINFO  (MSGCOUNT=3, MEMORYLIMIT=4096) **********
                             0004  0 000000 050BB8   QTANKINFO  (ROWSIZE=5, BLOCKSIZE=3000)
                             0005  0 000000 000000   QBUFFDESC
                             0006  B C32000 0000B9   QLOCKEVENT
                             0007  0 000000 000000
                             0008  B C32000 0000BA   QINSERTEVENT
                             0009  0 000000 000000
                             000A  0 000000 000000   QTIBDESC
                             000B  0 000000 000000   QLOCKCONTEND
                             000C  0 000000 000000   QLOCKOWNER
000C     5 800000 D035CE   (QUEUE ADDR = 00025EAC)
                             0000  0 000000 000000   QLINKAGE  (QTAIL=00000, QHEAD=00000)
                             0001  0 000370 0C0001   QINFO  (ACTIVATING SNR=037, USERS=1)
                             0002  0 000000 000000   QSIZE  (TOTAL SIZE=0, MEMORY SIZE=0)
                             0003  0 000000 0003E8   QMSGINFO  (MSGCOUNT=0, MEMORYLIMIT=1000)
                             0004  0 000000 050BB8   QTANKINFO  (ROWSIZE=5, BLOCKSIZE=3000)
                             0005  0 000000 000000   QBUFFDESC
                             0006  B C32000 0000A5   QLOCKEVENT
                             0007  0 000000 000000
                             0008  B C32000 0000A6   QINSERTEVENT
                             0009  0 000000 000000
                             000A  0 000000 000000   QTIBDESC
                             000B  0 000000 000000   QLOCKCONTEND
                             000C  0 000000 000000   QLOCKOWNER
000B     5 800000 D035CA   (QUEUE ADDR = 000260F8)
                             0000  0 000000 000000   QLINKAGE  (QTAIL=00000, QHEAD=00000)
                             0001  0 000370 0B0001   QINFO  (ACTIVATING SNR=037, USERS=1)
                             0002  0 000000 000000   QSIZE  (TOTAL SIZE=0, MEMORY SIZE=0)
                             0003  0 000000 0003E8   QMSGINFO  (MSGCOUNT=0, MEMORYLIMIT=1000)
                             0004  0 000000 050BB8   QTANKINFO  (ROWSIZE=5, BLOCKSIZE=3000)
                             0005  0 000000 000000   QBUFFDESC
                             0006  B C32000 0000A3   QLOCKEVENT
                             0007  0 000000 000000
                             0008  B C32000 0000A4   QINSERTEVENT
                             0009  0 000000 000000
                             000A  0 000000 000000   QTIBDESC
                             000B  0 000000 000000   QLOCKCONTEND
                             000C  0 000000 000000   QLOCKOWNER
000A     5 800000 D03290   (QUEUE ADDR = 00027D4C)
                             0000  0 000000 000000   QLINKAGE  (QTAIL=00000, QHEAD=00000)
                             0001  0 000370 0A0001   QINFO  (ACTIVATING SNR=037, USERS=1)
                             0002  0 000000 000000   QSIZE  (TOTAL SIZE=0, MEMORY SIZE=0)
                             0003  0 000000 0003E8   QMSGINFO  (MSGCOUNT=0, MEMORYLIMIT=1000)
                             0004  0 000000 050BB8   QTANKINFO  (ROWSIZE=5, BLOCKSIZE=3000)
                             0005  0 000000 000000   QBUFFDESC
```

Figure 4–24. QUEUE Command Output

READYQ

The READYQ command prints the ready queue in priority order. The ready queue is a list of processes that are waiting for a processor.

```
— READYQ ——————————————————————————————————————————————————————|
```

RECESS

The RECESS command allows the user to exit from DUMPANALYZER without removing the "pseudorecovery" file created during DUMPANALYZER's initialization process. For information on the usage of the pseudorecovery file, refer to "DUMPANALYZER Files" in this section.

<recess>

```
— RECESS ——————————————————————————————————————————————————————|
```

RELEASE

The RELEASE command causes the current output file to be closed and saved.

<release>

```
— RELEASE ─┬──────────────┬─────────────────────────────────────|
           └─ DISKFILE ───┘
```

The following text describes the meaning of each option:

RELEASE	Closes the current printer file. Repeated use of this command allows the output from the DUMPANALYZER run to be split into several printer files, which print when the originating job or MCS session terminates.
RELEASE DISKFILE	Closes and saves the current disk output file. If output was previously routed to the disk file, the output mode is changed to either the remote terminal or the printer (through an ODT run) by default. An immediate DISKFILE "<file title>" command can be used to route the subsequent output to the newly assigned disk output file. If there is a disk output file existing before normal termination of the run, DUMPANALYZER automatically closes and saves the disk file.

For more information, see the discussion of the following commands in this section:

- DISKFILE
- RELX
- REPEAT
- SHOW

RELX

The RELX command causes the current line printer file to be closed and printed while DUMPANALYZER is still running. The RELX command differs from the RELEASE command in that RELX causes immediate printing.

```
<relx>

— RELX ———————————————————————————————————————————————————————————————————|
```

The printout is preceded by a heading page containing the usercode and mix number of the DUMPANALYZER task, and the job name DPAOUTPUT.

REMOTE

The REMOTE command routes output to the terminal rather than to a line printer or diskfile. REMOTE is not valid when DUMPANALYZER is run from the ODT.

```
<remote>

— REMOTE —————————————————————————————————————————————————————————————————|
    ——
```

REPEAT

The REPEAT command causes the previous interactive command to be repeated. The output from the command is routed to either the printer or the remote terminal.

```
<repeat>

— REPEAT ─┬──────┬─┬─────────────┬────────────────────────────────────────|
  ———     └─ TO ─┘ ├─ PRINTER ───┤
                   └─ DISKFILE ──┘
                      —
```

The following text describes the meaning of each option:

REPEAT	The previous command is repeated. Output is repeated to whatever peripheral device has been initially specified.
REPEAT PRINTER	
REPEAT TO PRINTER	The previous command is repeated and the output is directed to the line printer. If REMOTE operation is in effect, each REPEAT to the printer is in a separate printer backup file. This version of the REPEAT command can be used to obtain a hard copy of command output during an interactive session at a remote terminal.
REPEAT DISKFILE	
REPEAT TO DISKFILE	The previous command is repeated and the output is directed to the user-assigned disk output file. See the DISKFILE and RELEASE commands for information on assigning and saving the disk output file.

For more information, see the discussion of the following commands in this section:

- DISKFILE
- RELEASE
- SHOW

RESULTQ (EMS Systems)

The RESULTQ command invokes output of the result queues generated by I/O operations. These result queues display the linking together of IOCBs after the I/O operations are complete. This command is only valid for dumps from EMS systems.

```
<resultq>
— RESULTQ ——————————————————————————————————————————|
```

Example

Figure 4–25 shows example output from the RESULTQ command.

```
INPUT: RESULTQ

********** 5 800000 90197E -- RESULT QUEUES
00  0 10CF00 000000  (RESULTQ MARK)
01  0 000000 000000  (EMPTY QUEUE)
02  0 000000 000000  (EMPTY QUEUE)
03  5 C00001 E01C30  (MOST RECENT IOCB FINISHED)
.... LINKED IOCB @ 00028F4F

**** WARNING: THIS IOCB IS IN PROCESS. ****
**** THE ANALYSIS MAY BE INCONSISTENT. ****
                  IOP CNTRL: 0 10CB00 028409 TEST, CHAR, FORWARD, PROTECT, IMMEDIATE, DON'T COUNT, CAUSE IO FINISH,
                                               QUEUE AT HEAD
                DLP ADDRESS: 0 000001 100010 IOP ID=1, IOP PORT=0, LEM PORT=1, REL DLP=0
                 UNIT QUEUE: 5 C00001 6019A0
                   SELF PTR: 5 C00001 E01C30
                COMMAND PTR: 5 E00001 901C30
                 RESULT PTR: 5 E00001 B01C30
                C/R LENGTHS: 0 000000 0A000C
                RESULT MASK: 0 000000 000000
                 RSLT QUEUE: 5 E00000 30197E
                  NEXT LINK: 5 C00001 E026CF
                   DATA PTR: 0 000000 000000
                CURR LENGTH: 0 000000 000000
                   IOP RSLT: 0 000000 000025 EXCEPTION, DLP ERROR, COMPLETED AFTER Q SUSPENDED
                 START TIME: 0 000553 D64966 15:15:15 (00:37:49.82936 PRIOR TO DUMP)
                FINISH TIME: 0 00058C 5759D1 15:53:10 (00:00:05.32850 AFTER DUMP)
                 INFO1 WORD: 0 01202D 78000C UNITNUMBER = 50, OWNER STKNO = 012, INITIATING STKNO = 02D, REQUESTOR = 30 (KERNELIO)
                 INFO2 WORD: 0 000813 C91302 STATE = 3 (ACTIVE), FIXED PATH, PATHNAME = 01, PROC NO = 9, IOP NO = 1, IOTYPE = 2, FATE
                   LOG DESC: 0 000000 000000
                       IOCW: 0 000000 000000 IOSTANDARDFIELD: WRITE, 6-BIT
                BUFFER DESC: 0 000000 000000
                  EVENT REF: 0 000000 000000
                   FIB DESC: 0 000000 000000
                    IO MASK: 0 000000 000000
                 LOGICAL RD: 0 000000 000000
                    IO INFO: 0 000000 000000
                   CMD/RSLT: 0 216000 000000
                   CMD/RSLT: 0 000000 000000
                   CMD/RSLT: 0 004100 000000
                   CMD/RSLT: 0 000000 000000
                 SAVED INFO: 0 000000 000000
.... LINKED IOCB @ 0008D7A5

**** WARNING: THIS IOCB IS IN PROCESS. ****
**** THE ANALYSIS MAY BE INCONSISTENT. ****
                  IOP CNTRL: 0 10CB00 028409 TEST, CHAR, FORWARD, PROTECT, IMMEDIATE, DON'T COUNT, CAUSE IO FINISH,
                                               QUEUE AT HEAD
                DLP ADDRESS: 0 000001 100010 IOP ID=1, IOP PORT=0, LEM PORT=1, REL DLP=0
                 UNIT QUEUE: 5 C00001 60199F
                   SELF PTR: 5 C00001 E026CF
                COMMAND PTR: 5 E00001 9026CF
                 RESULT PTR: 5 E00001 B026CF
                C/R LENGTHS: 0 000000 0A000C
                RESULT MASK: 0 000000 000000
                 RSLT QUEUE: 5 E00000 30197E
                  NEXT LINK: 5 C00001 E01C2E
                   DATA PTR: 0 000000 000000
                CURR LENGTH: 0 000000 000000
                   IOP RSLT: 0 000000 000025 EXCEPTION, DLP ERROR, COMPLETED AFTER Q SUSPENDED
                 START TIME: 0 000563 A3A51E 15:25:51 (00:27:13.54952 PRIOR TO DUMP)
                FINISH TIME: 0 00058C 5745D5 15:53:10 (00:00:05.31622 AFTER DUMP)
                 INFO1 WORD: 0 012014 78000B UNITNUMBER = 49, OWNER STKNO = 012, INITIATING STKNO = 014, REQUESTOR = 30 (KERNELIO)
                 INFO2 WORD: 0 000813 C91302 STATE = 3 (ACTIVE), FIXED PATH, PATHNAME = 01, PROC NO = 9, IOP NO = 1, IOTYPE = 2, FATE
                   LOG DESC: 0 000000 000000
```

Figure 4–25. RESULTQ Command Output

RJ

Note *The RJ command has been deimplemented. The RIGHTJUSTIFY option of the TERMINAL command can be used instead.*

SAVE

The SAVE command saves the contents of memory dump tapes and relevant information from the MCP code file in a disk file for later analysis. The file is saved under the <file title> given; <file title> must appear in quotes. LINEINFO must be set and the MCP global names must be available. The AREASIZE of the saved file is 10 records or 500 segments.

When the TAPEIN file has been file-equated to a DP file and the saved dump file is to reside on the same pack family, the DP file is updated with the MCP code file information and the title of the DP file is changed to the <file title> given. The original DP file will no longer exist if the SAVE command completes successfully. This occurs only if the DP file that is file-equated to TAPEIN has your usercode; otherwise, a new saved dump file with the title <file title> is created.

The file created by the SAVE command can be quite large (from 30,000 to 150,000 disk segments). It might be desirable to copy the file to tape rather than leaving it on disk for this reason.

The file created using the SAVE command is not removed when DUMPANALYZER is exited. It must be removed using CANDE commands or MARC screen functions.

```
<save>
— SAVE — " —<file title>— " ——————————————————————————————|
```

SEARCH

The SEARCH command, when used in conjunction with the MASK and PATTERN commands, provides the ability to check each word in a memory dump for a specified pattern of bits; this pattern can include all 48 bits within the word and its four tag bits, or it might search for all words that have a subset of those 48 bits in common. SEARCH can cause a search for a pattern that is specified by the PATTERN command and limited by the mask set in the MASK command. Patterns for a search can also be specified within the SEARCH command; these patterns do not use the mask and pattern registers.

The addresses of the words that match the specified pattern, along with the contents of those words, can be saved and later retrieved as desired. The saved words can be searched again with different search patterns.

<search>

```
                                  ┌←──────────────────────────────────────┐
— SEARCH ─┬──────────────┬────────┴┬─/1\- REFS ─<simple address>─────────┬┴──→
          ├─ ABSENTCOPY ─┤         ├─/1\- RANGE ─<multiple addresses>─┤
          ├─ ABSENTDESC ─┤         ├─/1\- MASK ─<simple value>────────┤
          ├─ MEMERRORS ──┤         └─/1\- PATTERN ─<simple value>─────┘
          ├─ MOMDESC ────┤
          └─ PREVIOUS ───┘

→─┬────────────────────┬──────────────────────────────────────────────────┤
  └─ : ─┬┬─ T ─┬┬──────┘
        │└─ K ─┘│
        └←──────┘

— ?AX ─┬─ WHERE ───────────────┬──────────────────────────────────────────┤
       ├─ STOP ────────────────┤
       ├─ - ─ TEXT ────────────┤
       ├─ + ─ TEXT ────────────┤
       ├─ - ─ KEEP ────────────┤
       └─ SKIP ────────────────┘
```

The following text describes the meaning of each construct:

SEARCH	If SEARCH is used without any modifying tokens, the contents of the pattern and mask registers are used in the search. Refer to the MASK and PATTERN commands for information about how these registers are set.
ABSENTCOPY	ABSENTCOPY causes a search for all absent copy descriptors. An absent copy descriptor references a data structure on disk.
ABSENTDESC	ABSENTDESC causes a search for all absent mom descriptors. An absent mom descriptor references a data structure on disk.
MEMERRORS	MEMERRORS causes a search for regions of memory that contain memory parity errors. The MASK and PATTERN options are ignored when MEMERRORS is used; however, the other SEARCH options are unaffected.
MOMDESC	MOMDESC causes a search for all present mom descriptors. A present mom descriptor references a data structure in core.
PREVIOUS	PREVIOUS causes a search from among a group selected from a previous SEARCH command and stored via the K option described below.
REFS <simple address>	REFS causes a search for present descriptors that reference the section of memory including the specified address.
RANGE <multiple addresses>	RANGE <multiple addresses> indicates a range of memory over which to search. <multiple addresses> indicates which set or sets of addresses constitute the range. The syntax for <multiple addresses> is provided in "Basic Constructs" in this section.
MASK <simple value>	MASK <simple value> can be used to specify a mask to be used for the duration of the search only, but not to be placed in the mask register. The contents of the mask is <simple value>, a hexadecimal representation of a 48-bit word with an optional tag, which places a 1 in each bit that is significant in the pattern, and a 0 to mask all insignificant bits.

continued

continued

PATTERN <simple value>	PATTERN can be used to specify a pattern for which to search; however, this pattern will not be placed in the pattern register. The contents of the pattern is <simple value>, a hexadecimal representation of a 48-bit word with an optional tag, whose significant bits are pointed to by the mask. The SEARCH command takes this pattern and finds words which match the pattern in the significant bits which have been selected by the mask.
:T	*:T* lists the words that match the current pattern modified by the current mask and the hexadecimal address for each word next to its contents.
:K	*:K* retains a list of all words that match the current pattern modified by the current mask, and the addresses for those words. This list can be SEARCHed again later using the PREVIOUS modifier. All of memory is searched unless RANGE or PREVIOUS is used.
?AX	The search can be controlled asynchronously through the ?AX (Accept) option. The search occurs from the high end of memory toward 0; thus, memory indices printed in response to ?AX WHERE decrease. The different combinations of tokens for the ?AX command follow.
?AX WHERE	The WHERE option asks where the search is.
?AX STOP	The STOP option stops the search and reprompts the user with a ":READY" message.
?AX –TEXT	
?AX +TEXT	The +TEXT and –TEXT options turn the listing of the text on and off, respectively.
?AX –KEEP	The –KEEP option suppresses the formation of the list of kept matches.
?AX SKIP	The SKIP option abandons the current range and skips to the next range (if any).

Examples

The following command searches stack 368 for all words that match the criteria previously set by the MASK and PATTERN commands:

```
SEARCH RANGE STK 368 BOSR TO LOSR
```

The following command finds all mom descriptors in memory and retains them in a list:

```
SEARCH MOMDESC RANGE Ø TO END :K
```

The following command searches the list of mom descriptors stored by the previous command, and finds those that refer to the region including address 381F6:

```
SEARCH REFS 381F6 PREV : T
```

For more information, see the discussion of the following commands in this section:

- MASK
- PATTERN

SHOW

The SHOW command shows the previous input.

```
— SHOW ─┬───────┬──────────────────────────────────────┤
        └─ ALL ─┘
```

The following text describes the meaning of each option:

SHOW

SHOW ALL	In addition to the previous input, the output mode (remote, printer, or the name of the disk output file) is also displayed.

For more information, see the discussion of the following commands in this section:

- DISKFILE
- REPEAT
- RELEASE

SIB

Note: *Previously, structure information blocks (SIBs) were treated as stacks. SIBs are now treated like any other array. Therefore, the SIB command is no longer needed and has been deimplemented.*

STACK

The STACK command displays the formatted and interpreted contents of a stack.

```
<stack>

— STACK —<first parameter>—<second parameter>————————————————————|

<first parameter>

——┬— ACTIVE ———┬——————————————————————————————————————————————————|
  ├— ALL ——————┤
  ├— DATABASE —┤
  ├— DUMPING ——┤
  │ ┌←—— , ——┐ │
  └—┴<number>┴—┘

<second parameter>

——┬————————————————————————————————————————————————————————┬——————|
  ├— HISTORY ——————————————————————————————————————┬———————┤
  ├— SUMMARY ———————————————————————————————┬——————┤ └— PIB —┘
  └— OFFSET —<simple value>—┬———————————————┤
                            └ - —<simple value>—┘
```

The first parameter of the STACK command is used to specify the stack or set of stacks to be displayed. The following text describes the elements of the first parameter:

ACTIVE	All stacks that are alive or active at the time of the dump, including the stack that is performing the dump, are displayed.
ALL	All stacks on the system are displayed.
DATABASE	All task stacks using Data Management System II (DMSII), including database stacks, and database task stacks are displayed.
DUMPING	The stack performing the dump is displayed. If a different stack initiated the dump, then the initiating stack is also displayed.
<number>	All the specified stacks are displayed.

The second parameter of the STACK command is used to restrict the display of the stack or stacks. If no restriction is present, the entire stack and its PIB are displayed. The following text describes the elements of the second parameter:

HISTORY	The MSCWs and RCWs in the stack or stacks are displayed. In HARDWAREINTERRUPT environments, the P1 and P2 parameters are also shown and analyzed.
SUMMARY	The summary information for the stack or stacks is displayed.
OFFSET <simple value>	The contents of the stack or stacks starting at the offset <simple value> and continuing to the base of the stack are displayed.
OFFSET <simple value> – <simple value>	The contents of the stack or stacks between the two specified offsets are displayed. Note that the first <simple value> must be greater than the second <simple value>.
PIB	The PIB of the stack is displayed.

Example 1

The following example lists stack 368 from 10B to F5.

```
STACK 368  OFFSET 10ØB - F5
```

Examples 2 through 4

A complete example of the STACK command output is presented in three parts in Figures 4–26, 4–27, and 4–28.

Process Stack (Figure 4–26)

The first part of the STACK command output, shown in Figure 4–26, includes the following items:

Item	Meaning
STACKDUMP FOR STACK <number>	Specifies the stack number.
MIX NUMBER	Lists the job number and mix number for the process.
NAME	Lists the name of the process, in this case ONESECONDBURDEN.
JOB MESSAGES	Lists RSVP, DISPLAY, or ACCEPT messages for the process.
STACK KIND	Lists the type of process, in this case "I.R." for "independent runner."
STATUS	Lists the current status of the process.
PROCESS TYPE	Lists the process type as recorded by the TYPE task attribute: CALL (synchronous task), PROCESS (asynchronous task), RUN (job written in a language other than WFL), or JOBSTACK (WFL job).
BOSR	The absolute memory address of the base of the stack area.
LOSR	The absolute memory address of the upper limit of the stack area.
LENGTH	The length of the stack area in hexadecimal and decimal notation.

Following these items is a listing of the contents of each word in the process stack, starting with the S register and working down, one word per line. Each entry is divided into the following columns, from left to right:

- A 4-digit hexadecimal number giving the offset of the word from the base of the stack.
- Address couples (such as 01, 000C) giving the lexical level and offset of items declared in each procedure.

- The contents of the word, expressed in hexadecimal. The leading digit expresses the tag bits. The remaining 48 bits are represented by two groups of 6 hexadecimal digits. Each hexadecimal digit represents a group of four bits, starting with [47:4] and working over to [3:4].
- Analysis of the contents of the word.

The following are some features of the analyses presented for various types of words:

- Operands are printed in octal, decimal, and EBCDIC according to the following rules:
  - Decimal operand values can be formatted in integer, fixed-point, or exponential notation, as appropriate.
  - EBCDIC operands are printed in EBCDIC only if all characters are graphics. However, if an operand contains right-justified EBCDIC characters with null fill, the EBCDIC characters are printed.
  - The MODE command, discussed elsewhere in this section, specifies modes controlling the expansion of various types of words.
- DESC (word data descriptor) analyses indicate whether data is present in, or absent from, main memory, whether the descriptor is a copy or the original descriptor, the type and length of the data item, and (if required) the address of the data item.
- DROP (double-precision operand) analyses indicate the value of each word in octal, scientific notation, and double-precision scientific notation.
- MSCW (mark stack control word) analyses are preceded by an asterisk (*) to make them easy to spot, and include a reference to the position of the previous MSCW.
- PCW (program control word) analyses include the lexical level of the procedure being entered, the segment number and relative address within that segment where procedure entry occurs, and the interrupt state.
- RCW (return control word) analysis includes the lexical level, state (control state or normal state), segment number, relative address within the segment where procedure entry occurred, and the sequence number of the record where the procedure is located in the source file. The RCW analysis is followed by the following entries:
  - SEG DESC (segment descriptor) analysis, which includes the name of the procedure referenced by the row.
  - CODE analysis, which includes right and left broken brackets (>   <) around the word where procedure entry occurred.
- SCW (software control word) analyses specify the type or dimension of arrays, the type of declaration (file, interrupt, or fault), or the presence of a nonlocal GO TO.
- SIRW (stuffed indirect reference word) analyses indicate the offset into the specified process stack.
- SIW (step index word) analyses indicate the current value, increment value, and final value of an iteration statement.

Process Information Block (Figure 4–27)

The second part of the STACK command output, shown in Figure 4–27, starts with a pointer to the SPIBVECTOR, which is parallel to the stack vector and stores descriptors for active process information blocks (PIBs). Then, for each word in the PIB, there is an entry separated into the following columns, from left to right:

- The hexadecimal offset of the word in the PIB.
- The hexadecimal contents of the word.
- Analysis of the contents of the word.

Task Attribute Block (Figure 4–28)

The third part of the STACK command output, shown in Figure 4–28, shows the contents of the task attribute block (TAB) for the process. The format of this output is similar to that used for the PIB printout. Note that arrows (=>) appear in front of active task attributes. Following the TAB listing is a dump of the contents of the environment TAB for the process.

```
INPUT: STACK 34
                                            STACKDUMP FOR STACK 034        MIX NUMBER 0/0
         NAME: ONESECONDBURDEN.

  JOB MESSAGES: (NULL)
   LOCK STATUS: (NONE FOUND)
    STACK KIND: I.R.
                                                              STACKINFO = 0 000803 580000
                                                            STACKSTATUS = 0 1E640F 000010
        STATUS: READY                                 NORMAL
  PROCESS TYPE: RUN

BOSR=0002204B  LOSR=000221DD  LENGTH=00192 (402)

0026               3 000641 285021  RCW:  LL=1, CNTRL STATE                [MCP SEGMENT      @ 1021:0412:3 (01064800/13504615/135
                                    SEG DESC:  3 A0016E 301100 ------ BLAST WAITING STACK
                                    CODE:  3 00D7B2 158CA0   3 241101 04A244   3 11B09C 2704B0  >3 B6959E 6002B8<  3 B23595 B84326
0025 (01,0004)     0 000000 065B9B
0024 (01,0003)     0 000000 000002
0023 (01,0002)     0 000000 000000
0022               3 01003C 085021  RCW:  LL=1, CNTRL STATE                [MCP SEGMENT      @ 1021:03C0:0 (04280770/13504251/135
                                    SEG DESC:  3 A0016E 301100 ------ READYQINSERT
                                    CODE:  3 C5E3C9 D4C500   3 9542AB ABAE40   3 B5B0B2 02B1AB  >3 B5A200 8CB5B5<  3 A3FFFF FFFFFF
0021 -----D[1]=>3 012000 84400D     *MSCW: PREVIOUS MSCW @ 0014; D[0]=0008 IN STACK 012

0020 (01,000C)     6 A00000 000000  SCW: (NO-GOPAST)
001F (01,000B)     0 000808 185021
001E (01,000A)     7 000007 189021  PCW
001D (01,0009)     4 025FF3 E23000  "ON <FAULT>" MARKER
001C (01,0008)     0 000000 000000
001B (01,0007)     0 000000 000000
001A (01,0006)     0 000000 00AF9E
0019 (01,0005)     0 000000 000006
0018 (01,0004)     0 000364 0C4825
0017 (01,0003)     0 000000 00AEE1
0016 (01,0002)     0 000005 055017
0015               3 00028F E890E4  RCW:  LL=2, CNTRL STATE                [MCP SEGMENT      @ 10E4:08FE:1 (17458000)]
                                    SEG DESC:  3 A00146 603277 ------ SCHEDULER
                                    CODE:  3 4A6870 C30C30   3 B0B096 2BABA3   3 FFAE40 7BABA3  >3 FFB234 B22695<  3 B88180 95B9A2
0014 -----D[1]=>3 012000 844002     *MSCW: PREVIOUS MSCW @ 0012; D[0]=0008 IN STACK 012

0013               3 000000 002000  RCW:  DUMMY (RUN)
0012 -----D[2]=>3 000327 3C8011     *MSCW: PREVIOUS MSCW @ 0001;

0011               5 270001 241117  THEFILE
0010               0 000000 000000  STRINGPOOLMOM
000F               0 000000 000000  STRINGPOOLMARK
000E               0 000000 000000  LOCKCOUNT
000D               0 000000 000000  JUNK
000C               0 000000 004608  BLOWBY
000B               0 000000 000000  WARNINGLIST
000A               5 000000 400000  SEGMNTARRAY
0009               0 000000 000000  CONDLISTDESC
0008               0 000000 000000  OATDESC
0007               0 000000 000000  AITDESC
0006               7 034000 000000  STACKINFOTOP
0005               0 000000 000000  OLAYINFODESC
0004               6 800000 000000  OLAYCW
0003               0 000000 000000  INTMAPLINKS
0002               0 000000 000000  BASEOFSTACKRCW
0001               3 034000 040000  BASEOFSTACKMSCW
0000               3 000002 600005  TOSCWD
```

Figure 4–26. STACK Command Output: Process Stack

```
SPIBVECTOR[034] = 5 800005 B03273

00      3 012000 844000     PIBMSCW
01      0 000000 000000     CODEHEADERS
02      0 000000 000012     CODELINKS           SEG DICT = 012
03      0 000000 000000     COMPILERINFO
04      0 400000 004E18     COREINTEGRAL        = -19992
05      0 000000 0001DC     COREINUSE           = 476
06      0 000000 0001DC     COREINUSESAVED      = 476
07      0 000000 0001DC     MAXSAVEMEMORYUSED   = 476
08-0C   0 000000 000000     ASDSINUSE           MAXASDSINUSE      MAXSTACKCHECK     GRAPHHEADWORD     OLAYCNTL
0D-11   0 000000 000000     OLAYSACTIVE         SEARCHINFO        SUBSPACEID        D1TIMESTARTED     RUNNINGCOUNT
12      7 000079 E8900B     WS_PALACE           PCW:  LL=2, D[0] SEGMENT @ 100B:079E:0, CNTRL STATE (05374780) WS_JUDGE_JACKET
13      1 000327 3C0012     WS_PALACEREF        SIRW: OFFSET=0012 (0000+0012) SEGMENT = 03273
14-15   0 000000 000000     DBSINFO             YOURNAME
16      0 000340 000000     BDINFO
17      0 000000 000000     EVENTCAR
18      0 800000 000000     BEDWORD
19-1D   0 000000 000000     EVENTCOUNT          BLOCK             ALARMTIMES        ASSUMEIOTIME      ASSUMEPROCESSTIME
1E      0 4E3845 797D8C     BOTTIMESTAMP        = 01/24/90 12:25:59
1F-20   0 000000 000000     CPINFO              DCKEYPIBINFO
21      1 012000 84026C     ENTRYPOINT          SIRW: OFFSET=0274 (0008+026C) STACK = 012   (PCW) READYQINSERT
22      0 000000 000000     RESTARTRCW
23      8 C32000 000040     EXCEPTIONEVENT      EVENT 00040 NOT HAPPENED
24-26   0 000000 000000     (EXCEPTIONEVENT2) EXTERNALFAMILYLIN FREEPBITS
27      0 000000 000063     HEADERACCESS
28-2B   0 000000 000000     INSTRETRYINFO       INTERCEPTEDRCW    INTERCESSION      JOBDECKINFO
2C      0 034000 000000     JOBINFO
2D-31   0 000000 000000     JOBMSGS             LIBRARYSTATUS     PIBLOCKS47        PORTLIB PIB       PROCESSFAMILYLINK
32-36   0 000000 000000     REMPRINTMASK        REMPUNCHMASK      REPLY             REPLYEVENTX       (REPLYEVENT2)
37-39   0 000000 000000     ROLLOUTINFO         SOFTINTQ          TASKINFO
3A      5 800003 A03276     PIBIOCB
3B      0 000000 000002     TASKTYPE
3C-40   0 000000 000000     WAITSTART           DRCPIBINFO        PRTABLE           ACTIONBITS        ACTIONQHEAD
41      0 000000 000000     RECENTPROC
42      7 273100 189021     PALACE              PCW:  LL=2, D[0] SEGMENT @ 1021:1001:0, CNTRL STATE (14650720) UPDATESTACKPROCS
43      1 000327 3C0042     PALACEREF           SIRW: OFFSET=0042 (0000+0042) SEGMENT = 03273
44      7 2730F8 B89021     HOVEL               PCW:  LL=2, D[0] SEGMENT @ 1021:0F8B:0, CNTRL STATE (14650720) UPDATESTACKPROCS
45      1 000327 3C0044     HOVELREF            SIRW: OFFSET=0044 (0000+0044) SEGMENT = 03273
46      0 000000 002000     USAGE               = NOTINHERITMCSSTATUS
47      0 000000 000000     DMSBED
48      0 000000 000005     READYON             @ 0:00:00 AFTER HALTLOAD = 202.308679 SECONDS PRIOR TO DUMP
49      0 000000 000000     PROCESSOR
4A      0 400000 564C7A     CLOCKONTIME         @ 0:00:13 AFTER HALTLOAD = 188.735074 SECONDS PRIOR TO DUMP
4B-4C   0 000000 000000     LETGEORGEDOIT       READYTIME
4D      0 400000 000455     PROCESSTIME         = -0.0026616 SECONDS
4E-52   0 000000 000000     IOTIME              VOLCOUNT          OFFSPRINGACCOUNTS FILEACCOUNTS      VALIDITYBITS
53      0 007FFF FFFFFF     MAXPROCESSTIME      = 549755813887
54      0 007FFF FFFFFF     MAXIOTIME           = 549755813887
55      0 000000 000000     MISCINFO
56      B C32000 000041     PIBEVENT            EVENT 00041 NOT HAPPENED
57-58   0 000000 000000     (PIBEVENT2)         MLINFO
59      5 800004 F03274     TASKING TAB
```

Figure 4–27. STACK Command Output: PIB

```
- - - TAB

00      0 400000 564DFA     TIMESTARTED         @ 0:00:13 AFTER HALTLOAD = 188.734152 SECONDS PRIOR TO DUMP
01-02 0 000000 000000     DISKINTPERM         DISKINTTEMP
03      0 000000 000003     INITPBITKOUNT       = 3
04      0 000000 000455     INITPBITTYME        =  0.0026616 SECONDS
05-09 0 000000 000000     IOCOUNT1            IOCOUNT2            HISTORY             IOTIMETAB           MYSTACKHISTORY
0A-0E 0 000000 000000     OTHERPBITCOUNT      OTHERPBITTIME       PROCESSTIME         STOPPOINT           CARDSPUNCHED
0F-13 0 000000 000000     DISKUSED            HSINFO              ITINERARY           LINESPRINTED        SURROGATEINFO
14-18 0 000000 000000     TASKPARAMS          TEMPFILEMBYTES      WFLJOBINFO          ACCESSCODE          BACKUPFAMILY
19-1D 0 000000 000000     CHARGECODE          DBEQTNBLK           DESTNAME            FAMILYSUB           HOSTNAME
1E-21 0 000000 000000     JSTITLE             MYBONAME            MYFPB               MYLPB
22      5 800000 403275     MYNAME              = ONESECONDBURDEN
23-27 0 000000 000000     MYPPB               MYPRINTDEFAULTS     SUPPRESSWARNING     SOURCENAME          SUBSYSTEMID
28-2B 0 000000 000000     TASKMCSNAME         TASKSTRING          LOCKEDEVENT         (LOCKEDEVENT2)
2C      0 000000 104000     VALIDITYBITSTAB
2D-31 0 000000 000000     AVALUE              CODECORE            DATACORE            DISKLIMIT           ELAPSEDLIMIT
32-35 0 000000 000000     EXCEPTIONTASK       MAXIOTIMETAB        MAXPROCESSTIMETAB MAXWAIT
36      0 000028 000004     MISCTABINFO
37-3A 0 000000 000000     OPTION              PARTNER             PATHCONTROL         PRINTLIMIT
3B      0 000000 000066   =>PRIORITY            = 102
3C-3D 0 000000 000000     PUNCHLIMIT          RESTARTCOUNT
3E      0 007FFF FFFFFF     SAVEMEMORYLIMIT     = 549755813887
3F      0 034000 000000     SERIAL
40      0 000000 001770     STACKLIMIT          = 6000
41      0 000000 00017C   =>STACKSIZE           =0+380=380
42-46 0 000000 000000     STATIONINFO         SWAPSPEX            TAPECOUNT           TAPEPOOL            TARGETTIME
47-4B 0 000000 000000     TASKERROR           TASKLIMIT           TEMPFILELIMIT       USERCODE            (USERCODE2)
4C-4E 0 000000 000000     (USERCODE3)         USERCODEPRIV        WAITLIMIT

5A      5 C00004 F03274     ENVIRONMENT TAB

STACK 034 TCP DATA: 000000 005014  000000 01002E  000000 0E1995  000000 0E1981
ANALYSIS: STATE = NOT BLOCKED, READY, RETURN
          ORIG. BASE PRI = 3866, BASE PRI = 3866, FINE PRI = 01, CLASS = 5
          PROCESS STATISTICS:
          00        000000 000000     MAXPROCESSTIME         = 0
                                      TOTAL MAXPROCESSTIME = 549755813887
          01        410000 01413F     PROCESSTIME            =  0.1973736 SECONDS
                                      TOTAL PROCESSTIME      =   0.194712 SECONDS
          02        530000 47761A     READYTIME              =  11.239896 SECONDS
                                      TOTAL READYTIME        =  11.239896 SECONDS
          03        000000 0000BF     PROCESSOR VISITS
          04        000000 000000
          05        000015 28DB3A     EXPIRATION TIME @ 0:03:23 AFTER HALTLOAD = -0.8542248 SECONDS PRIOR TO DUMP
          06        000000 000000
          07        000000 000000
          08        000000 000001     RETURN PARAMETER 1
          09        000000 000000     RETURN PARAMETER 2
          0A        000000 000000     P2 PARAMETER
          0B        000000 000000
          0C        920A95 E74000     LAST ALIVE   @ 0:03:22 AFTER HALTLOAD =  0.1466328 SECONDS PRIOR TO DUMP.
          0D        920A95 E74143     LAST WAITING @ 0:03:22 AFTER HALTLOAD =  0.1458576 SECONDS PRIOR TO DUMP
          0E        920A95 ED9000     LAST READY   @ 0:03:23 AFTER HALTLOAD = -0.8542248 SECONDS PRIOR TO DUMP
          0F        000000 000000
          10        000000 000000
          11        000000 000000
          12        000000 000000     HEAD OF OWNED EVENTS QUEUE
          13        000000 000000     EVENT WAITING TO PROCURE00000
          14-1F     000000 000000
```

Figure 4–28. STACK Command Output: TAB

STACKWINDOW

The STACKWINDOW command displays selected areas of a stack within a memory dump. The display contains approximately 22 lines of data. In particular, the user's view of the specified stack centers on a particular stack cell. It is suggested that the HISTORY option of the STACK command be used before using the STACKWINDOW command. That option provides a view of the locations of the environments within the stack.

Use the STACKWINDOW command when it is not convenient to print the whole stack on paper.

```
<stackwindow>

— STACKWINDOW ─┬──────────────────────────────────────┬──────────────────┤
               ├─<stack number>─┘ └─ # ──<offset>─────┤
               ├─ + ─┐ ┌───────────────────────────────┤
               ├─ - ─┘ └─<# cells>─────────────────────┤
               └─ ? ───────────────────────────────────┘
```

The following text describes the meaning of each construct:

STACKWINDOW	This form of the command reopens a window created by a STACKWINDOW <stack number> form of the command entered.
STACKWINDOW <stack number>	
STACKWINDOW <stack number> # <offset>	These forms of the command create a window to view the specified stack. If no offset is specified, the view is from the top of the stack. If an offset is specified, the view of the stack centers on that point in the stack. Once a stack window is created, the window is kept unless changed by another STACKWINDOW command. All stack structures are limited to a maximum of five lines of display, and arrays are limited to one line of data values. If the user needs to view more of a particular stack structure, the PV SW or PV SW # <offset> form of the PV command should be used to expand the information displayed.
STACKWINDOW # <offset>	This form of the command allows the user to center the window on the specified point in the current stack.
STACKWINDOW +	
STACKWINDOW –	These forms of the command scroll the window up (+) or down (–) the stack by approximately one screen.
STACKWINDOW + <# cells>	
STACKWINDOW – <# cells>	These forms of the command scroll the window up (+) or down (–) by a specified number of cells.
STACKWINDOW ?	This form of the command displays the stack number and offset of the current window.

For more information, see the discussion of the STACK and PV commands in this section.

Example

Figure 4–29 shows an example of the output from the STACKWINDOW command.

```
STACKWINDOW Ø31 #Ø2A8
     STACKDUMP FOR STACK Ø31    MIX NUMBER 5487/5487        WINDOW @ ØØ2A8
Ø2AD             3 21Ø4B2 5ØEØØ3  RCW:  LL=3, NORML STATE
[USER SEGMENT      @ ØØØ3:ØB25:2 (23661ØØØ)]
                                  SEG DESC:  3 8ØØ149 EØ8Ø1B
                                  CODE:  3 B95Ø32 B8BØ5Ø   3 CDB8AE 51625Ø   3
Ø8AF3Ø Ø27ØØ3  >3 BDAB7Ø Ø4B87Ø<  3 Ø3BDB2 Ø6D5BE
Ø2AC -----D[3]=>3 Ø31ØØ1 44CØØ5  *MSCW: PREVIOUS MSCW @ Ø2A7; D[2]=ØØ14 IN THIS
STACK
Ø2AB (Ø3,ØØØ4)  Ø ØØØØØØ ØØØØØØ
Ø2AA (Ø3,ØØØ3)  5 E4ØØØØ ØØC55Ø  DESC [PRESENT-COPY]: ASD=ØC55Ø, STRING
(8-BIT), INDEX=Ø+Ø (MOM @ ØØ92 IN  THIS STACK)
Ø2A9 (Ø3,ØØØ2)  Ø ØØØØØØ ØØØØ5Ø
Ø2A8             3 21Ø41B DØEØØ3  RCW:  LL=3, NORML STATE
[USER SEGMENT      @ ØØØ3:Ø1BD:2 (126425ØØ)]
                                  SEG DESC:  3 8ØØ149 EØ8Ø1B
                                  CODE:  3 5Ø7EA6 B2Ø49C   3 2AØ35Ø 72BAAE   3
516ØB2 5Ø5Ø72  >3 BDAB5Ø C6B99A<  3 Ø9Ø1AØ Ø1C1AE
Ø2A7 -----D[3]=>3 Ø31ØØ1 44CØØ2  *MSCW: PREVIOUS MSCW @ Ø2A5; D[2]=ØØ14 IN THIS
STACK
Ø2A6             3 21Ø88F 3ØEØØ3  RCW:  LL=3, NORML STATE
[USER SEGMENT      @ ØØØ3:Ø8F3:4 (226968ØØ)]
                                  SEG DESC:  3 8ØØ149 EØ8Ø1B
```

Figure 4–29. STACKWINDOW

STOP

The STOP command terminates DUMPANALYZER. A synonym for STOP is BYE.

<stop>

— STOP ——————————————————————————————|

SUBPORT

The SUBPORT command prints analysis of a specified subport. Groups of subports are associated with different port files. Each subport of a port file can be connected to a different process. There are two syntaxes for this command; one for BNA Version 1 and one for BNA Version 2.

One method of locating subport structures for use in the SUBPORT command involves using the SEARCH command. The PATTERN command should first be used to establish a pattern of E2E4C2D70000 & 0 TAG, and the MASK command should be used to establish a mask of FFFFFFFF0000 & F TAG. If the SEARCH command is then used, the search yields a list of absolute memory addresses of subport structures. These addresses can be used in the SUBPORT AT <address> command, as described in the following paragraphs.

BNA Version 1 Syntax

```
— SUBPORT ─┬─<index>──────────────┬──────────────────────────┤
           ├─ AT ──<address>──────┤
           └─ VIA ──<ASD number>──┘
```

The following text describes the meaning of each construct:

SUBPORT <index>	The <index> value is an index into the PSS array, which is a two dimensional, global data structure maintained by the Lists Module of the MCP. The user can determine the location of the PSS array by using the command *MD G PSS*. The *SUBPORT <index>* command actually selects PSS [<index>,*]. The PSS array contains many different structures, only some of which are subports. DUMPANALYZER verifies that the <index> points to a valid SUBPORT structure by checking word 0, field [47:32] of the element selected (PSS [<index>,0].[47:32]). If this field contains "SUBP" and field [15:16] of the same word is equal to <index>, then the element selected is a valid SUBPORT and is analyzed. If not, an "INVALID SABNO" message is displayed.
SUBPORT AT <address>	The subport at the given absolute memory address is analyzed. If the memory address specified does not contain a subport structure, then DUMPANALYZER displays the message "INVALID SABNO".
SUBPORT VIA <ASD number>	The subport at the address referenced by the supplied ASD number is analyzed. If the address referenced by the supplied ASD number does not contain a subport data structure, then DUMPANALYZER displays the message "INVALID SABNO".

BNA Version 2 Syntax

```
— SUBPORT ─┬─<index>─┬──────┬─┬────────┬─<port index>─┬──────────┤
           │         └─ OF ─┘ └─ PORT ─┘              │
           ├─ AT ──<address>──────────────────────────┤
           └─ VIA ──<ASD number>──────────────────────┘
```

The following text describes the meaning of each construct:

SUBPORT <index> <port index>

SUBPORT <index> OF <port index>

SUBPORT <index> PORT <port index>

continued

continued

SUBPORT <index> OF PORT <port index>	The subport with the given index within the port having the given port index is analyzed. The port index is found in the library information word of the file information block (FIB). The subport index specifies the index of the subport within the port.
SUBPORT AT <address>	The subport at the given address is analyzed.
SUBPORT VIA <ASD number>	An address is analyzed using the <ASD number>.

Example

The following is an example of the first page of output for the SUBPORT command:

```
INPUT: SUBPORT Ø OF PORT 5

SUBPORT ATTRIBUTE BLOCK DESCRIPTOR: 5 8ØØØØ1 FØ2667
ØØ(ØØ) Ø E2E4C2 D7ØØØØ MYSUBPORTADDRESS       =Ø(Ø)
Ø1(Ø1) Ø ØØØØØØ ØØØ5ØØ INTERFACE MODEL        =Ø(Ø)
                       PORTID                 =5(5)
                       LOCALTRANSATT          =Ø(Ø)
                       SUBPORTWASCLOSEDATT    =Ø(Ø)
                       DISCONNECTING          =FALSE
Ø2(Ø2) Ø ØØØ13F ØØØ78Ø PORTINBOUNDSTATE       = 'NOT READY'
                       PORTOUTBOUNDSTATE      = 'NOT READY'
                       SUBPORTOUTBOUNDSTATE   = 'NOT READY'
                       SUBPORTINUSE           =FALSE
                       SUBPORTDATAWASLOST     =FALSE
                       SUBPORT IS READABLE    =FALSE
                       BNAV2 CLEANUP KLUDGE   =FALSE
                       LOGREC PORTINFO SETUP  =FALSE
                       SUBPORTSTATE           =SUBPORT STATE(1)=CLOSED
                       MAXCENSUS              =63(3F)
                       MAXRECSIZE             =192Ø(78Ø)
Ø3(Ø3) Ø ØØØØØØ ØØØØØØ MAXBLOCKEDTIMEOUT      =Ø(Ø)
                       DIALOGCHECKINT         =Ø(Ø)
                       RESUMEREADY            =Ø(Ø)
                       DIALOGPROTOCOLTYPEATT  =Ø(Ø)
Ø4(Ø4) Ø ØØØØØØ ØØØØØØ ICPCOMPRESS            =FALSE
                       ICPCANCOMPRESS         =FALSE
                       COMPRESSIONREQUEST     =FALSE
                       COMPRESSIONCONTROL     =FALSE
                       COMPRESSIONUSEFULATT   =FALSE
                       TRANSLATINGATT         =FALSE
Ø5(Ø5) Ø ØØØØØØ B8ØØØØ PASSIVEOPEN            =FALSE
                       OPENAVAILABLE          =FALSE
                       DONOTSEARCHNETWORK     =FALSE
                       YOURUSERCODEUNSPECIFIED=TRUE
                       YOURHOSTUNSPECIFIED    =FALSE
                       YOURHOSTNULL           =TRUE
```

```
                        YOURHOSTGROUPNUL        =TRUE
                        YOURNAMENULL            =TRUE
                        SB_COLLECT_ERROR        =FALSE
                        RSVP DISPLAY CHANGES    =FALSE
                        IN SECONDARY QUEUE      =FALSE
                        ACTUALCHARACTERSET      =Ø(Ø)
                        SUBPORTERROR            =SUBPORT ERROR(Ø)=NO ERROR
Ø6(Ø6) Ø ØØØØØØ EØØØØØ SPREASONCODE            =Ø(Ø)
                        DIALOGPRIORITY          =Ø(Ø)
                        REQUESTEDDIALOGPRIORITY=Ø(Ø)
                        DIALOGPRIORITYLIMIT     =7(7)
                        MINORSTATE              =MINORSTATE (Ø)=CLOSEDØ
                        SUBPORT IS WRITEABLE    =FALSE
                        SECONDARY IN PROGRESS   =FALSE
                        RESPONSE TYPE           =Ø(Ø)
                        DIALOGUTILIZATION       =Ø(Ø)
```

SUMMARY

The SUMMARY command lists information about all the stacks in the system. The user can control what information is displayed about the stacks and can control the order in which the stacks are displayed.

```
<summary>

— SUMMARY ─┬───────────────────────────────────────┬──────────────────────┤
           │ ┌←──────────────────┐                  │
           └─┴─<summary item>────┴─┬──────────────────┤
                                   └─<sort specs>─────┘

<summary item>

──┬─ CREATION ─────────────────────────┬──────────────────────────────┤
  ├─ KIND ─────────────────────────────┤
  ├─ MIXNO ────────────────────────────┤
  ├─ NAME ─────────────────────────────┤
  ├─ PIB ──┬─<PIB cell name>───────────┤
  │        └─ # ──<offset>─────────────┤
  ├─ RELEASEID ────────────────────────┤
  ├─ STACK ──┬─<stack base cell name>──┤
  │          └─ # ──<offset>───────────┤
  ├─ STACKINFO ────────────────────────┤
  ├─ STACKNO ──────────────────────────┤
  ├─ STACKSTATUS ──────────────────────┤
  ├─ STATUS ───────────────────────────┤
  ├─ TAB ──┬─<TAB cell name>───────────┤
  │        └─ # ──<offset>─────────────┤
  ├─ TCP ──<hex number>────────────────┤
  └─ TYPE ─────────────────────────────┘

<sort specs>

— SORT BY ─┬←──────────────────────────────────────────┐
           └─/3\─ ITEM ──<integer>──┬───────────────┬───┴──────────────────┤
                                    ├─ ASCENDING ───┤
                                    └─ DESCENDING ──┘
```

The following text describes the meaning of each construct:

SUMMARY	The following information is displayed for each stack in the system: stack number, mix number, name, kind, status, and type. Active stacks and the stack that took the dump are identified.
SUMMARY <summary item>	This form of the command allows the user to specify which information is to be displayed for each stack in the system. The stack number is always displayed. There is a restriction on the number of items that can be displayed based on the size of each item selected and the display medium requested (for example, terminal or printer). If the user requests too many items, an error message is displayed. *PIB <PIB cell name>* and *PIB # <offset>* identify the word that is to be displayed from each stack's PIB. *STACK <stack base cell name>* and *STACK # <offset>* identify the word that is to be displayed from each stack. *TAB <stack base cell name>* and *TAB # <offset>* identify the word that is to be displayed from each TAB.
SUMMARY <summary item> <sort specs>	This form of the command allows the user to sort the stacks by up to three of the items specified in the summary item list. The item to be sorted is identified by its numerical position in the summary item list. For the first item in the list, use the <integer> 1; for the second, use <integer> 2, and so forth. Stack number is the default sort item, and any ambiguity remaining after all other sorting is finished is resolved by comparing stack numbers. The user can explicitly sort by stack number by referencing it as ITEM 0, which might be useful if stack number is to be the second of three sort items. The final option for the sort specification is the order that the item should be sorted in; ascending is the default. STACKINFO, STACKSTATUS, and items in the stack or the PIB are sorted numerically; all other items are sorted alphabetically.
CREATION	Displays the creation date and time for the object code file of each process.
KIND	Displays the process kind, such as MCP, independent runner, segment dictionary, or frozen library.
MIXNO	Displays the mix number of each process.
NAME	Displays the name of each process (normally the object code file title).
PIB <PIB cell name> PIB # <offset>	Displays the contents of the specified PIB cell for each process.
RELEASEID	Displays the RELEASEID file attribute of the object code file of each process. This file attribute, if not null, reflects the Mark release level of the object code file.
STACK <stack cell name> STACK # <offset>	Displays the contents of the specified stack cell for each process.

continued

continued

STACKINFO	Displays the STACKINFO word for each process.
STACKNO	Displays the stack number for each process.
STACKSTATUS	Displays the STACKSTATUS word for each process.
STATUS	Displays the current process status, such as READY, WAITING, or FROZEN.
TCP <hex number>	Displays the task control processor (TCP) with which a process is associated.
TYPE	Displays the process type, such as PROCESS, RUN, and so on.

Note: *If TCP is used as a summary item for a dump from a machine that has no TCP, then DUMPANALYZER displays the following error message: "ERROR: SCANNING TCP".*

Examples

The following is an example of the output from a SUMMARY command:

```
INPUT: SUMMARY
STK   MIX #                NAME                KIND   STATUS    TYPE
--- --------- ------------------------------ ---- ---------- ----
Ø12   16/16   *SYSTEM/MCP/39Ø18A ON DISK.    MCP  FROZEN
Ø14    Ø/Ø    ETERNALIR.                     I.R. READY      RUN
Ø2B    Ø/Ø    ANSWERING/SERVICE/Ø4.          I.R. WAITING    RUN
Ø2C    Ø/Ø    AREAMANAGER.                   I.R. ASLEEP     RUN
Ø2D    Ø/Ø    HANDLEIOPEVENTS.               I.R. READY      RUN
Ø2E    Ø/Ø    HUNGIOHANDLER.                 I.R. READY      RUN
Ø33    Ø/Ø    ANABOLISM.                     I.R. WAITING    RUN
Ø34    Ø/Ø    ONESECONDBURDEN.               I.R. READY      RUN
Ø36 4353/4353 *SYSTEM/38/CENTRALSUPPORT.     LIB  WAITING    RUN
Ø37    2/2    CONTROLLOR.                    I.R. READY      RUN
Ø38    3/3    MAINTLINEIR.                   I.R. READY      RUN
Ø39    Ø/Ø    PLM.                           I.R. READY      RUN
Ø3B           *SYSTEM/38/CENTRALSUPPORT ON D SEGD UNEMPLOYED
Ø3C    2/6    *SUBSIDIARYTASK.               I.R. WAITING    PROC
Ø3F 4354/4354 *SYSTEM/GENERALSUPPORT.        LIB  FROZEN     RUN
Ø4Ø           *SYSTEM/GENERALSUPPORT ON DISK SEGD UNEMPLOYED
Ø42 4355/4355 *SYSTEM/WFLSUPPORT.            LIB  FROZEN     RUN
Ø43           *SYSTEM/WFLSUPPORT ON DISK.    SEGD UNEMPLOYED
Ø46 4359/4359 "PRINT_SUBSYSTEM_INITIALIZATIO I.R. WAITING    RUN
Ø48 4361/4361 "JOBFILE/CONVERTER".           I.R. READY      RUN
Ø49 4356/4356 *SYSTEM/DISKCACHESUPPORT/38.   LIB  FROZEN     RUN
Ø4B 4357/4357 *SYSTEM/PRINT/SUPPORT.         LIB  WAITING    RUN
Ø4C           *SYSTEM/DISKCACHESUPPORT/38 ON SEGD UNEMPLOYED
Ø4D    Ø/Ø    "SLOG_MONITOR".                I.R. READY      RUN
Ø4E           *SYSTEM/PRINT/SUPPORT ON DISK. SEGD UNEMPLOYED
Ø4F 4358/4358 *SYSTEM/JOBFORMATTER.          LIB  FROZEN     RUN
Ø5Ø           *SYSTEM/JOBFORMATTER ON DISK.  SEGD UNEMPLOYED
Ø57    Ø/Ø    OLAYSCOUT.                     I.R. READY      RUN
Ø58    Ø/Ø    KEYIN.                         I.R. BLOCKED    RUN  DUMP
```

The following example requests that four items be displayed, and requests that the stacks are sorted by the PIB word CLOCKONTIME in descending order; in other words, the stacks that were most recently on the processor are displayed first.

```
INPUT: SUMMARY MIXNO NAME PIB CLOCKONTIME SORT BY ITEM 3 DESCENDING

STK   MIX #                NAME                 CLOCKONTIME
--- --------- ----------------------------- ---------------
ØF5 2776/2776 "->MARC<-".                   Ø ØØØ387 ØE8A72 ACT
7DB 8297/364Ø (CBG)OBJECT/PM/ACCESS ON PACK. Ø ØØØ387 ØE87EB DUMP
Ø2B    Ø/Ø    IDLE/PROCESSOR/Ø5.            Ø ØØØ387 ØE7265
Ø2D    Ø/Ø    IDLE/PROCESSOR/Ø4.            Ø ØØØ387 ØE719A
ØDC 2726/2763 *SYSTEM/COMS.                 Ø ØØØ387 ØE7ØB7
Ø7C 2722/2722 *SYSTEM/CANDE.                Ø ØØØ387 ØE6B4D
Ø37    2/2    CONTROLLOR.                   Ø ØØØ387 ØE68DØ
Ø6F 2718/2718 "NSP1Ø9/Ø3".                  Ø ØØØ387 ØE1D9B
ØAC 271Ø/2735 SYSTEM/BNAMCS.                Ø ØØØ387 ØDA99Ø
729 2738/5931 STATUS/CHANGE/SFA15AB.        Ø ØØØ387 ØD174C
Ø14 2751/2751 ETERNALIR.                    Ø ØØØ387 ØB953F
Ø6E 2717/2717 "NSP111/Ø4".                  Ø ØØØ387 ØB8DE4
ØEE 2773/2773 (ANGUS)OBJECT/PROGRAM/AGENT/CO Ø ØØØ387 ØB8Ø91
ØCF 271Ø/2754 SYSTEM/X25MONITOR.            Ø ØØØ387 ØA742Ø
ØD8 2726/276Ø COMS/TANK.                    Ø ØØØ387 ØA1873
Ø6D 2716/2716 *SYSTEM/PRINT/ROUTER.         Ø ØØØ387 Ø9DF97
1D5 2738/297Ø TASKING/MESSAGE/HANDLER.      Ø ØØØ387 Ø9D27Ø
Ø7A    Ø/Ø    "CALL_WS_SHERIFF".            Ø ØØØ387 Ø99BØ4
1D4 2738/2969 TASKING/STATE/CONTROLLER.     Ø ØØØ387 Ø98D77
14Ø 2722/285Ø *CANDE/STACKØ1.               Ø ØØØ387 Ø98A4Ø
```

SWAPANAL

Note *The SWAPANAL command has been deimplemented. This command had meaning only on systems with ASN memory, which is not supported on the Mark 3.9 system software release and later releases.*

TAB

The TAB command displays the contents of task attribute blocks (TABs).

```
<tab>

— TAB ─┬─ VIA ─<ASD number>──────┬─┬──────────────────┬───────────────────┤
       └─ AT ─<simple address>───┘ └─<tab constant>───┘
```

The following text describes the meaning of each construct:

TAB AT <simple address>	Displays the contents of the TAB starting at <simple address>.
TAB VIA <ASD number>	Displays the contents of the TAB pointed to by the ASD at <ASD number>.
<tab constant>	Limits the display to include only the TAB word with an offset of <tab constant>.

For more information about tasks, refer to the PIB command.

Example

Figure 4–30 shows an example of the output from the TAB command.

```
INPUT: TAB VIA 2AFO

- - - TAB

00      0 400008 17C331    TIMESTARTED          @ 0:05:25 AFTER HALTLOAD = 6935.84110 SECONDS PRIOR TO DUMP
01      0 266088 35C407    DISKINTPERM
02      0 000000 000000    DISKINTTEMP
03      0 000000 000131    INITPBITKOUNT        = 305
04      0 000000 059206    INITPBITTYME         =  0.8761488 SECONDS
05-09 0 000000 000000      IOCOUNT1             IOCOUNT2          HISTORY           IOTIMETAB         MYSTACKHISTORY
0A-0E 0 000000 000000      OTHERPBITCOUNT       OTHERPBITTIME     PROCESSTIME       STOPPOINT         CARDSPUNCHED
0F-13 0 000000 000000      DISKUSED             HSINFO            ITINERARY         LINESPRINTED      PROCESSINFO
14-16 0 000000 000000      SURROGATEINFO        TASKPARAMS        TEMPFILEMBYTES
17      0 002201 000000    WFLJOBINFO
18      0 000000 000000    ACCESSCODE
19      5 800000 202B15    BACKUPFAMILY         = SYSKITS
1A-1C 0 000000 000000      CHARGECODE           DBEQTNBLK         DESTNAME
1D      5 800000 402AF8    FAMILYSUB            DISK = MCPMAST OTHERWISE DISK
1E-1F 0 000000 000000      HOSTNAME             JSTITLE
20      5 800000 502B14    MYBDNAME             = LISTING/39COMSUTIL/BIND
21-22 0 000000 000000      MYFPB                MYLPB
23      5 800000 802AF1    MYNAME               = (MIKEB)OBJECT/PROGRAM/AGENT/COMS ON MCPMAST
24-26 0 000000 000000      MYPPB                MYPRINTDEFAULTS   SUPPRESSWARNING
27      5 800000 302AF3    SOURCENAME           = PSEUDO00000
28      0 000000 000000    SUBSYSTEMID
29      5 800000 402AF4    TASKMCSNAME          = *SYSTEM/COMS ON DISK
2A-2E 0 000000 000000      TASKSTRING           LOCKEDEVENT       (LOCKEDEVENT2)    PARTNERGRAPHHEAD  PARTNERGRAPHLINKS
2F      0 000020 304421    VALIDITYBITSTAB
30      0 000000 000000  =>AVALUE
31      0 000000 004E86    CODECORE
32      0 000000 00250F    DATACORE
33-34 0 000000 000000      DISKLIMIT            ELAPSEDLIMIT
35      5 E00000 00210E  =>EXCEPTIONTASK
36-38 0 000000 000000      MAXIOTIMETAB         MAXPROCESSTIMETAB MAXWAIT
39      0 000050 000004    MISCTABINFO
3A      0 000000 080162  =>OPTION
3B      0 000000 000000    PARTNER
3C      0 000000 02808D    PATHCONTROL          (ORG MCS=2 LSN=141)
3D      0 000000 000000    PRINTLIMIT
3E      0 000000 000050  =>PRIORITY             = 80
3F-40 0 000000 000000      PUNCHLIMIT           RESTARTCOUNT
41      0 007FFF FFFFFF    SAVEMEMORYLIMIT      = 549755813887
42      0 08601F 8C1F8C    SERIAL
43      0 000000 001770    STACKLIMIT           = 6000
44      0 800013 8004FD  =>STACKSIZE            =312+1277=1589
45      0 400000 00008D  =>STATIONINFO
46-49 0 000000 000000      SWAPSPEX             TAPECOUNT         TAPEPOOL          TARGETTIME
4A      0 000009 000000    TASKERROR            = 150994944
4B      0 05C000 000000    TASKLIMIT            TASKDECLARER=05C
4C      0 000000 000000    TEMPFILELIMIT
4D      0 090101 05D4C9  =>USERCODE             = MIKEB
4E      0 D2C5C2 000000    (USERCODE2)
4F      0 000000 000000    (USERCODE3)
50      0 000000 004003    USERCODEPRIV         PRIVLEDGED USER, MCS ABSTAINING, HS SPECIAL PRIVS ALLOWED
51      0 000000 000000    WAITLIMIT
```

Figure 4–30. TAB Command Output

TCPINFO (RMM Systems)

The TCPINFO command displays state information that is maintained by the task control processor (TCP). Because only RMM systems have TCPs, this command is valid only on dumps from RMM systems.

```
<tcpinfo>

— TCPINFO ─┬──────────────────────────────┬──────────────────────────┤
           ├─ ALL ────────────────────────┤
           ├─ EVENT ─┬─ ALL ──────────────┤
           │         └─<hexadecimal number>┤
           └─ STACK ─┬─ ALL ──────────────┤
                     └─<hexadecimal number>┘
```

The following text describes the meaning of each construct:

TCPINFO	This simple form of the command displays the number of events and process stacks that have been allocated.
TCPINFO ALL	Displays state information for all process stacks and events.
TCPINFO EVENT ALL	Displays state information for all events.
TCPINFO EVENT <hexadecimal number>	Displays state information for the specified event.
TCPINFO STACK ALL	Displays state information for all process stacks.
TCPINFO STACK <hexadecimal number>	Displays state information for the specified process stack.

Example

Figure 4–31 shows an example of the output from a TCPINFO STACK <hexadecimal stack number> command and a TCPINFO EVENTS ALL command.

```
INPUT: TCPINFO STACK CA

STACK OCA TCP DATA: 000000 085001  000000 014000  000000 061424  000000 061401
ANALYSIS: STATE = BLOCKED, DSED, EXIT
          ORIG. BASE PRI = 1850, BASE PRI = 1850, FINE PRI = 01, CLASS = 5
   _      PROCESS STATISTICS:
          00       000000 000000    MAXPROCESSTIME        = 0
                                    TOTAL MAXPROCESSTIME  = 549755813887
          01       5C0000 02443B    PROCESSTIME           =  0.3564936 SECONDS
                                    TOTAL PROCESSTIME     =  0.3138264 SECONDS
          02       240000 005919    READYTIME             =  0.0547416 SECONDS
                                    TOTAL READYTIME       =  0.0547416 SECONDS
          03       000000 0000A4    PROCESSOR VISITS
          04       000000 000000
          05       00001D 1B0420    EXPIRATION TIME @ 0:03:58 AFTER HALTLOAD =  -8.694696 SECONDS PRIOR TO DUMP
          06       000000 000000
          07       000000 000000
          08       000000 000002    RETURN PARAMETER 1
          09       000000 000000    RETURN PARAMETER 2
          0A       000000 000000    P2 PARAMETER
          0B       000000 000000
          0C       1E0E8D 7BE031    LAST ALIVE   @ 0:03:57 AFTER HALTLOAD =  -7.721604 SECONDS PRIOR TO DUMP.
          0D       1E0E8D 7BB44A    LAST WAITING @ 0:03:57 AFTER HALTLOAD = -7.6946304 SECONDS PRIOR TO DUMP
          0E       1E0E8D 7BE026    LAST READY   @ 0:03:57 AFTER HALTLOAD = -7.7215776 SECONDS PRIOR TO DUMP
          0F       000000 000000
          10       000000 000000
          11       000000 000000
          12       000000 000000    HEAD OF OWNED EVENTS QUEUE
          13       00000B BADBAD    EVENT WAITING TO PROCUREADBAD
          14-1F    000000 000000

:READY

INPUT: TCPINFO EVENTS ALL

   760 EVENTS ALLOCATED IN TCP

EVENT 00001      : 000000 000000  000000 000000  000000 000000  C08000 000000 NOT HAPPENED
EVENT 00002-0000F: 000000 000000  000000 000000  000000 000000  C00000 000000 NOT HAPPENED
EVENT 00010-00020: 000000 000000  000000 000000  000000 000000  C08000 000000 NOT HAPPENED
EVENT 00021-00022: 000000 000000  000000 000000  000000 000000  C00000 000000 NOT HAPPENED
EVENT 00023      : 000000 000000  000000 000000  000000 000000  C08000 000000 NOT HAPPENED
EVENT 00024      : 000000 000000  000000 000000  000000 000000  C00000 000000 NOT HAPPENED
EVENT 00025      : 000000 000000  000000 000000  000000 000000  C48000 00002C NOT HAPPENED, WAITING STACKS: 02C
EVENT 00026      : 000000 000000  000000 000000  000000 000000  C00000 000000 NOT HAPPENED
EVENT 00027-00029: 000000 000000  000000 000000  000000 000000  C08000 000000 NOT HAPPENED
EVENT 0002A-0002B: 000000 000000  000000 000000  000000 000000  C00000 000000 NOT HAPPENED
EVENT 0002C      : 000000 000000  000000 000000  000000 000000  C08000 000000 NOT HAPPENED
EVENT 0002D-0002E: 000000 000000  000000 000000  000000 000000  C00000 000000 NOT HAPPENED
EVENT 0002F      : 000000 000000  000000 000000  000000 000000  D08000 000000 HAPPENED
EVENT 00030      : 000000 000000  000000 000000  000000 000000  D00000 000000 HAPPENED
EVENT 00031      : 000000 000000  000000 000000  000000 000000  C48000 000035 NOT HAPPENED, WAITING STACKS: 035
EVENT 00032      : 000000 000000  000000 000000  000000 000000  D00000 000000 HAPPENED
EVENT 00033-00034: 000000 000000  000000 000000  000000 000000  C08000 000000 NOT HAPPENED
EVENT 00035      : 000000 000000  000000 000000  000000 000000  C00000 000000 NOT HAPPENED
EVENT 00036      : 000000 000000  000000 000000  000000 000000  C08000 000000 NOT HAPPENED
EVENT 00037      : 000000 000000  000000 000000  000000 000000  C48000 00002F NOT HAPPENED, WAITING STACKS: 02F
EVENT 00038-00039: 000000 000000  000000 000000  000000 000000  C00000 000000 NOT HAPPENED
EVENT 0003A-0003B: 000000 000000  000000 000000  000000 000000  D00000 000000 HAPPENED
EVENT 0003C-0003F: 000000 000000  000000 000000  000000 000000  C00000 000000 NOT HAPPENED
EVENT 00040      : 000000 000000  000000 000000  000000 000000  C08000 000000 NOT HAPPENED
EVENT 00041-00042: 000000 000000  000000 000000  000000 000000  D00000 000000 HAPPENED
EVENT 00043-00044: 000000 000000  000000 000000  000000 000000  C08000 000000 NOT HAPPENED
EVENT 00045-00047: 000000 000000  000000 000000  000000 000000  C00000 000000 NOT HAPPENED
```

Figure 4–31. TCPINFO Command Output

TERMINAL

The TERMINAL command permits the user to control the following features of terminal output when viewing DUMPANALYZER output:

- Whether the terminal output is sent a page at a time or a line at a time
- The number of characters per line
- The number of lines per page
- Whether the pages are transmitted continuously or with page breaks
- Whether the output is right-justified
- Whether characters at the end of a line are truncated or printed on the next line
- Whether the characters appear in upper- and lowercase or only in uppercase

The default values for the TERMINAL command are as follows:

- FULLPAGE ON (for screen devices); FULLPAGE OFF (for nonscreen devices)
- WAIT
- PAGE 22
- LINE 80
- TRUNCATE OFF
- UPPERCASE OFF
- RIGHTJUSTIFY OFF

The TERMINAL command can be used at any DUMPANALYZER prompt.

<terminal>

```
                ┌<──────────────────────────────┐
── TERMINAL ────┴┬──────────────────────────────┬┴──────────────────────────┤
                 ├─/1\─ FULLPAGE ─┬─────────────┤
                 │                └─ OFF ───────┤
                 ├─/1\─┬─ CONTINUOUS ───────────┤
                 │     └─ WAIT ─────────────────┤
                 ├─/1\─ PAGE ──<number>─────────┤
                 ├─/1\─ LINE ──<number>─────────┤
                 ├─/1\─ TRUNCATE ─┬─────────────┤
                 │                └─ OFF ───────┤
                 ├─/1\─ UPPERCASE ─┬────────────┤
                 │                 └─ OFF ──────┤
                 └─/1\─ RIGHTJUSTIFY ─┬─────────┤
                                      └─ OFF ───┘
```

The following text describes the meaning of each construct:

TERMINAL	This form of the command displays the current values for all of the TERMINAL options.
TERMINAL FULLPAGE (OFF)	This form of the command causes output to be sent a page (or line) at a time.
TERMINAL CONTINUOUS	This form of the command causes terminal output to be sent without pauses between pages.

continued

continued

TERMINAL WAIT	This form of the command causes terminal output to wait for user input between pages. Terminal output is continued by entering one or more blanks. If a positive number (N or +N) is entered, only the next N lines are displayed instead of the next page. Any other input entered at a page break is a new command; output is discontinued for the previous command.
TERMINAL PAGE <number>	This form of the command sets the number of lines per page.
TERMINAL LINE <number>	This form of the command sets the number of characters per line.
TERMINAL RIGHTJUSTIFY (OFF)	This option is valid only when TRUNCATE is disabled. When a line of output is longer than the terminal width, the excess characters are right justified on the following line if the RIGHTJUSTIFY option is enabled.
TERMINAL TRUNCATE (OFF)	This form of the command causes output that does not fit on a line to be right justified on the following line. The RIGHTJUSTIFY option must be enabled when the TRUNCATE option is enabled. When the TRUNCATE option is disabled, output that does not fit on a line is truncated.
TERMINAL UPPERCASE (OFF)	When the UPPERCASE option is enabled, lowercase output is converted to uppercase. When the UPPERCASE option is disabled, lowercase output is not converted to uppercase.

Example

DUMPANALYZER displays the current terminal settings in response to each TERMINAL command, as shown in the following example:

```
INPUT: TERM RJ
FULLPAGE ON, WAIT, PAGE = 23, RJ ON, TRUNCATE OFF, UP OFF, LINE = 80
```

TRACE

The TRACE command causes the current tracetable, or a portion of the tracetable, to be printed. The tracetable is generated by the trace facility, a general purpose operating system debugging aid available when the MCP compile-time option TRACE is set. The TRACE facility allows for tracetable entries to be made at predefined points in the MCP called CONDITIONALDUMP stops, each of which is associated with a particular statement in the MCP. Each stop has two parameters, a stop number and an info word. The stop number is a two digit hex number that is different for each stop. A new set of CONDITIONALDUMP stops is in effect for each Mark release level. Information about the conditions under which the stops are reached is given in the tracetable entry. The tracetable holds about 1650 entries in a circular queue; when the last slot has been filled, the next entry overwrites the first entry.

```
<trace>

                                 ┌←──────────────────────────┐
— TRACE ─┬─────────────────────┬─┴─┬─ CD ─┬───────────────────┬─┴──────────┤
         └─<entries number>────┘   │      │ ┌←───────────────┐│
                                   │      └─┴─<CD number>────┴┤
                                   ├─ CPM ─<CPM number>───────┤
                                   ├─ SHOW ───────────────────┤
                                   └─ STACK ─<stack number>───┘
```

<entries number>
<CD number>
<CPM number>
<stack number>

These are each <number>s.

The following text describes the meaning of each construct:

TRACE	The entire tracetable is printed.
TRACE <entries number>	The most recent <entries number> of entries to the tracetable are printed.
TRACE CD <CD number>	Tracetable entries with the specified conditional dump number are printed.
TRACE CPM <CPM number>	Tracetable entries with the specified central processing module (CPM) number are printed.
TRACE SHOW	All SHOW messages are printed. SHOW messages are displayed in EBCDIC and DEADSTOP information is displayed in hexadecimal form.
TRACE STACK <stack number>	Tracetable entries in the specified process stack are printed.
TRACE <entries number> CD <CD number> TRACE <entries number> CPM <CPM number> TRACE <entries number> SHOW TRACE <entries number> STACK <stack number>	The most recent <entries number> of entries to the tracetable are searched, and the entries meeting the specified CD, CPM, SHOW, or STACK criteria are printed.

Example

Figure 4–32 shows an example of the output from the TRACE command.

```
INPUT: TRACE 15
** MOST RECENT ENTRIES APPEAR FIRST **

CPM STK  CD#   P2 PARAMETER   IN PROCEDURE               @ SEQ# / RCW  CALLED FROM PROCEDURE        @ SDI:PIR:PSR / SEQUENCE
--- ---  ---   ------------   ------------               - ----   ---  ---------------------        - -----------   --------
 4  058  18  1 4001C1 0A0020  OPENLTFILE                   (12511905)  LOGHANDLER                     1B74:043A:2 (36907440)
 4  058  18  1 4001A1 0A0020  OPENLTFILE                   (12511905)  LOGHANDLER                     1B74:0430:2 (36907440)
 4  058  18  1 4001A1 0A1020  OPENLTFILE                   (12511905)                                 1B72:0089:1 (36869760)
 4  058  6F  0 0A0BC2 B00409  INITIATEUSERPSEUDOIO         (29085255)  INITIATEUSERPSEUDOIO           103E:07E0:5 (29179920)
 4  058  63  0 000115 804809  INITIATEUSERPSEUDOIO         (29085255)  INITIATEUSERPSEUDOIO           103E:07E0:5 (29179920)
 4  058  27  0 250100 000000  INITIATEUSERPSEUDOIO         (29085230)  INITIATEUSERPSEUDOIO           103E:07E0:5 (29179920)
 4  058  26  0 000115 800000  INITIATEUSERPSEUDOIO         (29085230)  INITIATEUSERPSEUDOIO           103E:07E0:5 (29179920)
 4  058  6F  0 0A0BC2 B00434  INITIATEDIRECTCHARIO         (29291240)  UDCACHESIZE                    11C8:004F:2 (76981260)
 4  058  63  0 000000 004834  INITIATEDIRECTCHARIO         (29291155)  UDCACHESIZE                    11C8:004F:2 (76981260)
 4  058  62  0 BC2A00 404819  INITIATEDIRECTCHARIO         (29252500)  UDCACHESIZE                    11C8:004F:2 (76981260)
 4  058  6F  0 0A0BC2 A0041E  INITIATEUSERPSEUDOIO         (29134220)  INITIATEDIRECTCHARIO           103E:0C14:4 (29232520)
 4  058  63  0 000000 00481E  INITIATEUSERPSEUDOIO         (29134220)  INITIATEDIRECTCHARIO           103E:0C14:4 (29232520)
 4  058  62  0 BC2800 404819  INITIATEUSERPSEUDOIO         (29097830)  INITIATEDIRECTCHARIO           103E:0C14:4 (29232520)
 4  058  09  0 005800 020000  BLAST_WAITING_STACK          (13509160)  DUMPDCDLP                      1195:0002:0 (32197771)
 4  02D  09  0 002D00 2D0003  SUPERWAIT                    (14650720)  BLAST_WAITING_STACK            1021:045B:3 (13509160)
```

Figure 4–32. TRACE Command Output

USE

The USE command lists commands that were saved by the KEEP command. (Refer to the KEEP command.)

```
<use>

— USE ─┬─<number>─┬──────────────────────────────────┤
       └─ ? ──────┘
```

The following text describes the meaning of each construct:

USE <number>	The command that was saved and labeled <number> by the KEEP command is listed. The <number> variable must evaluate to a decimal integer between 0 and 9.
USE?	A list of the commands saved by the KEEP command is printed.

Example

```
INPUT: USE ?

 ØØ  : HDR
 Ø1  : MD 1C33 FOR 6
 Ø2  : MD STK 6A BOSR FOR 3
```

For more information, see the discussion of the KEEP command in this section.

WHERE

The WHERE command displays the location of a specified <global ID>. The location is expressed as the offset relative to D[0].

All aliases for MCP stack cells are retained and recognized by the WHERE command.

```
<where>

— WHERE —<global ID>────────────────────────────────┤
```

Example

```
INPUT: WHERE GETJOBINFO
SEG #156A GETJOBINFO OF JOBREQUEST OF GETSTATUS
PCW @ Ø321 SEG #1111 GETJOBINFO
```

WHO

The WHO command displays the global identifier for the specified D[0] offset (cell). An outer-block procedure PCW is shown as SEG:PIR:PSR from the D[0] stack image. For an outer-block procedure code segment, the names, PCW D[0] offsets, and PCWs are shown for all procedures in that segment.

All aliases for MCP stack cells are retained and are returned by the WHO command.

<who>

```
— WHO —<number>——————————————————————————————————————————————————|
```

Examples

```
INPUT: WHO 321
Ø321 (PCW 1111:ØØØØ:Ø) GETJOBINFO

INPUT: WHO 32Ø
32Ø (PCW 7D6:ØØ44:Ø) CCCHECK FORMORE
```

Error Messages

The following error messages are displayed if an abnormal condition occurs.

ACCEPT: WRONG CODE FILE - OK OR RESTART.

The code file timestamp on disk did not match the timestamp on the dump tape. The operator should enter OK to use the code file on disk or RESTART to cause the code file to be closed and DUMPANALYZER to be suspended. DUMPANALYZER proceeds if OK is specified.

DISPLAY: DUMP TAPE HAS BAD INFORMATION IN RECORD <#>, AT LOCATION <#>.

Data on the dump tape failed the consistency check, causing DUMPANALYZER to terminate.

DISPLAY: BAD DATA RECOVERY IN RECORD <#>, AT LOCATION <#>.

Data on the dump tape failed the consistency check, but redundancies in the way that these data were stored on the tape allowed limited recovery of that data. DUMPANALYZER then proceeds.

DISPLAY: BAD DUMPANALYZER INPUT CARDS

DUMPANALYZER was unable to decipher an input card. The card image appears on the printout with a line of asterisks (*) pointing to the unknown word, and processing terminates.

DISPLAY: BAD MCP STACK POINTER

DUMPANALYZER found that the stack vector descriptor at D[0]+2 did not address present memory. This condition is usually due to a premature end-of-file condition on the dump tape or to improperly taking the last memory module offline when no MEMDUMP disk is available, causing termination of processing. DUMPANALYZER should be rerun with RAWDUMP and DEBUG set; DUMPANALYZER then produces a raw dump of the contents of the tape without checking D[0]+2.

DUMPANALYZER FAULT <#> messages are given for the first, second, third, fourth, and tenth faults and for every tenth fault thereafter (20, 30, 40, and so forth).

DISPLAY: CANNOT ANALYZE - MCP INCOMPATIBLE WITH DUMPANALYZER.

The MCP level recorded on the dump tape did not match the level of DUMPANALYZER. This incompatibility can be avoided by ensuring that the level of DUMPANALYZER is always exactly the same as the level of the MCP that wrote the dump tape.

DISPLAY: CANNOT ANALYZE – USE PREVIOUS DUMP ANALYZER

The level of the MCP as recorded on the dump tape was lower than the level compiled into DUMPANALYZER, causing DUMPANALYZER termination. The proper level of DUMPANALYZER must be used for a rerun.

DISPLAY: ERROR UNABLE TO GENERATE GLOBAL IDENTIFIERS, CAUSE = <cause>.

One of four <cause>s caused failure in the MCP global identifier routine. Two of these <cause>s indicate either improper compilation, a DUMPANALYZER bug, or code file corruption (LEVEL and SIZE). I/O exceptions give rise to the other two <cause>s, RDMCP (code file) and RDFILE (internal file). Global name generation is terminated, NONAMES is invoked, and DUMPANALYZER analysis proceeds. DUMPANALYZER can be rerun with the DEBUG command to get a program dump of the terminated global identifier routine.

DISPLAY: SAVE ABORTED FOR INSUFFICIENT DISK SPACE

The SAVE command was aborted because there was not enough disk space available to hold the saved dump file.

DISPLAY: SAVE ABORTED FOR DIRECT I/O ERROR, RD=<result descriptor>

The SAVE command was aborted due to a disk unit read or write error. The <result descriptor> gives information about the error.

Section 5
HARDCOPY and PRINTCOPY

This section describes the HARDCOPY and PRINTCOPY utility programs, which provide a way to save operator display terminal (ODT) input commands and system messages in a disk file and print the contents of that file.

HARDCOPY

The HARDCOPY utility program uses the DCALGOL intrinsic ATTACHSPOQ for initialization to receive copies of all ODT "traffic," which includes all input commands and system responses. System responses are received in the HARDCOPY task language. All messages received by HARDCOPY are reformatted by replacing control characters with blanks, condensing multiple blanks to a single blank, and dividing messages into 132-character print lines. HARDCOPY then writes the reformatted commands and messages into a disk file titled *HARDCOPY. This disk file is created with a PROTECTION file attribute value of PROTECTED.

On systems running InfoGuard security enhancement software, the *HARDCOPY file can be public or private, depending on the value of the NONUSERFILES option of the SECOPT (Security Options) system command. For information about this option, refer to to *A Series Security Administration Guide.* On non-InfoGuard systems, the *HARDCOPY file is public.

When the HARDCOPY program is initiated, it searches for an existing file titled *HARDCOPY. If the file HARDCOPY exists, the last recorded message is located, and a message indicating a restart is added at the end of the file. If the file HARDCOPY is not found, one is created. Every 15 minutes, a message giving the time and date can be inserted into the file. The insertion is done only if entries have been made in HARDCOPY.

After the first message that extends into the last segment of HARDCOPY is written, the file HARDCOPY is closed and its title is changed to HARDCOPIES/<integer>. The integer is incremented by one each time a new HARDCOPIES file is created. A new file titled HARDCOPY is created at this time. PRINTCOPY is automatically initiated to print the file.

The contents of HARDCOPY can also be printed in sequential pieces. Each time a HI (Cause EXCEPTIONEVENT) system command is directed at HARDCOPY, a run of PRINTCOPY is requested that prints all messages since the last HI-initiated printing. (Refer to the *A Series System Commands Operations Reference Manual* for further information about the HI command.)

Running HARDCOPY

In order to maintain a hard copy of ODT traffic, an installation must first ensure that HARDCOPY and PRINTCOPY are loaded on disk, and then do one of the following:

- Designate HARDCOPY as a supervisor program by entering the following CS (Change Supervisor) system command. (Specifying HARDCOPY as a supervisor program causes the program to be automatically initiated after a halt/load.)

```
CS SYSTEM/HARDCOPY
```

- The following is an example of a Work Flow Language (WFL) job that runs HARDCOPY:

```
BEGIN JOB;
   RUN SYSTEM/HARDCOPY;
END JOB
```

Disk File Format

Record number 0 of a HARDCOPY file contains the following information, which is used for naming copies and controlling PRINTCOPY:

Word	Contents
0	The integer used for naming the file
1	The record number of the first word following the last recorded message
2	The number of valid words within the record specified by word 1
3	The record number containing the first word of the first message that has not been printed by a HI (Cause EXCEPTIONEVENT) system command
4	The number of valid words within the record specified by word 3, which precedes the first unprinted record
5–29	Not used

The remaining records contain ODT input commands and system messages. The first word of each message has field 47:8 equal to 1 and field 39:40 equal to the length of the message (not including the first word) as five EBCDIC numeric characters. The last message in the file is always followed by a word with all bits equal to zero.

PRINTCOPY

The PRINTCOPY utility program is primarily a disk-to-printer program. PRINTCOPY reads records from a HARDCOPY disk file and writes them to a direct line printer file.

Running PRINTCOPY

The following is an example of a WFL job that runs PRINTCOPY:

```
BEGIN JOB;
  RUN SYSTEM/PRINTCOPY (<integer>);
END JOB
```

The <integer> variable is a value that determines which file and how much of the file is to be printed. If <integer> is zero, the file selected is HARDCOPY. Otherwise, the file selected is HARDCOPIES/<file number> where <file number> is the absolute value of <integer>. If a HARDCOPIES/<file number> file is selected, it will be removed after being printed.

If a negative <integer> is specified, only previously unprinted portions of the file are printed. Otherwise the entire file is printed. When HARDCOPY initiates PRINTCOPY in response to a HI command, HARDCOPY passes a negative value to PRINTCOPY.

The following example specifies that the file HARDCOPIES/3 is to be printed:

```
RUN SYSTEM/PRINTCOPY (3)
```

Section 6
HDU System Balancing

System balancing is a feature of the operating system that is used to balance or tune an HDU (A 12 or A 15) system to a set of operations. The user can monitor the overall system utilization and can change various software parameters that affect system utilization.

SYSTEMSTATUS is used to collect the system utilization statistics. The U (Utilization) system command is used to display the current system utilization statistics, including percentages of central processor unit (CPU), I/O, operating system (MCP), and total system use. The U command is described in the *A Series System Commands Operations Reference Manual*.

The Statistics display is divided into two parts: processing utilization and I/O utilization. These statistics and the accounts that they are applied to are described in the *A Series System Commands Operations Reference Manual* and the *A Series SYSTEMSTATUS Programming Reference Manual*.

Dynamic Variation of System-Balancing Parameters

Dynamic Variation of System-Balancing Parameters is the ability to change, while the operating system is running, certain software parameters that control the flow of the operating system. In particular, the user is given the ability to specify the operating system I/O interrupt scheme and the ability to change the system utilization time interval. Both of these parameters can be altered or interrogated by using the SBP (System Balancing Parameters) system command. (Refer to the *A Series System Commands Operations Reference Manual* for more information about the use of this command.)

The keyword SBP can be entered followed by one of two parameters: INTERVAL (used to change the system utilization time interval) and IOINTERRUPT (used to change the operating system I/O interrupt schemes). SBP with no parameter following it causes the current values of INTERVAL and IOINTERRUPT to be displayed. The INTERVAL is expressed in terms of seconds and IOINTERRUPT is a choice of four interrupt strategies. The SBP command can take one of the following forms:

```
SBP (request display of current balancing parameters)
SBP INTERVAL <number> (for example, SBP INTERVAL 300)
SBP IOINTERRUPT QEMPTY
SBP IOINTERRUPT IOFINISH
SBP IOINTERRUPT IDLE
SBP IOINTERRUPT WAITING
SBP IOINTERRUPT WAITING QEMPTY
```

INTERVAL Parameter

INTERVAL is a user-specified parameter, and has a default value of 10 seconds (a small time interval). A small interval gives an immediate picture of system utilization, and a large interval gives a more overall, averaged picture. With a small interval, radical changes are accurately reflected. With a large interval, radical changes are "smoothed out." The user can use a larger interval for an overall picture, but one smaller than 10 seconds is not recommended because the statistics tend to fluctuate too much.

IOINTERRUPT Parameter

IOINTERRUPT is the system balancing feature that allows a choice of four I/O interrupt strategies. The differences among the four involve the way various types of jobs interact and include the amount of CPM time devoted to handling I/O completes. The four interrupt strategies are as follows:

QEMPTY	Specifies that the IOM or IOMs interrupt a CPM when the last I/O request for any unit is completed and on every I/O completion if a CPM is idle.
IOFINISH	Specifies that IOM or IOMs interrupt a CPM upon handling every I/O completion. This causes I/O bound jobs to run in a shorter elapsed time, but requires more CPM time to handle I/O completions.
IDLE	Provides for no I/O interrupts except when a CPM is idle. The effect of this is to bias CPM usage toward CPM-bound jobs. This requires the least amount of CPM time for handling I/O completions.
WAITING	Specifies that I/O interrupts occur upon completion of an I/O operation only if a process is waiting for that I/O or if a CPM is idle. As long as the progress of no process is dependent on the completion of a particular I/O operation, the IOM generates no interrupt. The effect of this strategy is to use just enough CPM time to keep I/O bound jobs running.

This WAITING interrupt strategy can be specified along with QEMPTY by using the following form of the SBP system command:

```
SBP IOINTERRUPT WAITING QEMPTY
```

IOINTERRUPT WAITING QEMPTY is the default strategy when none is specified.

Implementation

The SETSTATUS procedure of the operating system handles the SBP command. Within SETSTATUS, operating system global variables are altered to change the interval or IOINTERRUPT strategy. If the user wishes to change the system balancing parameters from a user program, he can use the DCKEYIN function of DCALGOL, or he can invoke SETSTATUS, in which case he should refer to the SYSTEMBALANCINGPARAMETERS procedure in the CONTROLLER software for specific details of input formats, and so forth.

System-Balancing Usage

System balancing can be used in a number of ways. By observing system utilization and changing system-balancing parameters, the user can “tune” the system to help achieve maximum throughput. The user can also “bias” the system toward a specific type of application; for instance, IOINTERRUPT on idle biases the system towards process-bound jobs and IOINTERRUPT on every I/O FINISH biases the system towards IO-bound jobs. Another use is to record and graph utilization throughout the day, enabling the user to distribute the system load on an equitable basis. In general, system balancing should enhance the user’s knowledge and control of the system.

Section 7
LOGANALYZER

The LOGANALYZER program is designed to produce a report consisting of all SUMLOG entries that correspond to parameters set by the user. LOGANALYZER extracts the specified types of entries from the log and formats them for display. The report LOGANALYZER produces can be listed on a terminal screen, sent to a printer, or placed in a file.

LOGANALYZER examines any log file specified by the user. However, if no log file is specified, it analyzes the current SUMLOG file. The user can limit the LOGANALYZER search to entries in the log file that were made during a particular time period.

Any of various <loganalyzer options> can be used to extract log entries of a certain type, such as log entries that record job activity or that record various kinds of errors. If none of these <loganalyzer options> are specified, all entry types are included in the report. A description of all types of log entries can be found in "SUMLOG" in this manual.

Installing Related Libraries

LOGANALYZER uses the system libraries JOBFORMATTER and SDASUPPORT. These libraries must be present as support libraries that appear in the SL (Support Library) system command display.

Depending on user privileges and the logfile security set for systems with InfoGuard installed, the SDASUPPORT library is used as a filter on the analyzed SUMLOG file records. Refer to the USERCODE option for a description of the SDASUPPORT filtering procedure.

Additional system libraries are required for analysis of certain log entry types. The following are these libraries, which should be defined by the SL system command:

- The BNAV2TRANSLATION library is necessary for analysis of BNA Version 2 log entries.
- The HWERRORSUPPORT library is necessary for complete analysis of Major Type 2, Minor Type 23 (Mainframe Hardware Report) entries. If this library is not available, LOGANALYZER performs a partial analysis of these log entries.
- The SITESUPPORT library is necessary for analysis of Major Type 7 (Installation) entries. It can also be used for customizing log reports. If this library is unavailable, LOGANALYZER generates a HEX dump of the SUMLOG entry for Major Type 7 or performs the standard formatting of the SUMLOG entries for any other Major Type.

Analyzing Logs from Different Releases

The current LOGANALYZER is able to analyze logs generated under the same Mark release level or the immediately preceding Mark release level. For example, the 4.0 LOGANALYZER could analyze 4.0 or 3.9 logs.

It is also possible to run LOGANALYZER on a system running a different Mark release. LOGANALYZER automatically checks the JOBFORMATTER and SDASUPPORT libraries to determine if they are of the same Mark release as LOGANALYZER. If not, LOGANALYZER attempts to link to these libraries under function names prefixed with the Mark level – for example, 40JOBFORMATTER and 40SDASUPPORT. The operator should use the SL command to install these libraries under the correct function names.

Finding Missing Log Entries

If the LOGANALYZER report does not include log entries that you expected to be there, it might be due to one of the following reasons:

- You might need to use the LOGGING (Logging Options) system command to enable logging of the specified log entry type. Refer to “Selecting the Entry Types to be Logged” in Section 12, “SUMLOG.”
- On InfoGuard systems, by default LOGANALYZER excludes entries with a usercode different from the usercode under which the LOGANALYZER process runs. This is the case even if LOGANALYZER is running under a privileged usercode. To see all log entries, it might be necessary to run LOGANALYZER under a privileged usercode and also to specify “USERCODE .” in the LOGANALYZER command. Refer to the discussion of the USERCODE option later in this section.

Running LOGANALYZER

LOGANALYZER can be initiated from a CANDE or MARC session, an ODT, or a WFL program. The following table shows the commands or selections available for initiated LOGANALYZER from each of these sources:

Source	Input
CANDE, ODT, WFL	*LOG <option list>* command
CANDE, MARC, ODT, WFL	*RUN *SYSTEM/LOGANALYZER ("<option list>")* command
MARC	LOG selection on LOG menu

Note: *When the RUN command is used, the entire <option list> must be enclosed in quotation marks. For individual options that include string values, these string values must be enclosed in double sets of quotation marks.*

If LOGANALYZER is to be run on an operator display terminal (ODT) configured to be in data comm mode, it should be run through Menu-Assisted Resource Control (MARC).

Once LOGANALYZER is running, the user can find out how much of the log has been analyzed at a given moment by entering ?HI on a CANDE terminal or <mix number>HI on an ODT. The following is an example of the display that is returned when ?HI or ?2660HI (through CANDE) is entered:

```
#2660 DISPLAY:  LOG 60% READ.
```

When the LOG option is selected from MARC, a series of screens is presented that allow the user to select options for a LOGANALYZER report.

Option List

The <option list> allows the user to specify the input file to be used by LOGANALYZER, the destination of the LOGANALYZER output, the range of records to be searched, and the types of records to be searched for.

```
<option list>

──┬────────────────────┬─┬───────────────────────┬──────────────────────────→
  └─<device option>────┘ ├─ / ──<integer>────────┤
                         └─ " ──<logname>── " ───┘

→─┬─────────────────────────┬─┬───────────────────────────────┬─────────────→
  ├─<time>─┬──────────┬─────┤ └─ TO ─┬─<time>─┬──────────┬─────┤
  │        └─<date>───┘     │        │        └─<date>───┘     │
  └─<date>──────────────────┘        └─<date>──────────────────┘

→─┬────────────────────────┬─┬────────┬─────────────────────────────────────┤
  └─<loganalyzer options>──┘ └─ STOP ─┘

<device option>

──┬─ ODT ──────────────────────────────────┬────────────────────────────────┤
  ├─ CONSOLE ──────────────────────────────┤
  ├─ REMOTE ───────────────────────────────┤
  ├─ DCSTATION ────────────────────────────┤
  ├─ PRINTER ──────────────────────────────┤
  ├─ LINEPRINTER ──────────────────────────┤
  └─ TEXT ── " ──<file title>── " ─────────┘

<logname>

──┬───────────────┬─<name of log file>─┬─────────────────────────┬──────────┤
  ├─<usercode>────┤                    └─ ON ──<family name>─────┘
  └─ * ───────────┘
```

If the <option list> is empty, the effect is the same as if the user had specified LOG ALL. Refer to “Selection Options” in this section for information about the ALL option.

The following text describes the meaning of each variable:

<device option>

Controls the output destination of the LOGANALYZER output.

The PRINTER and LINEPRINTER options cause the output to be printed in lines of 132 characters.

The TEXT option causes the output to be written to the file with the specified <file title>. If a file of the same <file title> already exists, LOGANALYZER replaces it with a new file. By default, the file has a FILEKIND file attribute value of DATA and 132 character records. If LOGANALYZER is run with the SW1 task attribute set to TRUE, the file is created with FILEKIND value of TEXTDATA and 80 character records, which are sequentially numbered with an 8-digit number at the end of each record. If LOGANALYZER is discontinued before completing its run, the DATA or TEXTDATA output file preserves the output generated up to that point.

When LOGANALYZER is invoked from an ODT, the ODT and CONSOLE options cause output to be displayed on that ODT, assuming that the device has been set up with visible ETX characters. The REMOTE and DCSTATION options also cause output to be displayed on the originating ODT, assuming that the device has been set up with invisible ETX characters. The output is displayed in pages of 23 lines, with 80 characters per line.

When LOGANALYZER is invoked from a remote terminal, the REMOTE, DCSTATION, ODT, and CONSOLE options cause output to be displayed on that terminal. If the terminal is set up as a screen device, the output is consistent with the terminal screen size and width definitions. You can use the CANDE *TERM* command to alter the defines of the terminal screen size and width. For more information, see the *A Series CANDE Operations Reference Manual.* The default device option for remote terminals is REMOTE; otherwise PRINTER is assumed.

The VALUE task attribute can be used to change the default number of lines per page and characters per line. The lower three digits of the VALUE attribute represent the number of characters per line; the fourth and fifth digits represent the number of lines per page. The number of lines per page can be from 12 through 64; the number of characters per line can be from 40 through 256. The following examples show the use of the VALUE attribute:

- LOG REMOTE;VALUE=27132 sets the page size to 27 lines, with 132 characters per line.
- LOG PRINTER;VALUE=80 sets the number of characters per line to 80.
- LOG ODT;VALUE=170000 sets the page size to 17 lines.

continued

continued

/<integer>

The /<integer> notation (where <integer> is a 6-digit number) specifies that the log file to be analyzed has the title SUMLOG/<system serial number>/<current date>/<integer> and is located on the same family as the current system log. The log file title generated by this specification has the same format as the title created for the old system log file when the log is released by the TL (Transfer Log) system command. Refer to the *A Series System Commands Operations Reference Manual* for information on the TL command. For example, if the user is on a system with a serial number of 227, and the current date is August 22, 1983, then including /000005 in the option list would cause LOGANALYZER to use the file named SUMLOG/227/082283/000005 as its input file.

<logname>

The <logname> is a valid file title for a log file and must be entered in uppercase. If a usercode is specified, LOGANALYZER appends the usercode to the file name; otherwise, it assumes a nonusercode, *(star) file, is requested. LOGANALYZER assumes all log files to be on the family that was specified for LOG when using the DL (Disk Location) system command unless the ON <family name> part of the <logname> syntax is used. Using the ON <family name> syntax causes the program to look for the log file on the specified family. Refer to the *A Series System Commands Operations Reference Manual* for information on the DL command.

Note: *If a RUN command is used to initiate LOGANALYZER, and the <option list> is enclosed in quotation marks, the <logname> must be enclosed in a double set of quotation marks.*

<time>

A four-digit integer denoting time in an HHMM format or a six-digit integer denoting time in an HHMMSS format (H is hours; M is minutes; S is seconds). For example, 1:45 is represented as 1345 and 3:30:18 p.m. is represented as 153018. Leading zeros are required. For example, 9:15 a.m. must be represented as 0915, and 12:05:12 a.m. as 000512. If no start time is specified, 0000 is used. If no stop time is specified, 2400 is used.

<date>

The date for the LOGANALYZER report in the form mm/dd/yy, where mm, dd, and yy stand for month, day, and year, respectively. For example, June 1, 1983 is 06/01/83. If only one date is specified, it is used as both the start and stop date for the time range. If no date is specified, the following factors determine the default range:

- If a <time> is specified, or if the last entry in the log file has today's date, then today's date is used as both the start and stop date for the time range.
- Otherwise, the date range includes the full range of dates in the log file.

STOP

Signifies the end of input. All input records after STOP are ignored by LOGANALYZER. The use of STOP is optional; if STOP is not entered, LOGANALYZER inserts a STOP at the end of the input.

LOGANALYZER Options

There are two kinds of LOGANALYZER options. Selection options select log entries of a particular type to be included in the LOGANALYZER output. Output options affect the destination to which the LOGANALYZER output will be sent and the format of the report.

<loganalyzer options>

```
  ┌←───────────────────┐
──┴─┬─<selection options>─┬─┴──────────────────────────────────┤
    └─<output options>────┘
```

Most LOGANALYZER options are additive options. That is, if more than one option is specified, all entries that satisfy any of the specified options are listed. However, the ABORT, ERRORS, IDENTITY, JOB, MIX, SESSION, and TASK options are restrictive options. If any restrictive options are specified, then only entries that satisfy all of the specified restrictive options are listed. If a combination of restrictive and additive options is used, then only entries that satisfy all of the restrictive options and at least one of the additive options are listed.

Selection Options

Selection options choose particular types of log entries for inclusion in the LOGANALYZER output. The output illustrations for the following options appear in "Output Illustrations" later in this section.

<selection options>

```
──┬─ DATE ─────────────────────────────────────────────┬──────────────┤
  ├─ ALL ──────────────────────────────────────────────┤
  ├─ RAW ──────────────────────────────────────────────┤
  │ ┌←─────────────────────────────────────────────┐   │
  └─┴─┬─<configuration and maintenance options>─┬─┴───┘
      ├─<data comm options>─────────────────────┤
      ├─<diagnostics options>───────────────────┤
      ├─<job/task/session options>──────────────┤
      └─<additional selection options>──────────┘
```

The following text describes the meaning of each option:

DATE	Lists the date and time of the first and last log entries. This information is printed on all LOGANALYZER reports. The DATE option, if specified, must be the only option requested. The output of the DATE option is illustrated in Figure 7–1.
ALL	Lists the entire log. The output of the ALL option is illustrated in Figure 7–2.
RAW	Lists an unanalyzed hexadecimal dump of the entire log file. The output of the RAW option is illustrated in Figure 7–3.

Configuration and Maintenance Options

These options select log records related to the hardware and software configurations of the system and problems that are encountered during operation.

```
<configuration and maintenance options>

──┬─ CONFIG ──────────────────┬──────────────────────────────┤
  ├─ CPUERROR ────────────────┤
  ├─ FASUMMARY ───────────────┤
  ├─ HL ──────────────────────┤
  ├─ IOSUMMARY ───────────────┤
  ├─ MAINFRAME ───────────────┤
  ├─ IOERROR ─┬─┬─────────────┤
  └─ MAINT ───┘ └─<device option>─┘
```

The following text describes the meaning of each option:

CONFIG

Lists all hardware and software configuration log records. The configuration is logged at each halt/load occurrence and again each time the log is transferred or initiated. The output of the CONFIG option is illustrated in Figure 7–4. Figure 7–5 shows an example of the CONFIG output for a memory configuration record on an A 10 system.

CPUERROR

This option is synonymous with MAINFRAME.

FASUMMARY

Lists a table of failure counts against Field Replaceable Units (FRUs) for the A 17 resource management module (RMM). The table resembles the output from the FAS (Failure Analysis Summary) system command. The FASUMMARY option is available only when analyzing A 17 logs. If the FASUMMARY option is used when analyzing a non-A 17 log, LOGANALYZER displays the following message:

```
FASUMMARY ONLY ALLOWED ON A17 SUMLOGS
```

The FASUMMARY is implicitly set if the MAINFRAME option is used when analyzing an A 17 log file.

The output of the FASUMMARY option is illustrated in Figure 7–6.

HL

Lists the system halt/load history. The output of the HL option is illustrated in Figure 7–7. On A 1 through A 6 systems, automatic power log entries are also analyzed. Examples of automatic power log entries appear at the end of this topic.

IOSUMMARY

Lists the table that summarizes IOERROR log entries. Individual errors are not printed. The summary is the same as that produced at the end of a MAINT or IOERROR request. The output of the IOSUMMARY option is illustrated in Figure 7–8.

MAINFRAME

Lists all mainframe log records such as box failures, system error actions, halt/load, and dump records. The output of the MAINFRAME option is illustrated in Figure 7–9.

If the log file being analyzed was produced on an A 17 system, the MAINFRAME option implicitly causes the FASUMMARY option to be set.

continued

continued

IOERROR — Lists all peripheral error entries. If IOERROR is followed by a <device option>, only the peripheral error entries for the selected device type and unit number are listed. The output of the IOERROR option is illustrated in Figures 7–10 and 7–11.

MAINT — Lists all mainframe errors, peripheral errors, and hardware configuration log records. If MAINT is followed by a <device option>, the hardware configuration log records are not included and only the mainframe and peripheral errors for the selected device type and unit number are listed. The output of the MAINT option is illustrated in Figures 7–12 and 7–13. This command also analyzes automatic power log entries generated on A 1 through A 6 systems. Examples of automatic power log entries appear at the end of this topic.

Figure 7–14 shows an example of the single–bit memory error report produced by the MAINT CPU command on an A 10 system.

<device option> — Can be specified after the IOERROR or MAINT option. Causes LOGANALYZER to select only those records that satisfy the given IOERROR or MAINT option and also apply to the selected device type or device and unit number. The output of the <device option> is illustrated in Figure 7–15.

The following diagram lists the many device options available.

```
<device option>

   ┌←──────────────────────────────────────────┐
───┴─┬─ DUMPEXTRD ─────────────────────────────┴────────────────────────────┤
     ├─ CD ──────────────┬───────────────────┤
     ├─ CP ──────────────┤  └─<unit number>──┘
     ├─ CPU ─────────────┤
     ├─ CR ──────────────┤
     ├─ DC ──────────────┤
     ├─ DECODEEXTRD ─────┤
     ├─ DK ──────────────┤
     ├─ HC ──────────────┤
     ├─ HY ──────────────┤
     ├─ IP ──────────────┤
     ├─ LP ──────────────┤
     ├─ MT ──────────────┤
     ├─ NRZ ─────────────┤
     ├─ NRZMT ───────────┤
     ├─ NRZ7 ────────────┤
     ├─ NRZ7MT ──────────┤
     ├─ NRZ9 ────────────┤
     ├─ NRZ9MT ──────────┤
     ├─ PE ──────────────┤
     ├─ PEMT ────────────┤
     ├─ PK ──────────────┤
     ├─ PP ──────────────┤
     ├─ SC ──────────────┤
     └─ VC ──────────────┘
```

The device mnemonic definitions are as follows:

Mnemonic	Definition
CD	CD-ROM unit
CP	Card punch unit
CPU	Central processing unit
CR	Card reader unit
DC	EMS data communications unit
DK	Memory Disk unit
HC	Host Control (HC) unit
HY	HYPERchannel® unit
IP	Image printer
LP	Line printer unit
MT	Magnetic tape unit
NRZ, NRZMT	Nonreturn-to-zero (NRZ) magnetic tape unit
NRZ7, NRZ7MT	Nonreturn-to-zero (NRZ) 7-track magnetic tape unit
NRZ9, NRZ9MT	Nonreturn-to-zero (NRZ) 9-track magnetic tape unit
PE, PEMT	Phase-Encoded (PE) magnetic tape unit
PK	Disk unit
SC	Operator display terminal
VC	Voice channel

The DUMPEXTRD and DECODEEXTRD options affect the analysis of log entries for those device types that have extended status information or request sense information (for example, native SCSI devices). For all such devices except HYPERchannel,® DUMPEXTRD displays a hexadecimal dump of the extended status information. DECODEEXTRD displays a formal analysis of the same information.

Expanded analysis for HYPERchannel devices is produced only if either DUMPEXTRD or DECODEEXTRD is used in conjunction with the MAINT HY option. Requesting the option DUMPEXTRD in conjunction with a MAINT HY request causes the message proper and sense bytes to be presented in raw format without description. For example, the "LOG MAINT HY DUMPEXTRD" request produces the following output in addition to the normal "LOG MAINT HY" output:

```
MESSAGE PROPER           :  <first 16 bytes of the message proper>

ADAPTER SENSE BYTES DATA : <adapter sense byte information>
```

HYPERchannel is a registered trademark of Network Systems Corporation.

If the option DECODEEXTRD is requested in conjunction with a MAINT HY request, the message proper and sense bytes are given expanded analysis.

Examples of Automatic Power Log Entries for A 1 through A 16 Systems

The following is an example of a scheduled automatic power-off entry:

```
SYSTEM 1ØØ1 AUTOMATIC POWER REQUEST    FEB Ø1, 1987 17:2Ø:16    REASON:
SCHEDULED
```

The following is an example of an unscheduled automatic power-off entry:

```
SYSTEM 1ØØ1 AUTOMATIC POWER REQUEST    FEB 28, 1987 Ø9:12:16    REASON:
UNSCHEDULED
```

The following is an example of a thermal overload warning entry:

```
SYSTEM 1ØØ1 AUTOMATIC POWER REQUEST    MAR 15, 1987 Ø4:48:59    REASON:
THERMAL WARNING
```

The following is an example of a thermal overload power-off entry:

```
SYSTEM 1ØØ1 AUTOMATIC POWER REQUEST    MAR 15, 1987 Ø4:55:37    REASON:
THERMAL OVERLOAD
```

The following is an example of a cancelled pending power-off operation entry:

```
SYSTEM 1ØØ1 AUTOMATIC POWER REQUEST    FEB Ø1, 1987 17:3Ø:27    REASON:
CANCELLED
```

The following is an example of an actual power-off entry:

```
SYSTEM 1ØØ1 AUTOMATIC POWER REQUEST    MAY Ø7, 1987 19:28:31    REASON:
ACTUAL POWEROFF
```

The following is an example of an actual blower failure entry:

```
SYSTEM 1ØØ1 AUTOMATIC POWER REQUEST    JUL 25, 1987 19:28:31    REASON:
BLOWER FAILURE
```

Data Comm Options

The following options select log records that are related to data communications:

```
<data comm options>

──┬─ BNA ──────────────────────────────────────────────────┤
  ├─ BNAV2 ─────────────┤
  ├─ COMS ──┬───────────┤
  │         ├─ CONFIG ──┤
  │         └─ WINDOW ──┤
  ├─ HLCN ──────────────┤
  ├─ IDC ───────────────┤
  ├─ MCS ───────────────┤
  ├─ NMS ──┬────────────┤
  │        ├─ LCF ──────┤
  │        ├─ NCF ──────┤
  │        ├─ SNMP ─────┤
  │        └─ OTHER ────┤
  ├─ NSP ───────────────┤
  ├─ SNA ──┬────────────┤
  │        ├─ REPORT ───┤
  │        ├─ TRACE ────┤
  │        ├─ ERROR ────┤
  │        ├─ INQUIRY ──┤
  │        ├─ COMMAND ──┤
  │        └─ OTHER ────┤
  └─ TCPIP ──┬──────────┤
             ├─ TCP ────┤
             ├─ IP ─────┤
             ├─ ARP ────┤
             ├─ RIP ────┤
             ├─ UDP ────┤
             ├─ ICMP ───┤
             ├─ TCPMGR ─┤
             ├─ PIM ────┤
             ├─ SNMP ───┤
             └─ OTHER ──┘
```

The following text describes the meaning of each option:

BNA	Lists all BNA Version 1 log records. The output of the BNA Version 1 option is illustrated in Figure 7–16.
BNAV2	Lists all BNA Version 2 log records. Analysis of BNA Version 2 records requires that the BNA translation library, BNAV2TRANSLATION, be specified using the SL (Support Library) system command. The output of the BNA Version 2 option is illustrated in Figure 7–17.
COMS	Lists the records reporting COMS events. If COMS is followed by CONFIG, only the COMS configuration change records are selected. If COMS is followed by WINDOW, only the COMS window activity records are selected.
HLCN	Lists the Host LAN Connection (HLCN) records.
IDC	Lists the IDC and ID records.
MCS	Lists all MCS log records. The output of the MCS option is illustrated in Figure 7–18.

continued

continued

NMS

Lists all nondiagnostic commands, responses, and reports generated for Network Management.

When NMS is followed by a subordinate option, LOGANALYZER selects only the entries related to that option. The following list defines the subordinate options:

- LCF. Lists the Local Control Facility entries.
- NCF. Lists the Network Control Facility entries.
- SNMP. Lists entries from the SNMP agent.
- OTHER. Lists the miscellaneous entries.

NSP

Lists all data comm log records. The output of the NSP option is illustrated in Figure 7–19.

SNA

Lists all nondiagnostic commands, responses, and reports generated for Systems Network Architecture (SNA).

When SNA is followed by a subordinate option, LOGANALYZER selects only the entries related to that option. The following list defines the subordinate option:

- REPORT. Lists noteworthy network events.
- TRACE. Lists the network frame reported as a result of the TRACE + command.
- ERROR. Lists unexpected fault conditions.
- INQUIRY. Lists the solicited status requests and responses.
- COMMAND. Lists the solicited action requests and responses.
- OTHER. Lists the miscellaneous events.

continued

continued

TCPIP

Lists all nondiagnostic commands, responses, and reports generated by the TCP/IP network provider.

When TCPIP is followed by a subordinate option, LOGANALYZER selects only the entries related to that option. The following list defines the subordinate options:

- TCP. Lists the Transmission Control Protocol layer entries.
- IP. Lists the Internet Protocol layer entries.
- ARP. Lists the Address Resolution Protocol function entries.
- RIP. Lists the Route Information Protocol function entries.
- UDP. Lists the User Datagram Protocol layer entries.
- ICMP. Lists the Internet Control Message Protocol function entries.
- TCPMGR. Lists the TCP Manager function entries.
- PIM. Lists entries for the connection open and close reports.
- SNMP. Lists entries generated during the process of an SNMP command. SNMP commands are logged under Major Type 28, Network Management.
- OTHER. Lists the miscellaneous entries.

Diagnostics Options

These options select log records created by diagnostics options of various system software products.

```
<diagnostics options>

— DIAGNOSTICS ————————————————————————————————————————————→

→┬──────────────────────────────────────────────────────────┤
 └ ( ┬─ ALL ───────────────────────────────────────────┬ ) ┘
     │┌←──────────────────────────────────────────────┐│
     └┴┬/1\- DIR ──┬─────────────────────────────────┬┘
       │           └ ( ┬─ ALL ────────────┬ ) ────────┤
       │               │┌←──────────────┐│            │
       │               └┴<DIR subtype>──┘┘            │
       ├/1\- DSS ──┬─────────────────────────────────┤
       │           └ ( ┬─ ALL ────────────┬ ) ────────┤
       │               │┌←──────────────┐│            │
       │               └┴<DSS subtype>──┘┘            │
       ├/1\- HLCN ─┬─────────────────────────────────┤
       │           └ ( ┬─ ALL ────────────┬ ) ────────┤
       │               │┌←───────────────┐│           │
       │               └┴<HLCN subtype>──┘┘           │
       ├/1\- MHS ──┬─────────────────────────────────┤
       │           └ ( ┬─ ALL ────────────┬ ) ────────┤
       │               │┌←──────────────┐│            │
       │               └┴<MHS subtype>──┘┘            │
       ├/1\- NIS ──┬─────────────────────────────────┤
       │           └ ( ┬─ ALL ────────────┬ ) ────────┤
       │               │┌←──────────────┐│            │
       │               └┴<NIS subtype>──┘┘            │
       ├/1\- NMS ──┬─────────────────────────────────┤
       │           └ ( ┬─ ALL ────────────┬ ) ────────┤
       │               │┌←──────────────┐│            │
       │               └┴<NMS subtype>──┘┘            │
       ├/1\- OSI ──┬─────────────────────────────────┤
       │           └ ( ┬─ ALL ────────────┬ ) ────────┤
       │               │┌←──────────────┐│            │
       │               └┴<OSI subtype>──┘┘            │
       ├/1\- SNA ──┬─────────────────────────────────┤
       │           └ ( — ALL — ) ─────────────────────┤
       └/1\- TCPIP ┬─────────────────────────────────┤
                   └ ( ┬─ ALL ──────────────┬ ) ──────┘
                       │┌←────────────────┐│
                       └┴<TCPIP subtype>──┘┘
```

The DIAGNOSTICS options each correspond to minor types and subtypes of the Major Type 19 (Diagnostics) log entry. Note that the SNA minor type has no subtypes. For a description of this log entry, refer to the "SUMLOG" section of this manual. The following are the minor types corresponding to each DIAGNOSTICS option:

DIAGNOSTICS Option	Minor Type Number	Minor Type Name
DIR	12	OSI directory
DSS	1	Distributed systems service
HLCN	11	Host LAN Connection
MHS	4	Message Handling System
NIS	7	Network-independent software
NMS	13	Network Management System
OSI	5	Open Systems Interconnection
SNA	16	Systems Network Architecture
TCPIP	6	TCP/IP

The subtype options, such as the <DSS subtypes>, limit the search to specific subsets of the diagnostic records logged for each minor type. For a list of the possible subtypes for each minor type, refer to the discussion of the Diagnostics Record (Major Type 19) in the "SUMLOG" section of this manual. In that discussion, each subtype is identified by a JOBFORMATTER define name. The last part of each JOBFORMATTER define is the same as the corresponding LOGANALYZER subtype. For example, the SUMLOG discussion lists a define called LOGDIAG_DSS_FTAM under the DSS minor type. To request LOGANALYZER to report on this subtype, you would use the option DIAGNOSTICS DSS FTAM.

Table 7–1 shows examples of the ways DIAGNOSTICS options can be combined for various effects.

Table 7–1. DIAGNOSTICS Examples

Command	Result
DIAGNOSTICS	Retrieves all diagnostics log records.
DIAGNOSTICS (DSS)	Retrieves all diagnostics log records for distributed systems services (DSSs).
DIAGNOSTICS (DSS MHS)	Retrieves all diagnostics log records for distributed systems services (DSSs) and the Message Handling System (MHS).
DIAGNOSTICS (NIS (PARSE ROUTER))	Retrieves all diagnostic log records for the network-independent software (NIS) subtypes PARSE and ROUTER.
DIAGNOSTICS (NIS (PARSE ROUTER) MHS (AS MTA))	Retrieves all diagnostic log records for the network-independent software (NIS) subtypes PARSE and ROUTER and the Message Handling System (MHS) subtypes AS and MTA.

For examples of output from the DIAGNOSTICS option, refer to Figures 7–20, 7–21, and 7–22.

Job, Task, and Session Options

The following options select log records for a job, task, or session activity. Sample output for the job, task, and session options are shown in "Output Illustrations" later in this section.

```
<job/task/session options>

──┬─ ABORT ──────────────────────────────────────────────┬────────────┤
  ├─ BOJ ────────────────────────────────────────────────┤
  ├─ BOT ────────────────────────────────────────────────┤
  ├─ DMS ────────────────────────────────────────────────┤
  ├─ EOJ ────────────────────────────────────────────────┤
  ├─ EOT ────────────────────────────────────────────────┤
  ├─ ERRORS ─────────────────────────────────────────────┤
  ├─ FILE ───────────────────────────────────────────────┤
  ├─ IO ─────────────────────────────────────────────────┤
  ├─ LIB ────────────────────────────────────────────────┤
  ├─ PRINTS ─┬───────────────────────────────────────────┤
  │          ├────────────┬─<integer>─┬──────────────────┤
  │          ├─ REQUEST ──┘           └─ THRU ─<integer>─┤
  │          └─ JOB ─<job number>─┬──────────────────────┤
  │                               └─ THRU ─<job number>──┤
  ├─/1\─┬─ JOB ─┬─ " ─<name>─┬──────┬─ " ────────────────┤
  │     │       │            └─ /= ─┘                    │
  │     │       └─<mix number>─┬─────────────────────────┤
  │     │                      └─ THRU ─<mix number>─────┤
  │     └─ SESSION ─<mix number>─┬───────────────────────┤
  │                              └─ THRU ─<mix number>───┤
  └─/1\─┬─ IDENTITY ─┬───────────────────────────────────┤
        │            └─<mix number>─┬────────────────────┤
        │                           └─ THRU ─<mix number>┤
        ├─ MIX ─<mix number>─┬───────────────────────────┤
        │                    └─ THRU ─<mix number>───────┤
        └─ TASK ─┬─ " ─<name>─┬──────┬─ " ───────────────┤
                 │            └─ /= ─┘                   │
                 └─<mix number>─┬────────────────────────┤
                                └─ THRU ─<mix number>────┘
```

The following text describes the meaning of each option:

ABORT	Lists the records associated with tasks that terminated abnormally with a DS or QT. The ABORT option cannot be used simultaneously with the APPEND, CREATE, or UNSORTED option. The output of the ABORT option is illustrated in Figure 7–23.
BOJ	Lists the beginning of job records. The output of the BOJ option is illustrated in Figure 7–24.
BOT	Lists the beginning of job and beginning of task records. The output of the BOT option is illustrated in Figure 7–25.
DMS	Lists the database OPEN, CLOSE, FREEZE, and RESUME records. The output of the DMS option is illustrated in Figure 7–26.
EOJ	Lists the end of job records. The output of the EOJ option is illustrated in Figure 7–27.
EOT	Lists the end of job and end of task records. The output of the EOT option is illustrated in Figure 7–28.
ERRORS	Lists the records associated with compiler tasks that terminated with syntax errors. The ERRORS option cannot be used simultaneously with the APPEND, CREATE, or UNSORTED option. The output of the ERRORS option is illustrated in Figure 7–29.
FILE	Lists the following records: file OPEN, CLOSE, and INTERVAL; BNA port activity; and file status entries. The output of the FILE option is illustrated in Figure 7–30.
IO	Lists the file OPEN, CLOSE, INTERVAL, and BNA port activity records.

continued

continued

IDENTITY

Lists identity records. Identity records establish, change, or end the identity for a mix number. The following are identity records:

- BOJ = Beginning of job
- EOJ = End of job
- BOT = Beginning of task
- EOT = End of task
- LOGON = MCS session log-on
- LOGOFF = MCS session log-off
- EI = Establish identity

IDENTITY <mix number> lists the identity records associated with the specified mix number.

IDENTITY <mix number> THRU <mix number> lists the identity records for all mix numbers that fall within the specified range.

The IDENTITY option cannot be used simultaneously with the MIX or TASK options.

continued

continued

JOB

Lists the records associated with the specified job and its tasks.

JOB "<name>" lists the records for the job whose BEGIN JOB statement contains the specified <name>. The <name> parameter uses the same format as a file title.

JOB "<name>/=" lists all jobs whose high-level identifier matches the specified <name>. The comparison is like a file title tested for membership in a file directory. For example, JOB (ME)TOO/= selects jobs named (ME)TOO/A or (ME)TOO/B, but not (ME)TOO.

A name parameter can consist of a "<string>" construct. JOB ""<string>"" lists the records associated with the job whose job name matches the specified string. For example, the following command lists the records for the BEGIN JOB "MY JOB" statement:

```
LOG JOB ""MY JOB""
```

In general, you should classify jobs with names that have embedded blanks as strings rather than as file names. This rule applies to a number of system commands (or DCKEYIN) commands. For example, to list the log records of a RB ON PACK system command, use the following command:

```
LOG JOB ""RB ON MYPACK""
```

Note: *If a RUN command is used to initiate LOGANALYZER, and the <option list> is enclosed in quotation marks, then the <name> or <name>/= construct must be enclosed in a double set of quotation marks.*

JOB <mix number> lists the records for jobs with the specified mix number. JOB <mix number> THRU <mix number> lists the records for all jobs whose mix numbers fall within the specified range. The listed records include log entries created by the print system when processing print requests for the specified mix number or numbers.

The JOB option cannot be used simultaneously with the SESSION option. The output of the JOB option is illustrated in Figure 7–31.

LIB

Lists the library LINK, DELINK, FREEZE, and RESUME records. The output of the LIB option is illustrated in Figure 7–32.

MIX

MIX <mix number> lists the records associated with the specified mix number. MIX <mix number> THRU <mix number> lists the records associated with the specified range of mix numbers. A <mix number> of zero (0) lists all log records that have no associated mix numbers (for example, configuration records).

The MIX option cannot be used simultaneously with the IDENTITY or TASK options. The output of the MIX option is illustrated in Figure 7–33.

continued

continued

PRINTS

Lists logs entries that are created by the print system.

PRINTS <integer> lists the printing entries that are logged for the specified print request number. The REQ option is the default option. The <integer> THRU <integer> option lists the log entries for all print requests whose request numbers fall within the specified range.

PRINTS JOB <job number> lists the printing entries that are logged for the specified job number. The JOB <job number> THRU <job number> option lists the log entries for all jobs whose job numbers fall within the specified range.

SESSION

SESSION <mix number> lists the records associated with the specified CANDE or other MCS mix number. SESSION <mix number> THRU <mix number> lists the records associated with the CANDE or other MCS mix numbers within the specified range. The SESSION option cannot be used simultaneously with the JOB option. The output of the SESSION option is illustrated in Figure 7–34.

TASK

TASK "<name>" lists the records associated with the specified task name. The <name> parameter uses the same format as a file title.

TASK "<name>/=" lists all jobs whose high-level identifier matches the specified <name>. The comparison is like a file title tested for membership in a file directory. For example, TASK "(ME)TOO/=" lists tasks named (ME)TOO/A or (ME)TOO/B, but not (ME)TOO.

Note: *If a RUN command is used to initiate LOGANALYZER, and the <option list> is enclosed in quotation marks, then the <name> or <name>/= construct must be enclosed in a double set of quotation marks.*

TASK <mix number> lists the records for tasks with the specified mix number. TASK <mix number> THRU <mix number> lists the records for all the tasks whose mix numbers fall within the specified range. A <mix number> of zero (0) lists all log records that have no associated mix numbers (for example, configuration records).

The TASK option cannot be used simultaneously with the IDENTITY or MIX options. The output of the TASK option is illustrated in Figure 7–35.

The job, task, and session options use the following logic to select log records for listing:

- The ABORT, ERRORS, IDENTITY, JOB, MIX, PRINTS, SESSION, and TASK options are restrictive options. If more than one is specified, any log record that contains all the specified restrictive options is selected, and any log record that contains less than all the specified restrictive options is not selected.
- The BOJ, BOT, DMS, EOJ, EOT, FILE, LIB, and IO options are additive options. If one or more is specified, any log record that contains any of the specified additive options is selected.
- If no options are specified, all records are selected.
- If one or more restrictive options are specified, and no additive options are specified, any additive-type record related to the specified restrictive option is selected.
- If no restrictive options are specified, and one or more additive options are specified, any record containing any of the specified options is selected.
- If a combination of restrictive and additive options are specified, any record that contains all the specified restrictive options and any of the specified additive options is selected.

Additional Selection Options

The following options list various types of log entries. The output illustrations for these options appear in "Output Illustrations" later in this section.

<additional selection options>

```
──┬─ ACCESSCODE ─<code identifier>──────┬──────────────────────────────┤
  ├─ CHARGECODE ─<code identifier>──────┤
  ├─ COMMENT ───────────────────────────┤
  ├─ DEIMPLEMENTATION ──────────────────┤
  ├─ DRC ───────────────────────────────┤
  ├─ FIND ─ " ─<wild card string>─ " ───┤
  ├─ MSG ───────────────────────────────┤
  ├─ OPERATOR ─┬────────────────────────┤
  │            │ ┌←─────┐               │
  │            └─┴<type>┴───────────────┤
  │           ┌←─────────────┐          │
  ├─ RESULT ──┴┬─ SUCCESS ───┤──────────┤
  │            ├─ FAILURE ───┤          │
  │            ├─ RELEVANT ──┤          │
  │            └─ VIOLATION ─┘          │
  ├─ SECURITY ──────────────────────────┤
  ├─ UNKNOWN ───────────────────────────┤
  ├─ USERCODE ─<code identifier>────────┤
  ├─ USERDATA ──────────────────────────┤
  └─ VOLUMES ───────────────────────────┘
```

<code identifier>

```
──┬─<simple name>─┬──────────────────────────────────────┤
  ├─ . ───────────┤
  ├─ * ───────────┤
  └─ = ───────────┘
```

The following text describes the meaning of each option:

ACCESSCODE

Lists entries associated with the specified <code identifier>. The ACCESSCODE option is valid only on systems with the InfoGuard security enhancement software installed.

To list entries with no accesscode specification, use an asterisk (*) as the code identifier. To list entries with the accesscode of the LOGANALYZER process, specify an equals sign (=) as the code identifier. The accesscode is not interrogated for entry selection if a period (.) is specified as the code identifier. If ACCESSCODE is specified in combination with CHARGECODE, RESULT, and/or USERCODE, only the records that contain all the specified options are selected. If ACCESSCODE is used together with ALL, then all entries with the specified accesscode and all public records are selected. Public records include all maintenance and halt/load records.

CHARGECODE

Lists entries associated with the specified <code identifier>. The CHARGECODE option is valid only on systems with the InfoGuard security enhancement software installed.

To list entries with no chargecode specification, use an asterisk (*) as the code identifier. To list entries with the chargecode of the LOGANALYZER process, specify an equals sign (=) as the code identifier. The chargecode is not interrogated for entry selection if a period (.) is specified as the code identifier. If CHARGECODE is specified in combination with ACCESSCODE, RESULT, and/or USERCODE, only the records that contain all the specified options are selected. If CHARGECODE is used together with ALL, then all entries with the specified chargecode and all public records are selected. Public records include all maintenance and halt/load records.

COMMENT

Lists all comments entered into SUMLOG by means of the LC (Log Comment) and LJ (Log to Job) system commands. Refer to the System Commands Reference Manual for information on the LC and LJ commands. The output of the COMMENT option is illustrated in Figure 7–36.

DEIMPLEMENTATION

Lists the deimplementation warnings generated by the operating system with the associated mix number and task identification. The output of the DEIMPLEMENTATION option is illustrated in Figure 7–37.

DRC

Causes all disk resource control system entries to be listed. These entries include USERDATA record overflow, userdata error, and invalid family name in the familylist entry.

continued

continued

FIND

Limits the output to those log entries that contain text matching the specified <wild card string>. The <wild card string> can be any string of characters. FIND causes a case-sensitive, partial-word search (that is, the target text is found even if embedded in a larger word).

The <wild card string> can include two types of wild card characters: question marks (?) and equal signs (=). A question mark in the <wild card string> matches any single character. An equal sign in the <wild card string> matches any sequence of characters.

Note: *The <wild card string> normally should begin and end with equal signs, even if the <wild card string> text consists of complete words. For example, to find the phrase INVALID DESTINATION, the FIND option should specify* FIND "=INVALID DESTINATION=".

If multiple FIND options are specified, only the last one has effect. For example, the command LOG FIND "=HI=" FIND "=BYE=" *finds only entries that contain the string* BYE.

MSG

Lists all system messages and the mix numbers associated with these messages. The list includes operator entries as listed by the OPERATOR option. The output of the MSG option is illustrated in Figure 7–38.

OPERATOR

Lists all ODT entries. When OPERATOR is used with one or more <type> specifications, only log entries for the corresponding system commands are listed. Each <type> is a system command mnemonic, such as DS, MP, OK or PG. Most system commands that change the system state are allowed as <type>s. PS is a valid <type> and lists log entries that are associated only with print system commands that have caused a change in the print system. System commands that are inquiries but do not change the system state cannot be used as <type>s. Refer to the System Commands Reference Manual for all the system commands and their mnemonics. The output of the OPERATOR option is illustrated in Figure 7–39.

RESULT

Reports are created based on the results of the actions logged. These results can be any combination of successful actions, failed actions, security-relevant actions, and security violations. The results to be reported are specified by entering any combination of SUCCESS, FAILURE, RELEVANT, and VIOLATION.

continued

continued

Result reports for RELEVANT include actions on the USERDATAFILE, use of the system ??SECAD primitive and SECOPT commands, and attachments of guard files to files.

If RESULT is specified with ACCESSCODE, CHARGECODE, and/or USERCODE, only the records that contain all the specified options are selected.

The RESULT option is valid only on systems with the InfoGuard security enhancement software installed.

SECURITY

Lists security violation records such as INVALID USERCODE. If the logfile is private, the user must be privileged to display security violation entries. The output of the SECURITY option is illustrated in Figure 7–40.

UNKNOWN

Lists a hexadecimal dump of all SUMLOG records with major and minor types that LOGANALYZER does not recognize.

USERCODE

Lists entries associated with the specified <code identifier>. The USERCODE option is valid only on systems with the InfoGuard security enhancement software installed.

To list entries with no USERCODE specification, use an asterisk (*) as the <code identifier>. To list entries with the USERCODE of the LOGANALYZER process, specify an equals sign (=) as the <code identifier>. The USERCODE is not interrogated for entry selection if a period (.) is specified as the <code identifier>. If USERCODE is specified in combination with CHARGECODE, RESULT, and/or ACCESSCODE, only the records that contain all the specified options are selected. If USERCODE is used together with ALL, then all entries with the specified USERCODE and all public records are selected. Public records include all maintenance and halt/load records.

If LOGANALYZER has no direct read access to the analyzed SUMLOG file, the only valid <code identifier> is the USERCODE of the LOGANALYZER process; otherwise, any value is valid. For a discussion of direct read access, refer to the "Understanding Log Access Security" subsection of Section 12, "SUMLOG."

Note: *On InfoGuard systems, if the USERCODE option is not specified in the LOGANALYZER run, LOGANALYZER behaves as if USERCODE = were set. This is true even if LOGANALYZER is running with direct read access to the SUMLOG file. If LOGANALYZER has direct read access, then specifying a period as the <code identifier> disables usercode filtering.*

continued

continued

USERDATA — Lists USERDATA records. USERDATA records are records that describe the installation of a *SYSTEM/USERDATAFILE or changes to that file.

VOLUMES — Analyzes all the disk and tape volume status records generated by the system when a disk or tape volume comes online or goes offline, when a tape volume is purged by either the PG or SN system command, or when the first file on a tape volume is overwritten by a new file. The information displayed by LOGANALYZER includes the time, volume serial number, volume name, and volume status.

Output Options

The following options affect the destination of the LOGANALYZER output or the form in which the output is presented. The destination of the LOGANALYZER output can also be affected by the CON, PRINTER, and REMOTE options in "Option List" earlier in this section. The format of the LOGANALYZER output can also be affected by the RAW option in "Selection Options" earlier in this section. The output illustrations for these options appear in the subsection titled "Output Illustrations" later in this section.

```
<output options>

──┬─ APPEND ─┬─ " ─<file title>─ " ─┬──────────────────────────────┤
  ├─ CREATE ─┘                      │
  ├─ DUMP ──────────────────────────┤
  ├─<sort option>───────────────────┤
  └─ SORTSIZE ─ = ─<integer>────────┘

<sort option>

──┬─ SORTED ───┬───────────────────────────────────────────────────┤
  └─ UNSORTED ─┘
```

The following text describes the meaning of each option:

APPEND "<file title>" — Places the log records selected for analysis at the end of an existing file with the title <file title>, instead of listing them. The records are added in the same order as they appear in the log file being analyzed. The resulting file has the same format as the system log file and can in turn be analyzed by LOGANALYZER. The APPEND option cannot be used simultaneously with the ABORT, ERRORS, or SORTED option because APPEND unconditionally enables the UNSORTED option.

Note: *If a RUN command is used to initiate LOGANALYZER, and the <option list> is enclosed in quotation marks, then the <file title> must be enclosed in a double set of quotation marks.*

continued

continued

CREATE "<file title>"

Places the log records selected for analysis in a new file with the title <file title>, instead of listing them. If a file with the specified <file title> already exists, an error is reported and LOGANALYZER terminates. The records are placed in the new file in the same order as they appear in the log file being analyzed. The resultant file has the same format as the system log file and can in turn be analyzed by LOGANALYZER. The CREATE option cannot be used simultaneously with the ABORT, ERRORS, or SORTED option because CREATE unconditionally enables the UNSORTED option.

Note: *If a RUN command is used to initiate LOGANALYZER, and the <option list> is enclosed in quotation marks, then the <file title> must be enclosed in a double set of quotation marks.*

DUMP

Dumps in hexadecimal form each specified log record selected by other <loganalyzer options> before listing the record in its analyzed form. The output of the DUMP option is illustrated in Figure 7–41.

<sort option>

If SORTED is specified, then LOGANALYZER lists the log entries sorted into order first by mix number and then by type. If UNSORTED is specified, then LOGANALYZER lists the log entries in chronological order. The output from the UNSORTED option is illustrated in Figure 7–42. The output from the SORTED option is illustrated in Figure 7–43.

The SORTED and UNSORTED values cannot be used simultaneously. If neither SORTED nor UNSORTED is specified, the <sort option> defaults to UNSORTED in most cases. However, the following options can affect the <sort option>:

- The ABORT and ERRORS options implicitly set the <sort option> to SORTED. This value cannot be overridden.
- The APPEND and CREATE options implicitly set the <sort option> to UNSORTED. This value cannot be overridden.
- The MAINT option implicitly changes the <sort option> default to SORTED. This value can be overridden by an explicit assignment of UNSORTED.

SORTSIZE

Overrides the core size default value (4000) for the SORT.

Examples

Table 7–2 shows examples of LOGANALYZER commands.

Table 7–2. LOGANALYZER Examples

Command	Result
LOG and LOG ALL	Retrieves all entries.
LOG "SUMLOG/281/091782/000207"	Retrieves the same information as LOG, except the information is from the specified SUMLOG file.
LOG 5/11/89	Retrieves all entries for the date 5/11/89.
LOG 1200 TO 1700 MIX 345	Retrieves all entries for mix number 345 from 1200 to 1700.
LOG FILE and LOG IO	Retrieves all file OPEN, CLOSE, and INTERVAL entries.
LOG JOB "MYJOB"	Retrieves all entries for jobs named MYJOB and their tasks.
RUN *SYSTEM/LOGANALYZER("JOB ""MYJOB""");	Retrieves the same entries as LOG JOB "MYJOB". This example shows the use of a double set of quotes for embedded strings in a RUN command.
LOG NSP	Retrieves all data comm error records.
LOG MAINT DC	Retrieves all data comm I/O error entries.
LOG MAINT DK	Retrieves all maintenance log entries for memory disk units.
LOG MAINT 99	Retrieves all maintenance log entries for a peripheral unit with a unit number of 99.
LOG MAINT MT 97 60	Retrieves all maintenance log entries for magnetic tape units 97 and 60.
LOG MAINT PK IOSUMMARY	Gives a table summarizing all IOERROR records for all disks.
LOG MAINT PK DUMPEXTRD	Retrieves all mainframe errors and peripheral errors for disk devices, and includes the raw extended result descriptor for each entry.
LOG 0900 TO 1000 BNA APPEND "SUMLOG/BNA/ENTRIES"	Retrieves all BNA version 1 entries dated between 9:00 a.m. and 10:00 a.m. of the current day, and inserts them at the end of the already existing file SUMLOG/BNA/ENTRIES.
LOG MAINFRAME UNSORTED	Retrieves all mainframe log records and reports them in chronological order rather than sorting them first by type and then by job number.
LOG JOB 1260 THRU 1380 MAINT	Retrieves all mainframe error, peripheral error, and hardware configuration log records for jobs numbered within the range 1260 to 1380.
LOG PRINTER IOSUMMARY	Lists the table that summarizes IOERROR log entries and sends the report to a printer.

continued

Table 7–2. LOGANALYZER Examples (cont.)

Command	Result
LOG OPERATOR DS FREE ACQUIRE	Retrieves all entries which record the use of the system commands DS, FREE, or ACQUIRE.
LOG ERRORS BOT EOT MSG	Retrieves all beginning of task, end of task, and system message entries that are associated with tasks that terminated with syntax errors.
LOG BOJ EOJ BOT EOT MSG FIND "=(XYZ)="	Retrieves all beginning of job, end of job, beginning of task, end of task, and system message entries that contain the string *(XYZ)*.
LOG MSG FIND "=RECOVERY=REMOVED="	Retrieves all system message entries that contain both the words RECOVERY and REMOVED. The entries are retrieved only if the word RECOVERY precedes REMOVED. However, the two words can be separated by any amount of text.
LOG T "LOG/OUT" MSG	Retrieves all system message entries and writes them to the file called LOG/OUT.

DLP Type Abbreviations

Table 7-3 lists the data link processors (DLP) type abbreviations used in the LOGANALYZER output and the full names of the DLP types they stand for.

Table 7–3. DLP Abbreviations

Abbreviation	Name
CP1	CARD PUNCH
CR1	CARD READER
FR1	DATACOM LSP: SUB-BROADBAND (PROM)
FR2	DATACOM LSP: SUB-BROADBAND (RAM)
FR3	DATACOM LSP: 56KB BROADBAND BIT
HC2	HOST CONTROL-2
HT1	STANDARD HOST TRANSFER (DISK)
HTS1	SEQUENTIAL HOST TRANSFER (DISK)
HTS2	SEQUENTIAL HOST TRANSFER-2 (B9494-12/B9494-24 DISKS)

continued

Table 7–3. DLP Abbreviations (cont.)

Abbreviation	Name
HY1	HYPERchannel PERIPHERAL CONTROL
ICP1	INBUILT COMMUNICATIONS PROCESSOR
IPIPK1	IPI 9399-H DISK CONTROLLER
IPIPK2	IPI M9730 DISK CONTROLLER
MT1	PE MAG TAPE
MT2	GCR MAG TAPE
MT3	9-TRACK NRZ MAG TAPE
MT5	GCR FORMATTER-TYPE MAG TAPE
MT6	STREAMER TAPE
MTFIPS1	FIPS TAPE
MTFIPS3	CARTRIDGE TAPE
IP	IMAGE PRINTER
ODT1	OPERATOR DISPLAY
ODT2	OPERATOR DISPLAY-2
PK1SCSI(NATIVE)	Native Small Computer System Interface Disk
SC1	DATACOM NSP: STANDARD (MODEL 1 & 2)
SC2	DATACOM NSP: MULTIPLE HOST
SC3	DATACOM NSP: BLOCKED MESSAGES
SC4	DATACOM NSP: EXTENDED MEMORY
SC5	DATACOM NSP: DCDLP
SCSI1	SMALL COMPUTER SYSTEM INTERFACE DLP
SCSIDISK	SMALL COMPUTER SYSTEM INTERFACE DISK (131 SCSI, 130 SCSI, RESERVED FOR SCSI)
SCSITAPE	SMALL COMPUTER SYSTEM INTERFACE TAPE
SMD1	STORAGE MODULE DEVICE (226/236/256 DISK PACKS)
TP1	750/1100/1500 LPM TRAIN PRINTER
TP2	B9246-20 2000 LPM BUFFERED PRINTER
TP3	B924-B/B924-C 1200/2000 LPM BUFFERED DRUM LINE PRINTER
TP5	B9246-X AND B924-X BUFFERED PRINTER
VIM3	VOICE INTERFACE MODULE-3

Output Illustrations

The following figures illustrate the output for the various LOGANALYZER options discussed in this section. Only the first page of the output is shown, so not all the examples given are complete. Note that output differs depending on the type of hardware being used.

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 17:39:19

REQUEST:  PRINTER DATE (UNSORTED BY DEFAULT)

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 29773 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 17:40:00
```

Figure 7–1. DATE Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 11:50:21

REQUEST:  PRINTER 1100 TO 1103 ALL (UNSORTED BY DEFAULT)

SUMLOG #000786 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 10:55:49
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 22267 RECORDS FROM 04/04/1990 10:56:00 TO 04/04/1990 11:51:00

Wednesday, April 4, 1990
   11:00:01    OPEN 2532  EXT NAME: (ALIA)MAIL/NAMESKEY.
                          INT NAME: NAMESKEY.
                          FAMILY NAME: PACK
                          FILE ACCESS RULE = DECLARER   (ACTOR =      STACK 098, JOB 2532, TASK 2543)
                                                        (DECLARER = STACK 083, JOB 2532, TASK 2532)
                          USE = CLOSED KIND=PACK   FILEKIND= DATA     FILE SIZE 27 SEGS.  UNIT NUMBER 240
                          OPENTYPE: AVAILABLE, POSITION: 0, MOTION: NONE
   11:00:01    OPEN 2543  EXT NAME: (ALIA)MAIL/NAMESKEY.
                          INT NAME: NAMESKEY.
                          FAMILY NAME: PACK
                          FILE ACCESS RULE = DECLARER   (ACTOR =      STACK 098, JOB 2532, TASK 2543)
                                                        (DECLARER = STACK 083, JOB 2532, TASK 2532)
                          USE = CLOSED KIND=PACK   FILEKIND= DATA     FILE SIZE 27 SEGS.  UNIT NUMBER 240
                          OPENTYPE: AVAILABLE, POSITION: 0, MOTION: NONE
   11:00:01   CLOSE 2532  EXT NAME: (ALIA)MAIL/NAMESKEY.
                          INT NAME: NAMESKEY.
                          FAMILY NAME: PACK
                          FILE ACCESS RULE = DECLARER   (ACTOR =      STACK 098, JOB 2532, TASK 2543)
                                                        (DECLARER = STACK 083, JOB 2532, TASK 2532)
                          IO TIME      : 00:00:00.0000
                          CLOSE TYPE: NORMAL   ASSOCIATION: RELEASE   DISPOSITION: REWIND
                          TRANSACTION COUNT = 1         PHYSICAL (READ COUNT=1, WRITE COUNT=0)
                          FILE STRUCTURE: ALIGNED180
                          BUFFER CATEGORY: BUFFERGOAL NOT APPLICABLE    BUFFERSIZE: 90
                          BLOCKSIZE     = 90            MIN/MAXRECSIZE    = 0/ 3         INTMODE = SINGLE
                          PERM   SPEED=00   AREASIZE 102 SECTORS   FILE SIZE 27 SECTORS
                          SERIAL NO 682220   CREATION DATE 90093 CYCLE 1 VERSION 0 SAVEFACTOR 0
   11:00:01   CLOSE 2543  EXT NAME: (ALIA)MAIL/NAMESKEY.
                          INT NAME: NAMESKEY.
                          FAMILY NAME: PACK
                          FILE ACCESS RULE = DECLARER   (ACTOR =      STACK 098, JOB 2532, TASK 2543)
                                                        (DECLARER = STACK 083, JOB 2532, TASK 2532)
                          IO TIME      : 00:00:00.0000
                          CLOSE TYPE: NORMAL   ASSOCIATION: RELEASE   DISPOSITION: REWIND
                          TRANSACTION COUNT = 1         PHYSICAL (READ COUNT=1, WRITE COUNT=0)
                          FILE STRUCTURE: ALIGNED180
                          BUFFER CATEGORY: BUFFERGOAL NOT APPLICABLE    BUFFERSIZE: 90
                          BLOCKSIZE     = 90            MIN/MAXRECSIZE    = 0/ 3         INTMODE = SINGLE
                          PERM   SPEED=00   AREASIZE 102 SECTORS   FILE SIZE 27 SECTORS
                          SERIAL NO 682220   CREATION DATE 90093 CYCLE 1 VERSION 0 SAVEFACTOR 0
   11:00:01    OPEN 2532  EXT NAME: (ALIA)MAIL/NAMESKEY.
                          INT NAME: NAMESKEY.
                          FAMILY NAME: PACK
                          FILE ACCESS RULE = DECLARER   (ACTOR =      STACK 098, JOB 2532, TASK 2543)
                                                        (DECLARER = STACK 083, JOB 2532, TASK 2532)
                          USE = CLOSED KIND=PACK   FILEKIND= DATA     FILE SIZE 27 SEGS.  UNIT NUMBER 240
                          OPENTYPE: AVAILABLE, POSITION: 0, MOTION: NONE
   11:00:01    OPEN 2543  EXT NAME: (ALIA)MAIL/NAMESKEY.
```

Figure 7–2. ALL Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 11:49:01

REQUEST:  PRINTER 1100 TO 1103 RAW (UNSORTED BY DEFAULT)

SUMLOG #000786 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 10:55:49
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 21656 RECORDS FROM 04/04/1990 10:56:00 TO 04/04/1990 11:50:00

04/04/1990 11:00:01.5715   2532/2532  TYPE: 1.5  LENGTH: 22 (15 FIXED)
          010109E409E4  000000015FEE  0003D78402ED  3C1600010005  00000000032A  ...U.U  ....^.  ..PD..  ......  ......
          00000020000F  000000400011  800011C00000  000000000000  000000002002  ......  ......  ...{..  ......  ......
          0000000000F0  098009E409EF  083009E409E4  000000000000  000000100015  .....0  ...U..  ...U.U  ......  ......
          0C010108D5C1  D4C5E2D2C5E8  16030304D4C1  C9D304D4C1C9  D308D5C1D4C5  ....NA  MESKEY  ....MA  IL.MAI  L.NAME
          E2D2C5E80000  04D7C1C3D200                                            SKEY..  .PACK.

04/04/1990 11:00:01.5719   2532/2543  TYPE: 1.5  LENGTH: 22 (15 FIXED)
          010109E409EF  000000015FEE  0003D7840376  3C1600010005  00000000032A  ...U..  ....^.  ..PD..  ......  ......
          00000020000F  000000400011  800011C00000  000000000000  000000002002  ......  ......  ...{..  ......  ......
          0000000000F0  098009E409EF  083009E409E4  000000000000  000000100015  .....0  ...U..  ...U.U  ......  ......
          0C010108D5C1  D4C5E2D2C5E8  16030304D4C1  C9D304D4C1C9  D308D5C1D4C5  ....NA  MESKEY  ....MA  IL.MAI  L.NAME
          E2D2C5E80000  04D7C1C3D200                                            SKEY..  .PACK.

04/04/1990 11:00:01.5821   2532/2532  TYPE: 1.6  LENGTH: 37 (30 FIXED)
          010209E409E4  000000015FEE  0003D784142B  782500010006  00000000032A  ...U.U  ....^.  ..PD..  ......  ......
          00000020001E  000000400020  800011C00002  000010015FED  005A00000003  ......  ......  ...{..  ......  .}....
          000000000001  000000000000  400000000066  F6F8F2F2F2F0  000400001000  ......  ......  ......  682220  ......
          000000010101  000001000000  000000000000  000000000000  0000000000F0  ......  ......  ......  ......  .....0
          098009E409EF  083009E409E4  000000100024  000000000000  00000000005A  ...U..  ...U.U  ......  ......  .....]
          000000000001  400000000001  000000000001  000000000000  000400000000  ......  ......  ......  ......  ......
          0C010108D5C1  D4C5E2D2C5E8  16030304D4C1  C9D304D4C1C9  D308D5C1D4C5  ....NA  MESKEY  ....MA  IL.MAI  L.NAME
          E2D2C5E80000  04D7C1C3D200                                            SKEY..  .PACK.

04/04/1990 11:00:01.5825   2532/2543  TYPE: 1.6  LENGTH: 37 (30 FIXED)
          010209E409EF  000000015FEE  0003D7841481  782500010006  00000000032A  ...U..  ....^.  ..PD..  ......  ......
          00000020001E  000000400020  800011C00002  000010015FED  005A00000003  ......  ......  ...{..  ......  .}....
          000000000001  000000000000  400000000066  F6F8F2F2F2F0  000400001000  ......  ......  ......  682220  ......
          000000010101  000001000000  000000000000  000000000000  0000000000F0  ......  ......  ......  ......  .....0
          098009E409EF  083009E409E4  000000100024  000000000000  00000000005A  ...U..  ...U.U  ......  ......  .....]
          000000000001  400000000001  000000000001  000000000000  000400000000  ......  ......  ......  ......  ......
          0C010108D5C1  D4C5E2D2C5E8  16030304D4C1  C9D304D4C1C9  D308D5C1D4C5  ....NA  MESKEY  ....MA  IL.MAI  L.NAME
          E2D2C5E80000  04D7C1C3D200                                            SKEY..  .PACK.

04/04/1990 11:00:01.6214   2532/2532  TYPE: 1.5  LENGTH: 22 (15 FIXED)
          010109E409E4  000000015FEE  0003D7845400  3C1600010005  00000000032A  ...U.U  ....^.  ..PD..  ......  ......
          00000020000F  000000400011  800011C00000  000000000000  000000002002  ......  ......  ...{..  ......  ......
          0000000000F0  098009E409EF  083009E409E4  000000000000  000000100015  .....0  ...U..  ...U.U  ......  ......
          0C010108D5C1  D4C5E2D2C5E8  16030304D4C1  C9D304D4C1C9  D308D5C1D4C5  ....NA  MESKEY  ....MA  IL.MAI  L.NAME
          E2D2C5E80000  04D7C1C3D200                                            SKEY..  .PACK.

04/04/1990 11:00:01.6796   2532/2543  TYPE: 1.5  LENGTH: 22 (15 FIXED)
          010109E409EF  000000015FEE  0003D784B2D9  3C1600010005  00000000032A  ...U..  ....^.  ..PD.R  ......  ......
```

Figure 7–3. RAW Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 11:53:13

REQUEST:  PRINTER CONFIG (UNSORTED BY DEFAULT)

SUMLOG #000786 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 10:55:49
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 23504 RECORDS FROM 04/04/1990 10:56:00 TO 04/04/1990 11:54:00

Wednesday, April 4, 1990

* * * * * * * * * * * * * * * * * *   H A R D W A R E   C O N F I G U R A T I O N    04/04/1990 10:55:49    * * * * * * * * * * * *
A15 SYSTEM WITH 2 DATA PROCESSORS AND 1 IO PROCESSOR , PARTITION # 1
HOSTNAME : MPA15C.

PROCESSOR CONFIGURATION:
 2 PROCESSORS:
   2 CENTRAL PROCESSOR MODULES: 4-5
     CPM 4: [ERL= 08.0,SLL= 1F.A000,SCL= 0,FEATURES= 0 (ALL HEX),CREATION DATE = 09/11/1987]
     CPM 5: [ERL= 08.0,SLL= 1F.A000,SCL= 0,FEATURES= 0 (ALL HEX),CREATION DATE = 09/11/1987]

I/O CONFIGURATION:
 HDU 0:                                      FIRMWARE LEVEL: SLL=B1,FRL=1,ERL=1
     MLI 10000:                                 BASE 1000
     MLI 10001:                                 BASE 1100
     MLI 10010:                                 BASE 1200
     MLI 10011:                                 BASE 1300
     MLI 10020:                                 BASE 2000
     MLI 10021:                                 BASE 2100
BASE 1300:
     HTS2      DLP 1304                      FIRMWARE LEVEL: AD <PATCHED>
     HTS2      DLP 1305                      FIRMWARE LEVEL: AD <PATCHED>
     MTFIPS1 DLP 1307                        FIRMWARE LEVEL: 0215
BASE 2100:
     HTS1      DLP 2104                      FIRMWARE LEVEL: UC
BASE 1100:
     ODT1      DLP 1101
     HTS1      DLP 1104                      FIRMWARE LEVEL: UC
     HTS1      DLP 1106                      FIRMWARE LEVEL: UC
     HC2       DLP 100                       FIRMWARE LEVEL: 8502
BASE 2400: <NO PATH>
     LSP2      DLP 118 <NO PATH>             FIRMWARE LEVEL: 0100010E
     LSP2      DLP 119 <NO PATH>             FIRMWARE LEVEL: 0100010E
     LSP2      DLP 120 <NO PATH>             FIRMWARE LEVEL: 0100010E
BASE 1200:
     DCDLP     DLP 107                       FIRMWARE LEVEL: 28.5
     DCDLP     DLP 110                       FIRMWARE LEVEL: 28.5
     NSP3      DLP 111                       FIRMWARE LEVEL: 15.80
BASE 2000:
     HC2       DLP 101                       FIRMWARE LEVEL: 8502
     NSP3      DLP 109                       FIRMWARE LEVEL: 15.79
BASE 1500: <NO PATH>
     LSP2      DLP 115 <NO PATH>             FIRMWARE LEVEL: 02000205
     LSP2      DLP 117 <NO PATH>             FIRMWARE LEVEL: 02000205
BASE 1000:
     TP2       DLP 1003
     HTS1      DLP 1004                      FIRMWARE LEVEL: UC
```

Figure 7–4. CONFIG Option Output

```
- - - - - - - - - - - PHYSICAL MEMORY CONFIGURATION- - - - - - - - - - - -
PROCESSOR #1Ø
     MEMORY MODULE Ø ON PORT Ø,
          ADDRESS SELECTION   : Ø
          SUB_MODULES PRESENT : Ø,1
          SUB_MODULES ON-LINE : Ø,1 (4MW)
          INTERLACING         : ENABLED,  2-WAY ADDR2 SUB MOD Ø
     MEMORY MODULE 1 ON PORT 2,
          ADDRESS SELECTION   : 2
          SUB_MODULES PRESENT : Ø,1
          SUB_MODULES ON-LINE : Ø,1 (4MW)
          INTERLACING         : ENABLED,  2-WAY ADDR2 SUB MOD Ø
PROCESSOR #9
     MEMORY MODULE Ø ON PORT 1,
          ADDRESS SELECTION   : 1
          SUB_MODULES PRESENT : Ø,1
          SUB_MODULES ON-LINE : Ø,1 (4MW)
          INTERLACING         : ENABLED,  2-WAY ADDR2 SUB MOD Ø
     MEMORY MODULE 1 ON PORT 3,
          ADDRESS SELECTION   : 3
          SUB_MODULES PRESENT : Ø,1
          SUB_MODULES ON-LINE : Ø,1 (4MW)
          INTERLACING         : ENABLED,  2-WAY ADDR2 SUB MOD Ø
```

Figure 7–5. CONFIG Option Output (A 10 Systems) Record

```
             FAILURE ANALYSIS SUMMARY
             ------------------------

                           RMM Ø
         NON-FATAL               FATAL
 CARD:PRIMARY:SECONDARY:PRIMARY:SECONDARY:
------:-------:---------:-------:---------:
  IOP:       :    1    :       :         :
  DTU:       :         :       :         :
TCPCN:       :         :       :         :
TCPDT:       :         :       :         :
TCPTH:       :         :       :         :
  ITC:   2   :         :       :         :
MICAA:   1   :         :       :         :
MICDA:   1   :         :       :         :
 PA Ø:   1   :         :       :         :
 PA 1:       :         :       :         :
 PA 2:   1   :         :       :         :
 PA 3:       :         :       :         :
 PA 4:       :         :       :         :
 PA 5:       :         :       :         :
 PA 6:       :         :       :         :
------:-------:---------:-------:---------:
TOTAL:   6   :    1    :       :         :
```

Figure 7–6. FASUMMARY Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.037,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 822) ON 04/04/1990 12:13:13

REQUEST:  PRINTER HL (UNSORTED BY DEFAULT)

SUMLOG #002703 CREATED BY A15 (SYSTEM SERIAL = 822) ON 04/04/1990 11:43:19
TITLE = *SYSTEM/SUMLOG ON OPERATIONS (CURRENT SUMLOG).

FILE CONTAINS 22961 RECORDS FROM 04/04/1990 11:43:00 TO 04/04/1990 12:14:00

Wednesday, April 4, 1990

SYSTEM 822 HALT LOADED 04/04/1990 11:46:09 H/L REASON : MANUAL
      *SYSTEM/MCP/39023G ON DISK,  VERSION 3.9.024, (MCP/AS).

NORMAL TERMINATION FOR LOGANALYZER PROGRAM.
```

Figure 7–7. HL Option Output

```
LOGANALYZER VERSION: 39.022.045,  SDASUPPORT VERSION: 39.020.035,  JOBFORMATTER VERSION: 39.022.324.
MCP *SYSTEM/MCP/39022C VERSION: 39.022 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 14:04:03

REQUEST:  P /000705 IOSUMMARY (UNSORTED BY DEFAULT)

SUMLOG #000705 CREATED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 09:33:46
TITLE = *SUMLOG/8/031490/000705 ON PACK.

FILE CONTAINS 88014 RECORDS FROM 03/14/1990 09:34:00 TO 03/14/1990 13:47:00

Wednesday, March 14, 1990

 I/O ERROR RESULT SUMMARY FOR THE CURRENT  LOG ANALYSIS :

ERROR  UNIT                LOGICAL          RESULT                   # RECOVERED IN   # RECOVERED IN   IRRECOVERED
COUNT  TYPE   UNIT#  R/W   RESULT          ANALYSIS                  LEQ 3 ATTEMPTS   GEQ 4 ATTEMPTS      COUNT
-----  ------ -----  ---   -------  ------------------------------   --------------   --------------   -----------
   53  PACK     73   R      00000   SUCCESSFUL DATA RETRY                  53               0                0
    4  PACK     73   R      00000   DATA ERROR CORRECTION                   4               0                0
    2  TAPE     28     W    00081   DATA PARITY ERROR                       2               0                0
    2  PACK     45   R      00000   SUCCESSFUL DATA RETRY                   2               0                0
    2  PACK    247   R      00000   DATA ERROR CORRECTION                   2               0                0
    2  TAPE     31     W    00081   DATA PARITY ERROR                       2               0                0
    1  TAPE     31   R      00081   UNIT CHECK                              0               0                1
    1  PACK    241   R      00000   DATA ERROR CORRECTION                   1               0                0
    1  PACK    240   R      00000   DATA ERROR CORRECTION                   1               0                0
    1  PACK    246   R      00000   DATA ERROR CORRECTION                   1               0                0

NORMAL TERMINATION FOR LOGANALYZER PROGRAM.
```

Figure 7–8. IOSUMMARY Option Output

```
LOGANALYZER VERSION: 39.020.040,  SDASUPPORT VERSION: 39.018.034,  JOBFORMATTER VERSION: 39.020.314.
MCP *SYSTEM/MCP/39020/DIAG VERSION: 39.020 (MCP/AS)
ANALYZED BY MicroA (SYSTEM SERIAL = 65535) ON 03/14/1990 14:54:11

REQUEST:  P MAINFRAME (UNSORTED BY DEFAULT)

SUMLOG #000082 CREATED BY MicroA (SYSTEM SERIAL = 65535) ON 03/14/1990 14:37:16
TITLE = *SYSTEM/SUMLOG ON DISK (CURRENT SUMLOG).

FILE CONTAINS 967 RECORDS FROM 03/14/1990 14:37:00 TO 03/14/1990 14:55:00

Wednesday, March 14, 1990

MAINFRAME ERROR   03/14/1990 14:41:08  EVENT LOGGED AT 14:41:59
   TYPE  = MAINFRAME EVENT
   CAUSE = SYSTEM HALT LOADED (INIT)
MicroA CONFIGURATION:       DATA PROCESSOR=9
                            I/O  PROCESSOR=1
   REPORT:
      PROCESSOR 9 H/L REASON: ODT COMMAND

NORMAL TERMINATION FOR LOGANALYZER PROGRAM.
```

Figure 7–9. MAINFRAME Option Output

```
LOGANALYZER VERSION: 39.020.040,  SDASUPPORT VERSION: 39.018.034,  JOBFORMATTER VERSION: 39.020.314.
MCP *SYSTEM/MCP/39020/DIAG VERSION: 39.020 (MCP/AS)
ANALYZED BY MicroA (SYSTEM SERIAL = 65535) ON 03/14/1990 14:54:58

REQUEST:  P IOERROR (UNSORTED BY DEFAULT)

SUMLOG #000082 CREATED BY MicroA (SYSTEM SERIAL = 65535) ON 03/14/1990 14:37:16
TITLE = *SYSTEM/SUMLOG ON DISK (CURRENT SUMLOG).

FILE CONTAINS 1034 RECORDS FROM 03/14/1990 14:37:00 TO 03/14/1990 14:56:00

Wednesday, March 14, 1990

TAPE    49   (UNITTYPE=15 DLPTYPE=SCSITAPE).
                  IO START  AREA      AREA            IOP PATH    LOGICAL
 MIX     DATE       TIME    START     LENGTH            #  ADDR  RESULT DESC   SERIAL       FILE

 320 03/14/1990  14:39:53 0013D3DD 00018 CHARS       1  00010 000008008001 GAD

      NEW ERROR CODE = 2E                            SENSE KEY: BLANK CHECK                        TAPE STATUS: ?
      SENSE INFORMATION: 0000003C                    ADDITIONAL SENSE LENGTH: 6
      ADDITIONAL SENSE DATA: 000000002E00

                       RETRY  IOP           IOP                 DLP COMMAND             DLP RESULT
                            CONTROL    STATE&RESULT
                             00408     000000000005     27500000000000000003C0000   0401 0002 0000 003C
                       1 *   2040D     000000000027            "           "        0401 2000 0000 003C
                          (* INDICATES AN ACTUAL RETRY.  TOTAL # OF ACTUAL RETRIES FOR THIS I/O = 1)

                 14:40:07 000928F4 03F4E CHARS       1  00010 03F4E8008001 GAD

      NEW ERROR CODE = 2E                            SENSE KEY: BLANK CHECK                        TAPE STATUS: <ILLEGAL LENGTH>
      SENSE INFORMATION: 00003F4E                    ADDITIONAL SENSE LENGTH: 6
      ADDITIONAL SENSE DATA: 000000002E00

                       RETRY  IOP           IOP                 DLP COMMAND             DLP RESULT
                            CONTROL    STATE&RESULT
                             04428     000000000005     80500000000000003F4E0000   0409 0002 0000 3F4E
                       1 *   2442D     000000000027            "           "        C401 0000 0000 0000
                       2 *     "            "                  "           "        0401 0080 0000 0000
                          (* INDICATES AN ACTUAL RETRY.  TOTAL # OF ACTUAL RETRIES FOR THIS I/O = 2)


 I/O ERROR RESULT SUMMARY FOR THE CURRENT  LOG ANALYSIS :

ERROR  UNIT                  LOGICAL          RESULT                            # RECOVERED IN   # RECOVERED IN   IRRECOVERED
COUNT  TYPE   UNIT#   R/W    RESULT          ANALYSIS                           LEQ 3 ATTEMPTS   GEQ 4 ATTEMPTS      COUNT
-----  ------ -----   ---    -------- -------------------------------           --------------   --------------   -----------
    2  TAPE     49     R      08001  TIME LIMIT EXCEEDED                              2                0               0

I/O STATISTICS THAT HAVE BEEN ACCUMULATING FOR APPROXIMATELY 3 HOURS AND 27 MINUTES.

       UNIT             1000 BYTES                IO              IO            SUCCESSFUL
       TYPE   UNIT#    TRANSFERRED    R/W     OPERATIONS        ERRORS        H/W RECOVERIES
       ------ -----   ------------    ---     -----------       ------        --------------
       PACK     45          8764       R          2999             0                0
                                       W          1260             0                0
```

Figure 7-10. IOERROR Option Output (EMS Systems)

```
LOGANALYZER VERSION: 39.022.045,  SDASUPPORT VERSION: 39.020.035,  JOBFORMATTER VERSION: 39.022.324.
MCP *SYSTEM/MCP/39022C VERSION: 39.022 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 14:04:48

REQUEST:  P /000705 IOERROR MT (UNSORTED BY DEFAULT)

SUMLOG #000705 CREATED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 09:33:46
TITLE = *SUMLOG/8/031490/000705 ON PACK.

FILE CONTAINS 88014 RECORDS FROM 03/14/1990 09:34:00 TO 03/14/1990 13:47:00

Wednesday, March 14, 1990

TAPE   28  (UNITTYPE=15 DLPTYPE=MTFIPS1).
                      IO START  AREA     AREA          HDU  DLP     LOGICAL
 MIX     DATE           TIME     START   LENGTH          #          RESULT DESC  SERIAL      FILE

3425 03/14/1990  09:38:44 001ED569 0151E CHARS    0   1007 0151E0000081 DABMCP
                           EXTENDED RESULT DESC: 0844 089C 0040 2C00 001C 0000 D201 7FC0 9000 00F4 0000 1008

                      RETRY  HDU CONTROL   HDU STATE&RESULT DLP COMMAND      DLP RESULT          ANALYSIS BY MCP
                         *      40440          000000000005   400000000000  0401 1008 0000 0000 DATA PARITY ERROR
                        1       4C40D          000000000000   270200010000  0000 0000 0000 0000 ERROR FREE RESULT
                        2         "                 "         2E0000020000    "    "    "    "       "
                        3 *     4844D               "         400000000000    "    "    "    "       "
                         (* INDICATES AN ACTUAL RETRY.  TOTAL # OF ACTUAL RETRIES FOR THIS I/O = 2)

                 09:38:59 001ED569 0151E CHARS    0   1007 0151E0000081 DABMCP
                           EXTENDED RESULT DESC: 0844 109C 0040 2C00 001C 0000 D202 7FC0 9000 00F4 0000 1008

                      RETRY  HDU CONTROL   HDU STATE&RESULT DLP COMMAND      DLP RESULT          ANALYSIS BY MCP
                         *      40440          000000000005   400000000000  0401 1008 0000 0000 DATA PARITY ERROR
                        1       4C40D          000000000000   270200010000  0000 0000 0000 0000 ERROR FREE RESULT
                        2         "                 "         2E0000020000    "    "    "    "       "
                        3 *     4844D               "         400000000000    "    "    "    "       "
                         (* INDICATES AN ACTUAL RETRY.  TOTAL # OF ACTUAL RETRIES FOR THIS I/O = 2)


 I/O ERROR RESULT SUMMARY FOR THE CURRENT  LOG ANALYSIS :

ERROR  UNIT                LOGICAL        RESULT                        # RECOVERED IN   # RECOVERED IN    IRRECOVERED
COUNT  TYPE   UNIT#  R/W   RESULT         ANALYSIS                      LEQ 3 ATTEMPTS   GEQ 4 ATTEMPTS       COUNT
-----  ------ -----  ---   -------  ------------------------------      --------------   --------------    -----------
    2  TAPE    28     W    00081    DATA PARITY ERROR                         2                0                0
    2  TAPE    31     W    00081    DATA PARITY ERROR                         2                0                0
    1  TAPE    31    R     00081    UNIT CHECK                                0                0                1

NORMAL TERMINATION FOR LOGANALYZER PROGRAM.
```

Figure 7-11. IOERROR Option Output (HDU Systems)

```
LOGANALYZER VERSION: 39.020.040,  SDASUPPORT VERSION: 39.018.034,  JOBFORMATTER VERSION: 39.020.314.
MCP *SYSTEM/MCP/39020/DIAG VERSION: 39.020 (MCP/AS)
ANALYZED BY MicroA (SYSTEM SERIAL = 65535) ON 03/14/1990 14:55:32

REQUEST:  P MAINT (SORTED BY DEFAULT)

SUMLOG #000082 CREATED BY MicroA (SYSTEM SERIAL = 65535) ON 03/14/1990 14:37:16
TITLE = *SYSTEM/SUMLOG ON DISK (CURRENT SUMLOG).

FILE CONTAINS 1102 RECORDS FROM 03/14/1990 14:37:00 TO 03/14/1990 14:57:00

Wednesday, March 14, 1990

TAPE   49  (UNITTYPE=15 DLPTYPE=SCSITAPE).
                IO START  AREA      AREA            IOP PATH    LOGICAL
 MIX    DATE      TIME    START    LENGTH            #  ADDR  RESULT DESC  SERIAL       FILE

 320 03/14/1990  14:39:53 0013D3DD 00018 CHARS     1  00010 000008008001 GAD

      NEW ERROR CODE = 2E                         SENSE KEY: BLANK CHECK                   TAPE STATUS: ?
      SENSE INFORMATION: 0000003C                 ADDITIONAL SENSE LENGTH: 6
      ADDITIONAL SENSE DATA: 000000002E00

                         RETRY  IOP          IOP                DLP COMMAND              DLP RESULT
                             CONTROL    STATE&RESULT
                              00408     000000000005    2750000000000000003C0000  0401 0002 0000 003C
                           1 *  2040D     000000000027            "          "        0401 2000 0000 003C
                            (* INDICATES AN ACTUAL RETRY.  TOTAL # OF ACTUAL RETRIES FOR THIS I/O = 1)

                 14:40:07 000928F4 03F4E CHARS     1  00010 03F4E8008001 GAD

      NEW ERROR CODE = 2E                         SENSE KEY: BLANK CHECK                   TAPE STATUS: <ILLEGAL LENGTH>
      SENSE INFORMATION: 00003F4E                 ADDITIONAL SENSE LENGTH: 6
      ADDITIONAL SENSE DATA: 000000002E00

                         RETRY  IOP          IOP                DLP COMMAND              DLP RESULT
                             CONTROL    STATE&RESULT
                              04428     000000000005    80500000000000003F4E0000  0409 0002 0000 3F4E
                           1 *  2442D     000000000027            "          "        C401 0000 0000 0000
                           2 *    "           "                   "          "        0401 0080 0000 0000
                            (* INDICATES AN ACTUAL RETRY.  TOTAL # OF ACTUAL RETRIES FOR THIS I/O = 2)

# # # # # # #  UNIT COUNTS REPORTED FOR 5 UNITS ON  03/14/1990 14:37:17   # # # # # # # #
      UNIT              1000 BYTES               IO                IO          SUCCESSFUL
      TYPE   UNIT#     TRANSFERRED   R/W   OPERATIONS          ERRORS      H/W RECOVERIES
      ------ -----     -----------   ---   -----------         ------      --------------
      PACK    44             102     R             4              0               0
                                       W           0              0               0
      PACK    45            8764     R          2999              0               0
                                       W        1260              0               0
      PACK    47             811     R           137              0               0
                                       W           4              0               0
      TAPE    49               7     R            11              0               0
                                       W           3              0               0
      ODT      1            1382     R            48              0               0
                                       W        1293              0               0
                  THE ABOVE COUNTS HAVE BEEN ACCUMULATING FOR 3 HOURS AND 27 MINUTES.
```

Figure 7-12. MAINT Option Output (EMS Systems)

```
LOGANALYZER VERSION: 39.032.052,  SDASUPPORT VERSION: 39.030.042,  JOBFORMATTER VERSION: 39.032.400.
MCP *SYSTEM/MCP/39032D VERSION: 39.032 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 08/03/1990 18:27:59

REQUEST:  PRINTER MAINT (SORTED BY DEFAULT)

SUMLOG #001224 CREATED BY A15 (SYSTEM SERIAL = 8) ON 08/03/1990 15:40:58
TITLE = *SYSTEM/SUMLOG ON DISKB (CURRENT SUMLOG).

FILE CONTAINS 36956 RECORDS FROM 08/03/1990 15:41:00 TO 08/03/1990 18:29:00

Friday, August 3, 1990

PACK   63  (UNITTYPE=17  SUBTYPE=659 INT        DLPTYPE=HTS1).
                IO START  AREA     AREA          HDU  DLP     LOGICAL
 MIX    DATE      TIME    START   LENGTH           #        RESULT DESC  SERIAL  ROW FILE

7140 08/03/1990  16:18:38 00F81554 00A8C WORD   0   1104 00A8C0000000 659063

      CONTROLWARE ID: E4C3                  RD: 0080     R/D TAG: 0862                   MPM ADDRESS: 0000
      DISK DDP STATUS: 4100                 DISK DDP DIAGNOSTICS: 00009C0D                C/D: OP CODE-8 SUB OP-2 VARIANT-0 UNIT-3
      DRIVE STATUS/MESSAGE WORD 1: 0000  DRIVE MESSAGE WORD 2: 0000                       # OF RETRIES ON ERROR ADDR: 7
      ERROR DISK ADDRESS: 62717 (CYL 46 HD 9 SEC 37)      DISK ADDRESS FROM CDL: 62680 (CYL 46 HD 9 SEC 0)
      TEMP R/D: 000000000000  TAG: 0000  DRIVE MESSAGE WORD 2: 00000000                   BINARY CYLINDER ADDR: 23

                         RETRY  HDU CONTROL   HDU STATE&RESULT DLP COMMAND      DLP RESULT
                                   40C30          000000000005   80300000F4D8  0401 0080 F400 F4FD SUCCESSFUL DATA RETRY


PACK   244  (UNITTYPE=17  SUBTYPE=3682 SEQ       DLPTYPE=HTS2).
                IO START  AREA     AREA          HDU  DLP     LOGICAL
 MIX    DATE      TIME    START   LENGTH           #        RESULT DESC  SERIAL  ROW FILE

9706             16:38:52 002AC340 07E90 CHARS   0   1305 07E900000000 682244

      CONTROLWARE ID  : C1C4      R/D TAG: 0862     CTRL INFO: 0324                       SC STATUS 40-43: B940 0000 0000 2130
      ISI DDP STATUS 1: 0000                  ERROR SYMPTOM CODE: NOT VALID               C/D: OP CODE-8 SUB OP-0 VARIANT-0 UNIT-4
      CONFIGURE INFO  : B90C# OF RETRIES ON ERROR ADDR: 0                                  TOTAL # OF RETRIES: 0
      ERROR DISK ADDRESS   : 4013560 (CYL 1470 HD 2 SEC 96)
      DISK ADDRESS FROM CDL: 4013432 (CYL 1470 HD 1 SEC 150)

                         RETRY  HDU CONTROL   HDU STATE&RESULT DLP COMMAND      DLP RESULT
                                   40420          000000000005   8040003D3D78  0401 0080 F83D 3DF8 DATA ERROR CORRECTION


* * * * * * * * * *   H A R D W A R E   C O N F I G U R A T I O N   08/03/1990 15:40:58  * * * * * * * * *
A15 SYSTEM WITH 2 DATA PROCESSORS AND 1 IO PROCESSOR , PARTITION # 1

PROCESSOR CONFIGURATION:
 2 PROCESSORS:
   2 CENTRAL PROCESSOR MODULES: 4-5
     CPM 4: [ERL= 08.0,SLL= 28.A000,SCL= 0,FEATURES= 0 (ALL HEX),CREATION DATE = 08/31/1989]
     CPM 5: [ERL= 08.0,SLL= 28.A000,SCL= 0,FEATURES= 0 (ALL HEX),CREATION DATE = 08/31/1989]

I/O CONFIGURATION:
 HDU 0:                                   FIRMWARE LEVEL: SLL=B2,FRL=1,ERL=1
     MLI 10000:                              BASE 1000
     MLI 10001:                              BASE 1100
     MLI 10010:                              BASE 1200
```

Figure 7-13. MAINT Option Output (HDU Systems)

```
- - - - - - - - - - CORRECTABLE MEMORY ERROR ACTIVITY - - - - - - - - - -
A HARD CORRECTABLE ERROR CAUSED AN ERROR SAMPLING PERIOD TO BE
INITIATED.
SAMPLING PERIOD DURATION (MINUTES:SECONDS): 14:33
DURING 873 SAMPLES, 255 CORRECTABLE ERRORS WERE DETECTED IN 6 DEVICES.
ERROR OCCURRENCES BY DEVICE:
 DPM NUMBER  CARD NUMBER  CHIP ROW  CHIP COL  PROC NUMBER  # OF ERRORS
 --- ------  ---- ------  ---- ---  ---- ---  ---- ------  - -- ------
     3           21          M         26         Ø9           135
     Ø           17          F         14         1Ø           1Ø3
     1           Ø5          K         Ø7         Ø9            12
     Ø           Ø8          D         3Ø         1Ø             2
     2           17          P         Ø1         1Ø             1
     Ø           18          F         26         1Ø             1
     1           Ø4          F         11         Ø9             1
```

Figure 7–14. MAINT CPU Option Output (A 10 Systems)

```
LOGANALYZER VERSION: 39.022.045,  SDASUPPORT VERSION: 39.020.035,  JOBFORMATTER VERSION: 39.022.324.
MCP *SYSTEM/MCP/39022C VERSION: 39.022 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 14:08:51

REQUEST:  P /000705 MAINT PK (SORTED BY DEFAULT)

SUMLOG #000705 CREATED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 09:33:46
TITLE = *SUMLOG/8/031490/000705 ON PACK.

FILE CONTAINS 88014 RECORDS FROM 03/14/1990 09:34:00 TO 03/14/1990 13:47:00

Wednesday, March 14, 1990


PACK    45  (UNITTYPE=17  SUBTYPE=659 INT          DLPTYPE=HTS1).
                  IO START  AREA      AREA           HDU  DLP     LOGICAL
 MIX     DATE       TIME    START    LENGTH           #         RESULT DESC  SERIAL  ROW FILE

4356 03/14/1990  11:55:51 0028B12B 07E90 CHARS   0   1106 07E900000000 659045

      CONTROLWARE ID: E4C3                 RD: 0080     R/D TAG: 0862                 MPM ADDRESS: 0000
      DISK DDP STATUS: 4100                DISK DDP DIAGNOSTICS: 00009C0D             C/D: OP CODE-8 SUB OP-0 VARIANT-0 UNIT-1
      DRIVE STATUS/MESSAGE WORD 1: 0000  DRIVE MESSAGE WORD 2: 0000                   # OF RETRIES ON ERROR ADDR: 1
      ERROR DISK ADDRESS: 1931145 (CYL 1435 HD 11 SEC 80)     DISK ADDRESS FROM CDL: 1930972 (CYL 1435 HD 9 SEC 87)
      TEMP R/D: 000000000000  TAG: 0000  DRIVE MESSAGE WORD 2: 00000000           BINARY CYLINDER ADDR: 717

                      RETRY  HDU CONTROL   HDU STATE&RESULT DLP COMMAND      DLP RESULT         ANALYSIS BY MCP
                                40420         000000000005   8010001D76DC  0401 0080 F41D 7789 SUCCESSFUL DATA RETRY

4379             11:59:09 0028B12B 07E90 CHARS   0   1106 07E900000000 659045

      CONTROLWARE ID: E4C3                 RD: 0080     R/D TAG: 0862                 MPM ADDRESS: 0000
      DISK DDP STATUS: 4100                DISK DDP DIAGNOSTICS: 00009C0D             C/D: OP CODE-8 SUB OP-0 VARIANT-0 UNIT-1
      DRIVE STATUS/MESSAGE WORD 1: 0000  DRIVE MESSAGE WORD 2: 0000                   # OF RETRIES ON ERROR ADDR: 2
      ERROR DISK ADDRESS: 1931145 (CYL 1435 HD 11 SEC 80)     DISK ADDRESS FROM CDL: 1930972 (CYL 1435 HD 9 SEC 87)
      TEMP R/D: 000000000000  TAG: 0000  DRIVE MESSAGE WORD 2: 00000000           BINARY CYLINDER ADDR: 717

                      RETRY  HDU CONTROL   HDU STATE&RESULT DLP COMMAND      DLP RESULT         ANALYSIS BY MCP
                                40420         000000000005   8010001D76DC  0401 0080 F41D 7789 SUCCESSFUL DATA RETRY


PACK    73  (UNITTYPE=17  SUBTYPE=659 INT          DLPTYPE=HTS1).
                  IO START  AREA      AREA           HDU  DLP     LOGICAL
 MIX     DATE       TIME    START    LENGTH           #         RESULT DESC  SERIAL  ROW FILE

8259             10:08:09 00C3CA8E 00A8C WORD   0   1104 00A8C0000000 659073

      CONTROLWARE ID: E4C3                 RD: 0080     R/D TAG: 0862                 MPM ADDRESS: 0000
      DISK DDP STATUS: 4100                DISK DDP DIAGNOSTICS: 00009C0D             C/D: OP CODE-8 SUB OP-2 VARIANT-0 UNIT-D
      DRIVE STATUS/MESSAGE WORD 1: 0000  DRIVE MESSAGE WORD 2: 0000                   # OF RETRIES ON ERROR ADDR: 1
      ERROR DISK ADDRESS: 613929 (CYL 456 HD 6 SEC 69)        DISK ADDRESS FROM CDL: 613860 (CYL 456 HD 6 SEC 0)
      TEMP R/D: 000000000000  TAG: 0000  DRIVE MESSAGE WORD 2: 00000000           BINARY CYLINDER ADDR: 228
```

Figure 7–15. <device option> Output

```
LOGANALYZER VERSION: 39.022.045,  SDASUPPORT VERSION: 39.020.035,  JOBFORMATTER VERSION: 39.022.324.
MCP *SYSTEM/MCP/39022C VERSION: 39.022 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 14:14:32

REQUEST:  P /000705 0933 TO 0934 BNA USERCODE. (UNSORTED BY DEFAULT)

SUMLOG #000705 CREATED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 09:33:46
TITLE = *SUMLOG/8/031490/000705 ON PACK.

FILE CONTAINS 88014 RECORDS FROM 03/14/1990 09:34:00 TO 03/14/1990 13:47:00

Wednesday, March 14, 1990

03/14/1990 09:33:46.7423  8293  PLM   RPT    SEND PLM ID TO HOST  #296
                                             PLM-INCARNATION ID: 000004D233623362

03/14/1990 09:33:46.9541  8293  RTR   CMD    ROUTER CONTROL FRAME RECEIVED FROM NODE TRENGA #46
                                             ROUTER LEVEL PRIORITY FRAME    TRANSIT COUNT: 1
                                             FRAME TYPE: NET-CHANGE    FRAME LENGTH: 14 BYTES
                                             FRAME: 22010024002E03010062010206CD

03/14/1990 09:33:48.3929  8293  PLM   RPT    SEND PLM ID TO HOST  #296
                                             PLM-INCARNATION ID: 000004D233623362

03/14/1990 09:33:50.8916  8293  PLM   RPT    SEND PLM ID TO HOST  #296
                                             PLM-INCARNATION ID: 000004D233623362

03/14/1990 09:33:55.6507  8293  RTR   CMD    ROUTER CONTROL FRAME RECEIVED FROM NODE TRENGA #46
                                             ROUTER LEVEL PRIORITY FRAME    TRANSIT COUNT: 1
                                             FRAME TYPE: NET-CHANGE    FRAME LENGTH: 14 BYTES
                                             FRAME: 22010024002E030101A4010407CB
03/14/1990 09:34:00.5773  8298  BNA   RPT    X.25 MCS DEBUG: STACK #08F @ 72355000
                                             PQ MESSAGE: CLASS 17, LSN = 0, SIZE = 10 WORDS, TEXT LENGTH = 21 CHARA
                                               CTERS

03/14/1990 09:34:01.0112  8293  RTR   CMD    ROUTER CONTROL FRAME RECEIVED FROM NODE TRENGA #46
                                             ROUTER LEVEL PRIORITY FRAME    TRANSIT COUNT: 1
                                             FRAME TYPE: NET-CHANGE    FRAME LENGTH: 14 BYTES
                                             FRAME: 22010024002E030100DC010405C9

03/14/1990 09:34:01.5092  8278  NSM   RPT    FROM PROGRAM AGENT: HOST

03/14/1990 09:34:01.5278  8278  NSM   RPT    FROM PROGRAM AGENT: CA

03/14/1990 09:34:01.5366  8278  NSM   RPT    FROM PROGRAM AGENT: NODE

03/14/1990 09:34:06.0775  8293  PLM   RPT    SEND PLM ID TO HOST CSDMFG #340
                                             PLM-INCARNATION ID: 000004D233623362

03/14/1990 09:34:06.0806  8293  PLM   RPT    RECEIVED PLM ID FROM HOST CSDMFG #340
                                             REMOTE INCARNATION ID: 000005C4BF4AC200

03/14/1990 09:34:06.5775  8293  PLM   RPT    SEND PLM ID TO HOST  #296
```

Figure 7–16. BNA Version 1 Option Output

```
LOGANALYZER VERSION: 39.020.040,  SDASUPPORT VERSION: 39.018.034,  JOBFORMATTER VERSION: 39.020.314.
MCP *SYSTEM/MCP/39020/DIAG VERSION: 39.020 (MCP/AS)
ANALYZED BY MicroA (SYSTEM SERIAL = 65535) ON 03/14/1990 14:56:23

REQUEST:  P BNAV2 USERCODE. (UNSORTED BY DEFAULT)

SUMLOG #000082 CREATED BY MicroA (SYSTEM SERIAL = 65535) ON 03/14/1990 14:37:16
TITLE = *SYSTEM/SUMLOG ON DISK (CURRENT SUMLOG).

FILE CONTAINS 1190 RECORDS FROM 03/14/1990 14:37:00 TO 03/14/1990 14:57:00

Wednesday, March 14, 1990

03/14/1990 14:44:35.9226  0411  BNAV2  HEADER CONTROL LEVEL = 1                    SOURCE HOST         = SFB112
                                       SOURCE TIME STAMP    = 3/14/90, 14:44:34.61  IMMEDIATE LCF HOST =
                                       CONTROLLING AGENT: ODT AGENT, DIALOG# 1      CONTROLLED AGENT: LOCAL NSM
                                       MESSAGE IS A NW COMMAND                      @ CONTROL LEVEL      2

                                       NET +

03/14/1990 14:44:43.2620  0401  BNAV2  HEADER CONTROL LEVEL = 1                    SOURCE HOST         = SFB112
                                       SOURCE TIME STAMP    = 3/14/90, 14:44:43.20  IMMEDIATE LCF HOST = SFB112

                                       MESSAGE IS A LOG REPORT                      @ CONTROL LEVEL      1

                                       SUBPORT OPEN: PORT ID 5  BY USING PROCESS 0400/0400 FOR PORT INTERNAL NAME NETCONTRO
                                       LPORT, FILE NAME = NETCONTROLPORT SUBPORT #1, ENVIRONMENT = HOST-LOCAL (SUBPORT ELEM
                                       ENT ENVIRONMENT NUMBER = 0) APPLICATION GROUP = *BNA CNTRL AGENT, YOUR HOST = SFB112
                                        YOUR HOST GROUP = SFB112 ACTUAL YOUR USERCODE = , IGNORE DIRECTORY = TRUE AVAILABLE
                                        ONLY = FALSE, YOUR PORT ID = 2, YOUR SUBPORT INDEX = 2 MY USERCODE = ; OPEN RECEIVE
                                       D AT 3/14/90, 14:44:32.70
                                       MY NAME = LCF, YOUR NAME = ODT

03/14/1990 14:44:43.3147  0401  BNAV2  HEADER CONTROL LEVEL = 1                    SOURCE HOST         = SFB112
                                       SOURCE TIME STAMP    = 3/14/90, 14:44:43.27  IMMEDIATE LCF HOST = SFB112

                                       MESSAGE IS A LOG REPORT                      @ CONTROL LEVEL      1

                                       SUBPORT OPEN: PORT ID 2  BY USING PROCESS 0011/0011 FOR PORT INTERNAL NAME NWPORT, F
                                       ILE NAME = NETCONTROLPORT SUBPORT #2, ENVIRONMENT = HOST-LOCAL (SUBPORT ELEMENT ENVI
                                       RONMENT NUMBER = 0) APPLICATION GROUP = *BNA CNTRL AGENT, YOUR HOST = SFB112 YOUR HO
                                       ST GROUP = SFB112 ACTUAL YOUR USERCODE = , IGNORE DIRECTORY = FALSE AVAILABLE ONLY =
                                        FALSE, YOUR PORT ID = 5, YOUR SUBPORT INDEX = 1 MY USERCODE = ; OPEN RECEIVED AT 3/
                                       14/90, 14:44:28.84
                                       MY NAME = ODT, YOUR NAME = LCF

03/14/1990 14:44:45.4375  0411  BNAV2  HEADER CONTROL LEVEL = 1                    SOURCE HOST         = SFB112
                                       SOURCE TIME STAMP    = 3/14/90, 14:44:45.23  IMMEDIATE LCF HOST =
                                       CONTROLLING AGENT: ODT AGENT, DIALOG# 1      CONTROLLED AGENT: LOCAL NSM
                                       MESSAGE IS A NW POSITIVE RESPONSE            @ CONTROL LEVEL      2

                                       INITIALIZING NETWORK MODE WITH NETWORK INITIALIZATION FILE = (SASHA)SFB112/NETINIT/V
                                       2
```

Figure 7–17. BNA Version 2 Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 17:32:23

REQUEST:  PRINTER MCS USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 28328 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 17:33:00

Tuesday, April 3, 1990
   15:57:01   LGOFF 5817  ORIGINATING LSN: 196
                          USERCODE: WATTON.
                          MCS: 1
                          STATION NAME: T100B20.
                          SIGN OFF BY NORMAL LOG-OFF
                          PROCESSOR TIME: 00:00:00.0000
                          I/O       TIME: 00:00:00.0000
                          ELAPSED   TIME: 00:11:37.2129
   15:58:07   LOGON 5908  ORIGINATING LSN: 196
                          USERCODE: WJW.
                          USERCODE PRIVILEGE: SECADMIN
                          MCS: 1
                          CHARGE CODE: 6842.
                          STATION NAME: T100B20.
                          SIGN ON BY NEW LOG-ON
   15:58:11   LOGON 5909  ORIGINATING LSN: 546
                          USERCODE: WJW.
                          USERCODE PRIVILEGE: SECADMIN
                          MCS: 2
                          CHARGE CODE: 6842.
                          STATION NAME: T100B20/CANDE/1.
                          SIGN ON BY NEW LOG-ON
   15:59:51   LOGON 5933  ORIGINATING LSN: 590
                          USERCODE: RAVENFORD.
                          MCS: 2
                          CHARGE CODE: 6715.
                          STATION NAME: TB146/ED/1.
                          SIGN ON BY NEW LOG-ON
   15:59:55   LGOFF 5933  ORIGINATING LSN: 590
                          USERCODE: RAVENFORD.
                          MCS: 2
                          CHARGE CODE: 6715.
                          STATION NAME: TB146/ED/1.
                          SIGN OFF BY NORMAL LOG-OFF
                          PROCESSOR TIME: 00:00:00.0479
                          I/O       TIME: 00:00:00.0604
                          ELAPSED   TIME: 00:00:04.4208
   16:02:24   LGOFF 3199  ORIGINATING LSN: 521
                          USERCODE: JULOQUE.
                          MCS: 2
                          CHARGE CODE: 6842.
                          STATION NAME: T100B10/CANDE/1.
                          SIGN OFF BY NORMAL LOG-OFF
                          PROCESSOR TIME: 00:00:00.0428
                          I/O       TIME: 00:00:00.0566
                          ELAPSED   TIME: 07:34:25.6405
   16:02:31   LGOFF 3198  ORIGINATING LSN: 193
                          USERCODE: JULOQUE.
```

Figure 7–18. MCS Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 11:59:36

REQUEST:  PRINTER NSP USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000786 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 10:55:49
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 25929 RECORDS FROM 04/04/1990 10:56:00 TO 04/04/1990 12:01:00

Wednesday, April 4, 1990
04/04/1990 10:58:11.0049 01 LINE   ABORT LSN=000328;
                                         RSLT BYTE INDEX=1; LINESTATUS=00;
                                         LAST FLAG=0A; ERROR FLAGS=100400;
                                         STATION/LINE NOT RDY,FORMAT ERROR/INPUT;
                                         ORIGINAL DCWRITE TYPE=035; VARIANT=000
04/04/1990 10:58:15.5510 01 LINE   ABORT LSN=000328;
                                         RSLT BYTE INDEX=1; LINESTATUS=00;
                                         LAST FLAG=0A; ERROR FLAGS=100400;
                                         STATION/LINE NOT RDY,FORMAT ERROR/INPUT;
                                         ORIGINAL DCWRITE TYPE=035; VARIANT=000
04/04/1990 11:00:14.0519 01 LINE   ABORT LSN=000328;
                                         RSLT BYTE INDEX=1; LINESTATUS=00;
                                         LAST FLAG=06; ERROR FLAGS=100040;
                                         STATION/LINE NOT RDY,VERTICAL PARITY(CHAR);
                                         ORIGINAL DCWRITE TYPE=035; VARIANT=000
04/04/1990 11:16:53.7449 08 MCS    INIT  MCS = *SYSTEM/COMS10 ON UPS.

NORMAL TERMINATION FOR LOGANALYZER PROGRAM.
```

Figure 7–19. NSP Option Output

```
LOGANALYZER VERSION: 38.82.192, SDASUPPORT VERSION: 38.83.77.
MCP *SYSTEM/ASD/MCP/OSI/38605A VERSION: 38.605 (MCP/AS)
ANALYZED BY A5 (SYSTEM SERIAL = 1234) ON DEC 13, 1989 06:30:35

REQUEST:  MIX 9203 DIAGNOSTICS (DSS (FTAM))

SUMLOG #001596 CREATED BY A5 (SYSTEM SERIAL = 1234) ON DEC 13, 1989 06:22:25
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 13243 RECORDS FROM DEC 13, 1989 06:22 TO DEC 13, 1989 06:31

DEC 13, 1989
12/13/89  06:25:55.261  9203  DSS   FTAM   --40866620: FILE XFER -- LOCAL VOLUME ATTRIBUTES
12/13/89  06:25:55.264  9203  DSS   FTAM   --40866720:             FAMILYNAME = DISK.
12/13/89  06:25:55.267  9203  DSS   FTAM   --40866780: FILE XFER -- REMOTE VOLUME ATTRIBUTES
12/13/89  06:25:55.270  9203  DSS   FTAM   --40866880:             USERCODE   = JASMITH/DOC.
12/13/89  06:25:55.273  9203  DSS   FTAM   --40866980:             YOURHOST   = MPA38C.
12/13/89  06:25:55.309  9203  DSS   FTAM   --40882240: FILE XFER -- SESSION = 0, NEXT ACTION = 7, LAST CONDITION = 624
12/13/89  06:25:55.498  9203  DSS   FTAM   --40882240: FILE XFER -- SESSION = 0, NEXT ACTION = 8, LAST CONDITION = 624
12/13/89  06:25:56.057  9203  DSS   FTAM   --48378026: PROPOSING APPLICATION CONTEXT = 1 0 8571 1 1.
12/13/89  06:25:56.061  9203  DSS   FTAM   --48378430: PROPOSING PRESENTATION CONTEXT = ISO FTAM PCI
12/13/89  06:25:56.069  9203  DSS   FTAM   --48378672:    CONTEXT-ID = 3
12/13/89  06:25:56.072  9203  DSS   FTAM   --48378682:    CONTEXT    = 1 0 8571 2 1.
12/13/89  06:25:56.136  9203  DSS   FTAM   --48378360: PROPOSING PRESENTATION CONTEXT = ISO FTAM FADU
12/13/89  06:25:56.142  9203  DSS   FTAM   --48378672:    CONTEXT-ID = 5
12/13/89  06:25:56.145  9203  DSS   FTAM   --48378682:    CONTEXT    = 1 0 8571 2 2.
12/13/89  06:25:56.171  9203  DSS   FTAM   --48378290: PROPOSING PRESENTATION CONTEXT = ISO UNSTRUCTURED TEXT
12/13/89  06:25:56.176  9203  DSS   FTAM   --48378672:    CONTEXT-ID = 7
12/13/89  06:25:56.179  9203  DSS   FTAM   --48378682:    CONTEXT    = 1 0 8571 2 3.
12/13/89  06:25:56.204  9203  DSS   FTAM   --48378220: PROPOSING PRESENTATION CONTEXT = ISO UNSTRUCTURED BINARY
12/13/89  06:25:56.209  9203  DSS   FTAM   --48378672:    CONTEXT-ID = 9
12/13/89  06:25:56.213  9203  DSS   FTAM   --48378682:    CONTEXT    = 1 0 8571 2 4.
12/13/89  06:25:56.238  9203  DSS   FTAM   --48378150: PROPOSING PRESENTATION CONTEXT = NBS FILE DIRECTORY
12/13/89  06:25:56.243  9203  DSS   FTAM   --48378672:    CONTEXT-ID = 11
12/13/89  06:25:56.307  9203  DSS   FTAM   --48378682:    CONTEXT    = 1 3 9999 1 2 2.
12/13/89  06:25:56.435  9203  DSS   FTAM   --55813450: WINSTON INITIATOR -- SESSION = 1, STATE = 1, EVENT = 408
12/13/89  06:25:56.535  9203  DSS   FTAM   --44606000: NEVILLE-INITIATOR -- SESSION = 1, STATE = 1, EVENT = 1008, STEP = 1
12/13/89  06:25:56.838  9203  DSS   FTAM   --41739500 (LENGTH = 158 OCTETS): TEXT TO BE SENT WITH THE OPEN STATEMENT

                                           [00000]  618030800201 03A080A08080 020780812C55 4E4953595320 412053455249 455320465441
                                           [00036]  4D2033382E36 30352E313139 3038204E4253 205068617365 328302037884 03053D008502
                                           [00072]  0580860100A7 80400528C278 0201400528C2 7B0202400528 C27B02034005 28C27B02044E
                                           [00108]  0528C27B0503 4E0528C27B05 024E0528C27B 050100005609 424541554348 414D50718019
                                           [00144]  034A4D420000 000000000000 0000

12/13/89  06:25:57.221  9203  DSS   FTAM   --55571940: THE STATE OF THE PORT FILE HAS CHANGED;  FILESTATE = 2 (OFFERED)
12/13/89  06:26:27.186  9203  DSS   FTAM   --55571940: THE STATE OF THE PORT FILE HAS CHANGED;  FILESTATE = 3 (OPENED)
12/13/89  06:26:27.351  9203  DSS   FTAM   --36469014: CHECKING APPLICATION CONTEXT = 1 0 8571 1 1.
12/13/89  06:26:27.357  9203  DSS   FTAM   --36472554: ACCEPTING PRESENTATION CONTEXT = ISO FTAM PCI
12/13/89  06:26:27.362  9203  DSS   FTAM   --36472674:    CONTEXT-ID = 3
12/13/89  06:26:27.365  9203  DSS   FTAM   --36472686:    CONTEXT    = 1 0 8571 2 1
12/13/89  06:26:27.370  9203  DSS   FTAM   --36472474: ACCEPTING PRESENTATION CONTEXT = ISO FTAM FADU
12/13/89  06:26:27.374  9203  DSS   FTAM   --36472674:    CONTEXT-ID = 5
12/13/89  06:26:27.377  9203  DSS   FTAM   --36472686:    CONTEXT    = 1 0 8571 2 2
12/13/89  06:26:27.382  9203  DSS   FTAM   --36472394: ACCEPTING PRESENTATION CONTEXT = ISO UNSTRUCTURED TEXT
12/13/89  06:26:27.387  9203  DSS   FTAM   --36472674:    CONTEXT-ID = 7
12/13/89  06:26:27.390  9203  DSS   FTAM   --36472686:    CONTEXT    = 1 0 8571 2 3
12/13/89  06:26:27.394  9203  DSS   FTAM   --36472314: ACCEPTING PRESENTATION CONTEXT = ISO UNSTRUCTURED BINARY
12/13/89  06:26:27.399  9203  DSS   FTAM   --36472674:    CONTEXT-ID = 9
12/13/89  06:26:27.402  9203  DSS   FTAM   --36472686:    CONTEXT    = 1 0 8571 2 4
```

Figure 7-20. DIAGNOSTICS (DSS) Option Output

```
LOGANALYZER VERSION: 38.83.207,  SDASUPPORT VERSION: 38.83.77.
MCP *SYSTEM/ASD/MCP/38207T VERSION: 38.83 (MCP/AS)
ANALYZED BY A6 (SYSTEM SERIAL = 470) ON DEC 15, 1989 19:25:45

REQUEST:  PRINTER 1725 TO 1726 DIAGNOSTICS (MHS) UNSORTED

SUMLOG #000181 CREATED BY A6 (SYSTEM SERIAL = 470) ON DEC 15, 1989 17:20:40
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 77709 RECORDS FROM DEC 15, 1989 17:20 TO DEC 15, 1989 19:26

DEC 15, 1989
12/15/89  17:25:01.375  9130  MHS   RTS    Adding request to queue - entry = 3  Queue Type = 0  Function Type = 7
                                            MTA Assoc = 2  Buffer = 1048575  Length = 0  Params = 1048575
12/15/89  17:25:01.383  9130  MHS   RTS    RT TRANSFER REQUEST over MTA assoc 2  for message (M)[I=94][U=330]
                                            Transfer Time = 10  RTS assoc = 1  Result = 0
12/15/89  17:25:01.400  9134  MHS   RTS   S ACTIVITY START REQUEST for BNA port [2] Result = 0 Connection = 45
                                           Activity Id = [02](1) 1
12/15/89  17:25:01.472  9135  MHS   RTS    Adding request to queue - entry = 1  Queue Type = 1  Function Type = 62
                                            MTA Assoc = 0  Buffer = 0  Length = 12  Params = 1
12/15/89  17:25:01.486  9136  MHS   RTS   Message from BNA port [2]   Connection = 45  RTS assoc = 1  MTA assoc = 0
                                           Activity Start Request Activity Id = [02](1) 1
12/15/89  17:25:01.564  9134  MHS   RTS   S DATA REQUEST for BNA port [2] Result = 0 Connection = 45
                                           Length = 1024 Text = '1234567890123456789012345678901234567890...'
12/15/89  17:25:01.599  9135  MHS   RTS    Adding request to queue - entry = 7  Queue Type = 1  Function Type = 0
                                            MTA Assoc = 0  Buffer = 0  Length = 1030  Params = 1
12/15/89  17:25:01.612  9136  MHS   RTS   Message from BNA port [2]   Connection = 45  RTS assoc = 1  MTA assoc = 0
                                           .Data - length = 1030
                                           '?????1234567890123456789012345678901234567890 1234'
12/15/89  17:25:01.667  9134  MHS   RTS   S SYNC MINOR REQUEST for BNA port [2] Result = 0 Connection = 45
                                           SyncPoint = 1
12/15/89  17:25:01.743  9135  MHS   RTS    Adding request to queue - entry = 1  Queue Type = 1  Function Type = 59
                                            MTA Assoc = 0  Buffer = 2  Length = 12  Params = 1
12/15/89  17:25:01.764  9134  MHS   RTS   S DATA REQUEST for BNA port [2] Result = 0 Connection = 45
                                           Length = 1024 Text = '5678901234567890123456789012345678901234...'
12/15/89  17:25:01.799  9135  MHS   RTS    Adding request to queue - entry = 4  Queue Type = 1  Function Type = 0
                                            MTA Assoc = 0  Buffer = 1  Length = 1030  Params = 2
12/15/89  17:25:01.810  9136  MHS   RTS   Message from BNA port [2]   Connection = 45  RTS assoc = 1  MTA assoc = 0
                                           Sync Minor Request  SyncPoint = 1
12/15/89  17:25:01.863  9134  MHS   RTS   S SYNC MINOR REQUEST for BNA port [2] Result = 0 Connection = 45
                                           SyncPoint = 2
12/15/89  17:25:01.910  9135  MHS   RTS    Adding request to queue - entry = 7  Queue Type = 1  Function Type = 59
                                            MTA Assoc = 0  Buffer = 2  Length = 12  Params = 1
12/15/89  17:25:01.917  9136  MHS   RTS   S SYNC MINOR RESPONSE for BNA port [2] Result = 0 Connection = 45
                                           SyncPoint = 1
12/15/89  17:25:01.989  9136  MHS   RTS   Message from BNA port [2]   Connection = 45  RTS assoc = 1  MTA assoc = 0.
                                           .Data - length = 1030
                                           '?????5678901234567890123456789012345678901 2345678'
12/15/89  17:25:02.046  9131  MHS   RTS    Adding request to queue - entry = 2  Queue Type = 1  Function Type = 60
                                            MTA Assoc = 2  Buffer = 3  Length = 12  Params = 1
12/15/89  17:25:02.054  9134  MHS   RTS   S DATA REQUEST for BNA port [2] Result = 0 Connection = 45
                                           Length = 1024 Text = '9012345678901234567890123456789012345678...'
12/15/89  17:25:02.128  9135  MHS   RTS    Adding request to queue - entry = 1  Queue Type = 1  Function Type = 0
                                            MTA Assoc = 0  Buffer = 3  Length = 1030  Params = 2
12/15/89  17:25:02.136  9136  MHS   RTS   Message from BNA port [2]   Connection = 45  RTS assoc = 1  MTA assoc = 0
                                           Sync Minor Request  SyncPoint = 2
12/15/89  17:25:02.191  9134  MHS   RTS   S SYNC MINOR REQUEST for BNA port [2] Result = 0 Connection = 45
                                           SyncPoint = 3
12/15/89  17:25:02.236  9135  MHS   RTS    Adding request to queue - entry = 4  Queue Type = 1  Function Type = 59
                                            MTA Assoc = 0  Buffer = 1  Length = 12  Params = 1
```

Figure 7–21. DIAGNOSTICS (MHS) Option Output

```
LOGANALYZER VERSION: 39.609.706,  SDASUPPORT VERSION: 39.25.41.
MCP *SYSTEM/ASD/MCP/39026/DIAGNOSTICS VERSION: 39.26 (MCP/AS)
ANALYZED BY A3 (SYSTEM SERIAL = 2356) ON JUL 11, 1990 15:16:03

REQUEST:  P DIAGNOSTICS (NIS) UNSORTED

SUMLOG #001622 CREATED BY A3 (SYSTEM SERIAL = 2356) ON JUL 11, 1990 15:03:30
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 455 RECORDS FROM JUL 11, 1990 15:04 TO JUL 11, 1990 15:17

JUL 11, 1990
15:10:43.1378  DIAG 1907  NIS NS          20004070 - NS SERVANT SENDMESSAGE MSG LEN IS 54
15:10:43.2808  DIAG 1907  NIS NS          82246000 - PARSE COMMAND - COMMAND PARSED SUCCESSFULLY
15:10:43.3356  DIAG 1907  NIS TOKEN       AFTER SHIFT, BUFFER IS: ""
15:10:43.3703  DIAG 1907  NIS TOKEN       TOKEN_GET_NEXT - FOUND TOKEN: N CLASS = 8 TOKEN = 8 VALUE = -1 COLUMN
= 0 CHAR LEN = 1 CHARS REM = -1
15:10:43.3834  DIAG 1907  NIS NS          82248400 - PARSE COMMAND END
15:10:55.4782  DIAG 1907  NIS NS          45161000 - NETWORK SUPPORT - INPUT EVENT OCCURRED
15:10:55.4915  DIAG 1907  NIS NS          40570000 - FINDNWMESSAGE - BEGIN
15:10:55.5087  DIAG 1907  NIS NS          40237000 - REMOVE_PACONTROLINFO - BEGIN
15:10:55.5222  DIAG 1907  NIS NS          40408000 - REMOVE_PACONTROLINFO - REPORT
15:10:55.5361  DIAG 1907  NIS NS          40514000 - REMOVE_PACONTROLINFO - END
15:10:55.5505  DIAG 1907  NIS NS          40616000 - FINDNWMESSAGE - END

NORMAL TERMINATION FOR LOGANALYZER PROGRAM.
```

Figure 7–22. DIAGNOSTICS (NIS) Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 17:34:04

REQUEST:  PRINTER ABORT USERCODE .

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 28804 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 17:35:00

Tuesday, April 3, 1990
   17:29:44    EI    3193   *OBJECT/ED ON MCPMAST.
                            STACK NUMBER: 0AFF
                            TASK TYPE: DEPENDENT TASK (PROCESS)
                            USERCODE: POLTON.
                            CHARGECODE: 6692.
                            MCS NUMBER: 2  LSN NUMBER: 519
                            MCS NAME: SYSTEM/CANDE.
                            TIMESTAMP: 08:25:38
   17:29:44    CLOSE 3193   EXT NAME: REM.
                            INT NAME: REM.
                            FILE ACCESS RULE = DECLARER    (ACTOR = DECLARER = STACK AFF, JOB 3191, TASK 3193)
                            IO TIME        : 00:00:00.0000
                            CLOSE TYPE: NORMAL    ASSOCIATION: RELEASE   DISPOSITION: REWIND
                            TRANSACTION COUNT = 40         PHYSICAL (READ COUNT=0, WRITE COUNT=0)
                            BUFFER CATEGORY: BUFFERGOAL NOT APPLICABLE    BUFFERSIZE: 0
                            BLOCKSIZE      = 2560          MIN/MAXRECSIZE     = 0/ 2560        INTMODE = EBCDIC
   17:29:44    CLOSE 3193   EXT NAME: *XREFFILES/SYSTEM/PRINT/SUPPORT/DECS.
                            INT NAME: DECLARATIONS.
                            FAMILY NAME: SYS37
                            FILE ACCESS RULE = DECLARER    (ACTOR = DECLARER = STACK AFF, JOB 3191, TASK 3193)
                            IO TIME        : 00:00:00.5551
                            CLOSE TYPE: NORMAL    ASSOCIATION: RELEASE   DISPOSITION: BLOCKEXIT
                            TRANSACTION COUNT = 92         PHYSICAL (READ COUNT=45, WRITE COUNT=0)
                            FILE STRUCTURE: ALIGNED180
                            BUFFER CATEGORY: BUFFERGOAL NOT APPLICABLE    BUFFERSIZE: 510
                            BLOCKSIZE      = 510           MIN/MAXRECSIZE     = 0/ 17          INTMODE = SINGLE
                            PERM   SPEED=00   AREASIZE 255 SECTORS   FILE SIZE 3808 SECTORS
                            SERIAL NO 682246   CREATION DATE 89234 CYCLE 1 VERSION 0 SAVEFACTOR 0
   17:29:44    CLOSE 3193   EXT NAME: *XREFFILES/SYSTEM/PRINT/SUPPORT/REFS.
                            INT NAME: REFERENCES.
                            FAMILY NAME: SYS37
                            FILE ACCESS RULE = DECLARER    (ACTOR = DECLARER = STACK AFF, JOB 3191, TASK 3193)
                            IO TIME        : 00:00:00.1864
                            CLOSE TYPE: NORMAL    ASSOCIATION: RELEASE   DISPOSITION: BLOCKEXIT
                            TRANSACTION COUNT = 29         PHYSICAL (READ COUNT=23, WRITE COUNT=0)
                            FILE STRUCTURE: ALIGNED180
                            BUFFER CATEGORY: BUFFERGOAL NOT APPLICABLE    BUFFERSIZE: 210
                            BLOCKSIZE      = 210           MIN/MAXRECSIZE     = 0/ 210         INTMODE = SINGLE
                            PERM   SPEED=00   AREASIZE 210 SECTORS   FILE SIZE 1820 SECTORS
                            SERIAL NO 682246   CREATION DATE 89234 CYCLE 1 VERSION 0 SAVEFACTOR 0
   17:29:44    CLOSE 3193   EXT NAME: *SYMBOL/PRINT/SUPPORT.
                            INT NAME: SYMBOL.
                            FAMILY NAME: SYS38
                            FILE ACCESS RULE = DECLARER    (ACTOR = DECLARER = STACK AFF, JOB 3191, TASK 3193)
                            IO TIME        : 00:00:00.8348
                            CLOSE TYPE: NORMAL    ASSOCIATION: RELEASE   DISPOSITION: BLOCKEXIT
                            TRANSACTION COUNT = 296        PHYSICAL (READ COUNT=43, WRITE COUNT=0)
                            FILE STRUCTURE: ALIGNED180
```

Figure 7–23. ABORT Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 17:40:21

REQUEST:  PRINTER 1500 TO 1630 BOJ USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 30049 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 17:41:00

Tuesday, April 3, 1990
   15:53:36    BOJ  5875   "UPDATE FAULTLOG".
                           STACK NUMBER: 03D8
                           QUEUE: 0
                           ORIGINATING UNIT: 0
                           PRIORITY: 80
                           SCHEDULED FOR: 00:00:08.0400
   15:53:37    BOJ  5878   "BEGIN JOB;RUN SYS".
                           STACK NUMBER: 03DE
                           JOB ENTERED SYSTEM: 04/03/1990 15:53:37 FROM WFL 39.23
                           QUEUE: 0
                           ORIGINATING UNIT: 0
                           PRIORITY: 50
   15:54:19    BOJ  5807   HOSTCOMP/39.
                           STACK NUMBER: 03E6
                           JOB ENTERED SYSTEM: 04/03/1990 15:43:20 FROM WFL 39.23
                           QUEUE: 30
                           ORIGINATING LSN: 562 MCS: 2
                           PRIORITY: 40
                           USERCODE: ALESHA.
                           CHARGECODE: 6893.
                           SOURCENAME: SFA6AB/TB117/CANDE/1/CANDE/1.
                           INITIATING MCS: SYSTEM/CANDE.
   15:54:19    BOJ  5883   SERVER/LP5.
                           STACK NUMBER: 03EA
                           QUEUE: 0
                           ORIGINATING UNIT: 0
                           PRIORITY: 80
   15:54:39    BOJ  5886   *SYSTEM/IGSDASUPPORT.
                           STACK NUMBER: 03F0
                           CODE COMPILED: 03/19/1990 22:43:21 BY DCALGOL 39.23
                           PU PRIVILEGED PROGRAM
                           QUEUE: 0
                           ORIGINATING LSN: 560 MCS: 2
                           PRIORITY: 50
                           USERCODE: DBH.
                           CHARGECODE: 6685.
                           SOURCENAME: TB131S1/CANDE/1.
                           INITIATING MCS: SYSTEM/CANDE.
   15:54:39    BOJ  5887   (DOBB)OBJECT/UTIL/RUNPMTR ON PACK.
                           STACK NUMBER: 03F1
                           CODE COMPILED: 03/23/1990 14:47:23 BY ALGOL 38.93
                           QUEUE: 0
                           ORIGINATING LSN: 295 MCS: 1
                           PRIORITY: 50
                           USERCODE: JEFF.
                           CHARGECODE: 6381.
                           SOURCENAME: TB137.
                           INITIATING MCS: *SYSTEM/COMS.
```

Figure 7-24. BOJ Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 17:44:42

REQUEST:  PRINTER 1500 TO 1630 BOT USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 30865 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 17:46:00

Tuesday, April 3, 1990
   15:53:36    BOJ  5875   "UPDATE FAULTLOG".
                           STACK NUMBER: 03D8
                           QUEUE: 0
                           ORIGINATING UNIT: 0
                           PRIORITY: 80
                           SCHEDULED FOR: 00:00:08.0400
   15:53:37    BOJ  5878   "BEGIN JOB;RUN SYS".
                           STACK NUMBER: 03DE
                           JOB ENTERED SYSTEM: 04/03/1990 15:53:37 FROM WFL 39.23
                           QUEUE: 0
                           ORIGINATING UNIT: 0
                           PRIORITY: 50
   15:53:37    BOT  5879   *SYSTEM/FAULTLOGGER.
                           STACK NUMBER: 03E0
                           CODE COMPILED: 03/20/1990 13:55:08 BY DCALGOL 39.23
                           TASK TYPE: COROUTINE(CALL)
                           PU PRIVILEGED PROGRAM
                           PRIORITY: 50
   15:54:17    BOT  5880   SYSTEM/STAXFER/DIALOGINIT/16.
                           STACK NUMBER: 03E3
                           TASK TYPE: DEPENDENT TASK(PROCESS)
                           PRIORITY: 75
   15:54:18    BOT  5881   *OBJECT/WFLMSG ON MCPMAST.
                           STACK NUMBER: 03E4
                           CODE COMPILED: 06/04/1989 23:56:12 BY DCALGOL 38.73
                           TASK TYPE: COROUTINE(CALL)
                           PRIORITY: 40
                           USERCODE: FUJIYAMA.
                           CHARGECODE: 6715.
   15:54:19    BOJ  5807   HOSTCOMP/39.
                           STACK NUMBER: 03E6
                           JOB ENTERED SYSTEM: 04/03/1990 15:43:20 FROM WFL 39.23
                           QUEUE: 30
                           ORIGINATING LSN: 562 MCS: 2
                           PRIORITY: 40
                           USERCODE: NARAYANA.
                           CHARGECODE: 6893.
                           SOURCENAME: SFA3AB/TB117/CANDE/1/CANDE/1.
                           INITIATING MCS: SYSTEM/CANDE.
   15:54:19    BOT  5882   *SYSTEM/PATCH ON SYS39.
                           STACK NUMBER: 03E8
                           CODE COMPILED: 03/19/1990 16:52:49 BY ALGOL 39.23
                           TASK TYPE: COROUTINE(CALL)
                           PRIORITY: 40
                           USERCODE: NARAYANA.
                           CHARGECODE: 6893.
                           SOURCENAME: SFA3AB/TB117/CANDE/1/CANDE/1.
                           INITIATING MCS: SYSTEM/CANDE.
```

Figure 7–25. BOT Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.037,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 822) ON 04/05/1990 10:23:33

REQUEST:  PRINTER DMS USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #002710 CREATED BY A15 (SYSTEM SERIAL = 822) ON 04/05/1990 07:00:41
TITLE = *SYSTEM/SUMLOG ON OPERATIONS (CURRENT SUMLOG).

FILE CONTAINS 58258 RECORDS FROM 04/05/1990 07:01:00 TO 04/05/1990 10:25:00

Thursday, April 5, 1990
   08:58:40    DMS  2557   (JACQUE)EBBDB.  (STACK 1CD)
                           DATABASE FROZEN.
                           ACR NAME : *SYSTEM/ACCESSROUTINES.  (STACK 2FB)
   08:58:42    DMS  2519   *SYSTEM/SIM/DMSIISUPPORT ON SYS39.
                           OPEN DATABASE.
                           USERCODE        : WBW.
                           DATABASE NAME   : (JACQUE)EBBDB.  (2557/2557,  STACK 1CD)
                           INTERNAL NAME   : DB.
                           OPEN  TYPE      : 10
                           SIB LEVEL       : 08  ,      ACR LEVEL         : 060
                           UPDATE LEVEL    : 004 ,      LOGICAL DB NUMBER : 000
   08:58:43    DMS  2521   (IQF)EBBDB/1/2500.
                           OPEN DATABASE.
                           USERCODE        : WBW.
                           DATABASE NAME   : (JACQUE)EBBDB.  (2557/2557,  STACK 1CD)
                           INTERNAL NAME   : DB.
                           OPEN  TYPE      : 12
                           SIB LEVEL       : 08  ,      ACR LEVEL         : 060
                           UPDATE LEVEL    : 004 ,      LOGICAL DB NUMBER : 000
   08:58:49    DMS  2513   *SYSTEM/SIM/DMSIISUPPORT ON SYS39.
                           OPEN DATABASE.
                           USERCODE        : JACQUE.
                           DATABASE NAME   : (JACQUE)EBBDB.  (2557/2557,  STACK 1CD)
                           INTERNAL NAME   : DB.
                           OPEN  TYPE      : 10
                           SIB LEVEL       : 08  ,      ACR LEVEL         : 060
                           UPDATE LEVEL    : 004 ,      LOGICAL DB NUMBER : 000
                           OPEN  FAILED : ERROR #64
   08:58:50    DMS  2517   (JACQUE)EBBDB/1/2480.
                           OPEN DATABASE.
                           USERCODE        : JACQUE.
                           DATABASE NAME   : (JACQUE)EBBDB.  (2557/2557,  STACK 1CD)
                           INTERNAL NAME   : DB.
                           OPEN  TYPE      : 12
                           SIB LEVEL       : 08  ,      ACR LEVEL         : 060
                           UPDATE LEVEL    : 004 ,      LOGICAL DB NUMBER : 000
   09:01:50    DMS  2625   (QF)X.   (STACK 225)
                           DATABASE FROZEN.
                           ACR NAME : (QF)SYSTEM/ACCESSROUTINES.  (STACK 226)
   09:02:09    DMS  2391   (WATKINS)ADDSDB/2/2385.
                           CLOSE DATABASE.
                           USERCODE        : WATKINS.
                           DATABASE NAME   : (ADDSPATCH)ADDSDB.  (2218/2218,  STACK EFB)
                           CLOSE TYPE      : 0
   09:02:11    DMS  2624   (QF)OBJECT/UP/X/DKTEST ON DMSII.
                           CLOSE DATABASE.
                           USERCODE        : QF.
                           DATABASE NAME   : (QF)X.  (2625/2625,  STACK 225)
```

Figure 7–26. DMS Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 17:59:10

REQUEST:  PRINTER 1500 TO 1630 EOJ USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 33029 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 18:00:00

Tuesday, April 3, 1990
   15:53:37    EOJ  5875   "UPDATE FAULTLOG".
                           STACK NUMBER: 03D8                AVERAGE MEMORY USAGE: CODE=0, DATA=740
                           PROCESSOR TIME: 00:00:00.0211     MEMORY INTEGRAL: CODE=0.000, DATA=0.051
                           I/O       TIME: 00:00:00.0479     INITIAL PBITS: 11.
                           READYQ    TIME: 00:00:00.0390     MAXIMUM NUMBER OF ASDS USED: 5.
                           INITPBIT  TIME: 00:00:00.0035     MAXIMUM SAVE MEMORY USED: 696.
                           ELAPSED   TIME: 00:00:00.1165

   15:53:38    EOJ  5878   "BEGIN JOB;RUN SYS".
                           STACK NUMBER: 03DE                AVERAGE MEMORY USAGE: CODE=77, DATA=1086
                           PROCESSOR TIME: 00:00:00.0347     MEMORY INTEGRAL: CODE=0.009, DATA=0.132
                           I/O       TIME: 00:00:00.0865     INITIAL PBITS: 16, OTHER PBITS: 1.
                           READYQ    TIME: 00:00:00.0074     MAXIMUM NUMBER OF ASDS USED: 5.
                           INITPBIT  TIME: 00:00:00.0087     MAXIMUM SAVE MEMORY USED: 2127.
                           OTHERPBIT TIME: 00:00:00.0013
                           ELAPSED   TIME: 00:00:00.8176

   15:54:18    EOJ  5663   SASHA373MCP.
                           STACK NUMBER: 022E                USERCODE: SASHA.
                           PROCESSOR TIME: 00:00:00.4897     CHARGECODE: 6715.
                           I/O       TIME: 00:00:01.1173     AVERAGE MEMORY USAGE: CODE=731, DATA=2465
                           READYQ    TIME: 00:00:00.1540     MEMORY INTEGRAL: CODE=1.174, DATA=3.961
                           INITPBIT  TIME: 00:00:00.0874     INITIAL PBITS: 229, OTHER PBITS: 2.
                           OTHERPBIT TIME: 00:00:00.0020     MAXIMUM NUMBER OF ASDS USED: 35.
                           ELAPSED   TIME: 00:27:39.2064     MAXIMUM SAVE MEMORY USED: 2248.

   15:54:38    EOJ  5846   (SABE)OBJECT/TRANSFORMS ON MCPMAST.
                           STACK NUMBER: 039B                AVERAGE MEMORY USAGE: CODE=3179, DATA=1310
                           PROCESSOR TIME: 00:00:00.0385     MEMORY INTEGRAL: CODE=0.282, DATA=0.116
                           I/O       TIME: 00:00:00.0501     INITIAL PBITS: 26.
                           READYQ    TIME: 00:00:00.0376     MAXIMUM NUMBER OF ASDS USED: 15.
                           INITPBIT  TIME: 00:00:00.0133     MAXIMUM SAVE MEMORY USED: 1009.
                           ELAPSED   TIME: 00:05:24.6090

   15:55:01    EOJ  5886   *SYSTEM/IGSDASUPPORT.
                           STACK NUMBER: 03F0                AVERAGE DISK SECTORS IN USE BY PERMANENT FILES: 1089.
                           PROCESSOR TIME: 00:00:00.1213     USERCODE: POW.
                           I/O       TIME: 00:00:00.8821     CHARGECODE: 6685.
                           READYQ    TIME: 00:00:00.0330     AVERAGE MEMORY USAGE: CODE=2662, DATA=25799
                           INITPBIT  TIME: 00:00:00.0177     MEMORY INTEGRAL: CODE=2.671, DATA=25.888
                           ELAPSED   TIME: 00:00:21.5453     INITIAL PBITS: 51.
                                                             MAXIMUM NUMBER OF ASDS USED: 40.
                                                             MAXIMUM SAVE MEMORY USED: 31825.
```

Figure 7–27. EOJ Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 18:00:55

REQUEST:  PRINTER 1500 TO 1630 EOT USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 33333 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 18:02:00

Tuesday, April 3, 1990
   15:53:37     EOJ  5875  "UPDATE FAULTLOG".
                           STACK NUMBER: 03DB              AVERAGE MEMORY USAGE: CODE=0, DATA=740
                           PROCESSOR TIME: 00:00:00.0211   MEMORY INTEGRAL: CODE=0.000, DATA=0.051
                           I/O       TIME: 00:00:00.0479   INITIAL PBITS: 11.
                           READYQ    TIME: 00:00:00.0390   MAXIMUM NUMBER OF ASDS USED: 5.
                           INITPBIT  TIME: 00:00:00.0035   MAXIMUM SAVE MEMORY USED: 696.
                           ELAPSED   TIME: 00:00:00.1165

   15:53:38     EOT  5879  *SYSTEM/FAULTLOGGER.
                           STACK NUMBER: 03E0              AVERAGE MEMORY USAGE: CODE=3221, DATA=2680
                           PROCESSOR TIME: 00:00:00.0446   MEMORY INTEGRAL: CODE=0.420, DATA=0.350
                           I/O       TIME: 00:00:00.0858   INITIAL PBITS: 34, OTHER PBITS: 4.
                           READYQ    TIME: 00:00:00.0107   MAXIMUM NUMBER OF ASDS USED: 15.
                           INITPBIT  TIME: 00:00:00.0210   MAXIMUM SAVE MEMORY USED: 3119.
                           OTHERPBIT TIME: 00:00:00.0047
                           ELAPSED   TIME: 00:00:00.6122

   15:53:38     EOJ  5878  "BEGIN JOB;RUN SYS".
                           STACK NUMBER: 03DE              AVERAGE MEMORY USAGE: CODE=77, DATA=1086
                           PROCESSOR TIME: 00:00:00.0347   MEMORY INTEGRAL: CODE=0.009, DATA=0.132
                           I/O       TIME: 00:00:00.0865   INITIAL PBITS: 16, OTHER PBITS: 1.
                           READYQ    TIME: 00:00:00.0074   MAXIMUM NUMBER OF ASDS USED: 5.
                           INITPBIT  TIME: 00:00:00.0087   MAXIMUM SAVE MEMORY USED: 2127.
                           OTHERPBIT TIME: 00:00:00.0013
                           ELAPSED   TIME: 00:00:00.8176

   15:53:42     EOT  5873  *SYSTEM/DCALGOL ON SYS39.
                           STACK NUMBER: 03D7              AVERAGE DISK SECTORS IN USE BY PERMANENT FILES: 4447.
                           PROCESSOR TIME: 00:00:06.5326   USERCODE: PDW.
                           I/O       TIME: 00:00:04.3208   CHARGECODE: 6685.
                           READYQ    TIME: 00:00:00.5065   AVERAGE MEMORY USAGE: CODE=40198, DATA=47834
                           INITPBIT  TIME: 00:00:00.5068   MEMORY INTEGRAL: CODE=436.290, DATA=519.171
                           ELAPSED   TIME: 00:00:14.7170   INITIAL PBITS: 675.
                                                           MAXIMUM NUMBER OF ASDS USED: 300.
                                                           MAXIMUM SAVE MEMORY USED: 22557.

   15:53:48     EOT  5864  *OBJECT/ED ON MCPMAST.
                           STACK NUMBER: 03BE              AVERAGE DISK SECTORS IN USE BY PERMANENT FILES: 7871.
                           PROCESSOR TIME: 00:00:00.9052   USERCODE: PDW.
                           I/O       TIME: 00:00:03.0904   CHARGECODE: 6685.
                           READYQ    TIME: 00:00:00.3095   AVERAGE MEMORY USAGE: CODE=27023, DATA=17107
                           INITPBIT  TIME: 00:00:00.1244   MEMORY INTEGRAL: CODE=107.975, DATA=68.352
                           OTHERPBIT TIME: 00:00:00.0131   INITIAL PBITS: 404, OTHER PBITS: 9.
                           ELAPSED   TIME: 00:02:15.8809   MAXIMUM NUMBER OF ASDS USED: 176.
                                                           MAXIMUM SAVE MEMORY USED: 17226.
```

Figure 7–28. EOT Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 18:03:17

REQUEST:  PRINTER 1600 TO 1630 ERRORS USERCODE .

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 33672 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 18:04:00

Tuesday, April 3, 1990
   16:00:08     BOT  5941   *SYSTEM/CC.
                            STACK NUMBER: 0481
                            CODE COMPILED: 03/20/1990 07:10:36 BY PASCAL 39.15
                            TASK TYPE: DEPENDENT TASK(PROCESS)
                            PRIORITY: 50
                            CODEFILE: (JASMITH)CANDE/CODE6130 ON DISK.
                            USERCODE: JASMITH.
                            CHARGECODE: 6671.
                            SOURCENAME: TB263/E/1.
                            INITIATING MCS: SYSTEM/CANDE.
   16:00:09     OPEN 5941   EXT NAME: (JASMITH)EDITOR/TEMP/SPIDER/STREMUL/MACROS/H.
                            INT NAME: CARD.
                            FAMILY NAME: FIRE
                            FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK 481, JOB 4404, TASK 5941)
                            USE = CLOSED KIND=PACK   FILEKIND= CCSYMBOL     FILE SIZE 15 SEGS.  UNIT NUMBER 63
                            OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
   16:00:09     OPEN 5941   EXT NAME: (JASMITH)SPIDER/STREMUL/MACROS/H.
                            INT NAME: SOURCE.
                            FAMILY NAME: FIRE
                            FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK 481, JOB 4404, TASK 5941)
                            USE = CLOSED KIND=PACK   FILEKIND= CCSYMBOL     FILE SIZE 30 SEGS.  UNIT NUMBER 63
                            OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
   16:00:15     OPEN 5941   EXT NAME: (JASMITH)ERRORS/SPIDER/STREMUL/MACROS/H.
                            INT NAME: ERRORS.
                            FAMILY NAME: FIRE
                            FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK 481, JOB 4404, TASK 5941)
                            USE = CLOSED KIND=PACK   FILEKIND= DATA     FILE SIZE 0 SEGS.  UNIT NUMBER 63
                            OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
   16:00:16    CLOSE 5941   EXT NAME: (JASMITH)ERRORS/SPIDER/STREMUL/MACROS/H.
                            INT NAME: ERRORS.
                            FAMILY NAME: FIRE
                            FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK 481, JOB 4404, TASK 5941)
                            IO TIME      : 00:00:00.3538
                            CLOSE TYPE: NORMAL   ASSOCIATION: RELEASE   DISPOSITION: SAVE
                            TRANSACTION COUNT = 19         PHYSICAL (READ COUNT=0, WRITE COUNT=19)
                            FILE STRUCTURE: ALIGNED180
                            BUFFER CATEGORY: BUFFERGOAL NOT APPLICABLE    BUFFERSIZE: 14
                            BLOCKSIZE      = 80           MIN/MAXRECSIZE     = 0/ 80        INTMODE = EBCDIC
                            PERM   SPEED=00   UPDATED  AREASIZE 90 SECTORS   FILE SIZE 90 SECTORS
                            SERIAL NO 659063   CREATION DATE 90093 CYCLE 1 VERSION 0 SAVEFACTOR 0
   16:00:16    CLOSE 5941   EXT NAME: (JASMITH)SPIDER/STREMUL/MACROS/H.
                            INT NAME: SOURCE.
                            FAMILY NAME: FIRE
                            FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK 481, JOB 4404, TASK 5941)
                            IO TIME      : 00:00:00.0773
                            CLOSE TYPE: NORMAL   ASSOCIATION: RELEASE   DISPOSITION: BLOCKEXIT
                            TRANSACTION COUNT = 55         PHYSICAL (READ COUNT=3, WRITE COUNT=0)
                            FILE STRUCTURE: ALIGNED180
```

Figure 7–29. ERRORS Option Output

```
LOGANALYZER VERSION: 39.036.059,  SDASUPPORT VERSION: 39.036.044,  JOBFORMATTER VERSION: 39.036.432.
MCP *SYSTEM/MCP/39036D VERSION: 39.036 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 503) ON 10/08/1990 13:53:08

REQUEST:  PRINTER 1318 TO 1320 FILE (UNSORTED BY DEFAULT)

SUMLOG #001439 CREATED BY A9 (SYSTEM SERIAL = 3) ON 06/15/1990 13:17:54
TITLE = (JAHAR)LASTLOG ON PACKA.

FILE CONTAINS 721 RECORDS FROM 06/15/1990 13:18:00 TO 06/15/1990 13:36:00


Friday, June 15, 1990
   13:18:27    OPEN 6513   EXT NAME: (JAHAR)A1.
                           INT NAME: CARD.
                           FILE ACCESS RULE = DECLARER (ACTOR = DECLARER = STACK 08E, JOB 6502, TASK 6513)
                           FAMILYNAME MCPTESTS, KIND=PACK UNITNO 48, FILEKIND=ALGOLSYMBOL  FILE SIZE 14 SECTORS
                           USE = CLOSED, OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
   13:18:28    OPEN 6513   EXT NAME: (JAHAR)CANDE/CODE1830.
                           INT NAME: CODE.
                           FILE ACCESS RULE = DECLARER (ACTOR = DECLARER = STACK 08E, JOB 6502, TASK 6513)
                           FAMILYNAME MCPTESTS, KIND=PACK UNITNO 48, FILEKIND=DATA  FILE SIZE 0 SECTORS
                           USE = CLOSED, OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
   13:18:30   CLOSE 6513   EXT NAME: (JAHAR)CANDE/CODE1830.
                           INT NAME: CODE.
                           FAMILY NAME: MCPTESTS
                           FILE ACCESS RULE = DECLARER (ACTOR = DECLARER = STACK 08E, JOB 6502, TASK 6513)
                           CLOSE TYPE: NORMAL  ASSOCIATION: RELEASE  DISPOSITION: CRUNCH
                           IO TIME: 00:00:00.0723 TRANSACTION COUNT = 17 PHYSICAL (READ COUNT=0, WRITE COUNT=3)
                           FILESTRUCTURE = ALIGNED180, BUFFER SIZE SOURCE = BLOCKSIZE, BUFFERSIZE = 270
                           FRAMESIZE = 48, MIN/MAXRECSIZE = 0/ 30, BLOCKSIZE = 270
                           INTMODE = SINGLE                EXTMODE = SINGLE
                           UPDATED  CRUNCHED   AREASIZE 504 SECTORS, FILE SIZE 18 SECTORS
                           SERIAL NO 659048, CREATION DATE 90166 CYCLE 1 VERSION 0 SAVEFACTOR 999
   13:18:30   CLOSE 6513   EXT NAME: (JAHAR)A1.
                           INT NAME: CARD.
                           FAMILY NAME: MCPTESTS
                           FILE ACCESS RULE = DECLARER (ACTOR = DECLARER = STACK 08E, JOB 6502, TASK 6513)
                           CLOSE TYPE: NORMAL  ASSOCIATION: RELEASE  DISPOSITION: BLOCKEXIT
                           IO TIME: 00:00:00.0283 TRANSACTION COUNT = 19 PHYSICAL (READ COUNT=1, WRITE COUNT=0)
                           FILESTRUCTURE = ALIGNED180, BUFFER SIZE SOURCE = BLOCKSIZE, BUFFERSIZE = 420
                           FRAMESIZE = 48, MIN/MAXRECSIZE = 0/ 15, BLOCKSIZE = 420
                           INTMODE = EBCDIC                EXTMODE = EBCDIC
                           PERM   SPEED=00   AREASIZE 504 SECTORS, FILE SIZE 14 SECTORS
                           SERIAL NO 659048, CREATION DATE 90151 CYCLE 1 VERSION 0 SAVEFACTOR 30
   13:24:54    OPEN 6526   EXT NAME: F4.
                           INT NAME: F4.
                           FILE ACCESS RULE = DECLARER (ACTOR = DECLARER = STACK 0AA, JOB 6502, TASK 6526)
                           KIND=TAPE UNITNO 162, LABELKIND= B6500 USASI
                           USE = CLOSED, OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
   13:25:25   CLOSE 6526   EXT NAME: F4.
                           INT NAME: F4.
                           FILE ACCESS RULE = DECLARER (ACTOR = DECLARER = STACK 0AA, JOB 6502, TASK 6526)
                           UNITNO 162 DENSITY BPI1600
                           CLOSE TYPE: NORMAL  ASSOCIATION: RELEASE  DISPOSITION: BLOCKEXIT
                           IO TIME: 00:00:00.0046 TRANSACTION COUNT = 1 PHYSICAL (READ COUNT=0, WRITE COUNT=1)
                           BUFFER SIZE SOURCE = BLOCKSIZE, BUFFERSIZE = 10
                           FRAMESIZE = 48, MIN/MAXRECSIZE = 0/ 10, BLOCKSIZE = 10
                           INTMODE = EBCDIC                EXTMODE = EBCDIC
                           SERIAL NO JJAHAR, CREATION DATE 90166 CYCLE 1 VERSION 0 SAVEFACTOR 0 REEL 1
```

Figure 7–30. FILE Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 18:11:01

REQUEST:  PRINTER JOB 6619 USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 34705 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 18:12:00

Tuesday, April 3, 1990
   17:59:13    BOJ  6619  SI/WFLJOB/"900403-175911".
                          STACK NUMBER: 07F8
                          JOB ENTERED SYSTEM: 04/03/1990 17:59:13 FROM WFL 39.23
                          QUEUE: 5
                          ORIGINATING LSN: 183 MCS: 1
                          PRIORITY: 50
                          USERCODE: ALAJUA.
                          CHARGECODE: 6842.
                          SOURCENAME: TMPA15C.
                          INITIATING MCS: *SYSTEM/COMS.
   17:59:13    BOT  6620  (ALAJUA)S001.
                          STACK NUMBER: 07FA
                          TASK TYPE: DEPENDENT TASK(PROCESS)
                          PRIORITY: 50
                          USERCODE: ALAJUA.
                          CHARGECODE: 6842.
                          SOURCENAME: TMPA15C.
                          INITIATING MCS: *SYSTEM/COMS.
   17:59:13    BOT  6621  (ALAJUA)S002.
                          STACK NUMBER: 07FB
                          TASK TYPE: DEPENDENT TASK(PROCESS)
                          PRIORITY: 50
                          USERCODE: ALAJUA.
                          CHARGECODE: 6842.
                          SOURCENAME: TMPA15C.
                          INITIATING MCS: *SYSTEM/COMS.
   17:59:14         6620  MSRDISP3: DISPLAY:PROTOCOLS (TAPE)   ->  DISK (PACK,FAMILYINDEX=1) .
   17:59:14         6621  MSRDISP3: DISPLAY:PROTOCOLS (TAPE)   ->  DISK (PACK) .
   17:59:14    BOT  6622  *LIBRARY/MAINTENANCE.
                          STACK NUMBER: 07FC
                          TASK TYPE: COROUTINE(CALL)
                          PRIORITY: 50
                          USERCODE: ALAJUA.
                          CHARGECODE: 6842.
                          SOURCENAME: TMPA15C.
                          INITIATING MCS: *SYSTEM/COMS.
   17:59:14    BOT  6623  *LIBRARY/MAINTENANCE.
                          STACK NUMBER: 07FD
                          TASK TYPE: COROUTINE(CALL)
                          PRIORITY: 50
                          USERCODE: ALAJUA.
                          CHARGECODE: 6842.
                          SOURCENAME: TMPA15C.
                          INITIATING MCS: *SYSTEM/COMS.
   17:59:14         6622  MSRFIN9: NO FILE PROTOCOLS/FILE000 (MT) [000425] #1 FAMILYOWNER=.
   17:59:14         6623  MSRFIN9: NO FILE PROTOCOLS/FILE000 (MT) [000425] #1 FAMILYOWNER=.
   18:00:20         6622  MSRFIN6: UNMATCHED SERIALNO PROTOCOLS/FILE000 (MT) [000425] #1 FAMILYOWNER=.
   18:00:20         6623  MSRFIN6: UNMATCHED SERIALNO PROTOCOLS/FILE000 (MT) [000425] #1 FAMILYOWNER=.
```

Figure 7-31. JOB Option Output

```
LOGANALYZER VERSION: 40.074.021,  SDASUPPORT VERSION: 40.074.012,  JOBFORMATTER VERSION: 40.074.119.
MCP *SYSTEM/ASD/MCP/40074F VERSION: 40.074 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 12/04/1991 14:41:04

REQUEST:  PRINTER LIB USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #002688 CREATED BY A15 (SYSTEM SERIAL = 8) ON 12/04/1991 10:42:00
TITLE = *SYSTEM/SUMLOG ON DISKB (CURRENT SUMLOG).

FILE CONTAINS 78039 RECORDS FROM 12/04/1991 10:42:00 TO 12/04/1991 14:42:00


Wednesday, December 4, 1991
   14:39:33    LIB  6307  *OBJECT/MAIL ON SYS39.   (STACK BE4)
                          DELINK FROM LIBRARY.
                          ATTRIBUTES:  INTNAME = MAIL LIB,TITLE = *SYSTEM/MAILSUPPORT/TEST ON PACK.,FUNCTIONNAME = MAILSUPPORT,
                                      LIBACCESS = BYFUNCTION
                          LIBRARIES:
                          6803    (JANB)SYSTEM/MAILSUPPORT ON PACK.   (STACK 179)
                                  SECURITY CLASS=1
                          DELINKAGE WAS SUCCESSFUL.
   14:39:35    LIB  6319  *OBJECT/CHECKMAIL ON PACK.   (STACK C00)
                          LINK TO LIBRARY.
                          ATTRIBUTES:  INTNAME = MAIL LIB,FUNCTIONNAME = MAILSUPPORT, LIBACCESS = BYFUNCTION,
                                     TITLE = *SYSTEM/MAILSUPPORT/TEST ON PACK.
                          LIBRARIES:
                          6803    (JANB)SYSTEM/MAILSUPPORT ON PACK.   (STACK 179)
                                  SECURITY CLASS=1
                          LINKAGE WAS SUCCESSFUL (RESULT VALUE 2).
   14:39:35    LIB  6319  *OBJECT/CHECKMAIL ON PACK.   (STACK C00)
                          DELINK FROM LIBRARY.
                          ATTRIBUTES:  INTNAME = MAIL LIB,FUNCTIONNAME = MAILSUPPORT, LIBACCESS = BYFUNCTION,
                                     TITLE = *SYSTEM/MAILSUPPORT/TEST ON PACK.
                          LIBRARIES:
                          6803    (JANB)SYSTEM/MAILSUPPORT ON PACK.   (STACK 179)
                                  SECURITY CLASS=1
                          DELINKAGE WAS SUCCESSFUL.
   14:40:20    LIB  6322  *SYSTEM/ALGOL.   (STACK C06)
                          DELINK FROM LIBRARY.
                          ATTRIBUTES:  INTNAME = GENERALSUPPORT, LIBACCESS = BYFUNCTION
                          LIBRARIES:
                          6598    *SYSTEM/GENERALSUPPORT.   (STACK 041)
                                  SECURITY CLASS=3
                          DELINKAGE WAS SUCCESSFUL.
   14:40:43    LIB  6329  (CUVIER)OBJECT/SPOC/LIBRARY ON DISKB.   (STACK 3092)
                          LIBRARY FROZEN.
                          STATUS : TEMPORARY, BYTITLE, SHAREDBYALL, SECURITY CLASS=0.
   14:40:58    LIB  6329  (CUVIER)OBJECT/SPOC/LIBRARY ON DISKB.   (STACK 3092)
                          LIBRARY THAWED.
                          STATUS : TEMPORARY, BYTITLE, SHAREDBYALL, SECURITY CLASS=0.
   14:40:59    LIB  6330  "JOBFILE/CONVERTER".   (STACK C18)
                          DELINK FROM LIBRARY.
                          ATTRIBUTES:  INTNAME = MCPSUPPORT, LIBACCESS = BYFUNCTION
                          LIBRARIES:
                          0015    *SYSTEM/ASD/MCP/40074F.   (STACK 000)
                                  SECURITY CLASS=1
                          DELINKAGE WAS SUCCESSFUL.
  _14:41:04    LIB  6332  *SYSTEM/IGSDASUPPORT.   (STACK 3098)
                          LIBRARY FROZEN.
                          STATUS : TEMPORARY, BYFUNCTION, PRIVATE, SECURITY CLASS=4, TRUSTED.
```

Figure 7–32. LIB Option Output

```
LOGANALYZER VERSION: 39.036.059,  SDASUPPORT VERSION: 39.036.044,  JOBFORMATTER VERSION: 39.036.432.
MCP *SYSTEM/MCP/390360 VERSION: 39.036 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 503) ON 10/08/1990 13:53:08

REQUEST:  P MIX 6513 THRU 6531 (UNSORTED BY DEFAULT)

SUMLOG #001439 CREATED BY A9 (SYSTEM SERIAL = 3) ON 06/15/1990 13:17:54
TITLE = (JAHAR)LASTLOG ON PACKA.

FILE CONTAINS 721 RECORDS FROM 06/15/1990 13:18:00 TO 06/15/1990 13:36:00

Friday, June 15, 1990
   13:18:26    BOT   6513   *SYSTEM/ALGOL ON SYSTESTS.
                            CODE COMPILED: 05/29/1990 16:31:15 BY ALGOL 39.28
                            TASK TYPE: DEPENDENT TASK(PROCESS)
                            STACK NUMBER: 008E, PRIORITY: 50, SOURCENAME: TMPA9B1/CANDE/1.
                            CODEFILE: (JAHAR)CANDE/CODE1830 ON SYSTESTS.
                            USERCODE: JAHAR.
                            INITIATING MCS: SYSTEM/CANDE.
   13:18:27    OPEN  6513   EXT NAME: (JAHAR)A1.
                            INT NAME: CARD.
                            FILE ACCESS RULE = DECLARER (ACTOR = DECLARER = STACK 08E, JOB 6502, TASK 6513)
                            FAMILYNAME SYSTESTS, KIND=PACK UNITNO 48, FILEKIND=ALGOLSYMBOL  FILE SIZE 14 SECTORS
                            USE = CLOSED, OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
   13:18:28    OPEN  6513   EXT NAME: (JAHAR)CANDE/CODE1830.
                            INT NAME: CODE.
                            FILE ACCESS RULE = DECLARER (ACTOR = DECLARER = STACK 08E, JOB 6502, TASK 6513)
                            FAMILYNAME SYSTESTS, KIND=PACK UNITNO 48, FILEKIND=DATA  FILE SIZE 0 SECTORS
                            USE = CLOSED, OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
   13:18:30    CLOSE 6513   EXT NAME: (JAHAR)CANDE/CODE1830.
                            INT NAME: CODE.
                            FAMILY NAME: SYSTESTS
                            FILE ACCESS RULE = DECLARER (ACTOR = DECLARER = STACK 08E, JOB 6502, TASK 6513)
                            CLOSE TYPE: NORMAL  ASSOCIATION: RELEASE  DISPOSITION: CRUNCH
                            IO TIME: 00:00:00.0723 TRANSACTION COUNT = 17 PHYSICAL (READ COUNT=0, WRITE COUNT=3)
                            FILESTRUCTURE = ALIGNED180, BUFFER SIZE SOURCE = BLOCKSIZE, BUFFERSIZE = 270
                            FRAMESIZE = 48, MIN/MAXRECSIZE = 0/ 30, BLOCKSIZE = 270
                            INTMODE = SINGLE           EXTMODE = SINGLE
                            UPDATED  CRUNCHED   AREASIZE 504 SECTORS, FILE SIZE 18 SECTORS
                            SERIAL NO 659048, CREATION DATE 90166 CYCLE 1 VERSION 0 SAVEFACTOR 999
   13:18:30    CLOSE 6513   EXT NAME: (JAHAR)A1.
                            INT NAME: CARD.
                            FAMILY NAME: SYSTESTS
                            FILE ACCESS RULE = DECLARER (ACTOR = DECLARER = STACK 08E, JOB 6502, TASK 6513)
                            CLOSE TYPE: NORMAL  ASSOCIATION: RELEASE  DISPOSITION: BLOCKEXIT
                            IO TIME: 00:00:00.0283 TRANSACTION COUNT = 19 PHYSICAL (READ COUNT=1, WRITE COUNT=0)
                            FILESTRUCTURE = ALIGNED180, BUFFER SIZE SOURCE = BLOCKSIZE, BUFFERSIZE = 420
                            FRAMESIZE = 48, MIN/MAXRECSIZE = 0/ 15, BLOCKSIZE = 420
                            INTMODE = EBCDIC           EXTMODE = EBCDIC
                            PERM   SPEED=00    AREASIZE 504 SECTORS, FILE SIZE 14 SECTORS
                            SERIAL NO 659048, CREATION DATE 90151 CYCLE 1 VERSION 0 SAVEFACTOR 30
   13:18:30    EOT   6513  *SYSTEM/ALGOL ON SYSTESTS.
                            USERCODE: JAHAR.
                            STACK NUMBER: 008E            AVERAGE DISK SECTORS IN USE BY PERMANENT FILES: 205.
                            PROCESSOR TIME: 00:00:01.0895 AVERAGE MEMORY USAGE: CODE=13585, DATA=14285
                            I/O       TIME: 00:00:00.6281 MEMORY INTEGRAL: CODE=23.334, DATA=24.536
                            READYQ    TIME: 00:00:00.1435 INITIAL PBITS: 231.
                            INITPBIT  TIME: 00:00:00.4682 MAXIMUM NUMBER OF ASDS USED: 124.
                            ELAPSED   TIME: 00:00:03.8099 MAXIMUM SAVE MEMORY USED: 10014.
```

Figure 7–33. MIX Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 18:20:52

REQUEST:  PRINTER SESSION 5971 USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 36568 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 18:22:00

Tuesday, April 3, 1990
   16:07:00   LOGON 5971  ORIGINATING LSN: 521
                          USERCODE: WJW.
                          USERCODE PRIVILEGE: PU
                          MCS: 2
                          CHARGE CODE: 6751.
                          STATION NAME: TA274/CANDE/1.
                          SIGN ON BY NEW LOG-ON
   16:07:01    BOT  5972  *OBJECT/CHECKMAIL ON PACK.
                          STACK NUMBER: 04D4
                          CODE COMPILED: 03/28/1990 21:02:11 BY DCALGOL 38.93
                          TASK TYPE: DEPENDENT TASK(PROCESS)
                          PRIORITY: 50
                          USERCODE: WJW.
                          CHARGECODE: 6751.
                          SOURCENAME: TA274/CANDE/1.
                          INITIATING MCS: SYSTEM/CANDE.
   16:07:01    OPEN 5972  EXT NAME: REMOTE.
                          INT NAME: REMOTE.
                          FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK 4D4, JOB 5971, TASK 5972)
                          USE = CLOSED KIND=REMOTE
                          OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
   16:07:01   CLOSE 5972  EXT NAME: REMOTE.
                          INT NAME: REMOTE.
                          FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK 4D4, JOB 5971, TASK 5972)
                          IO TIME      : 00:00:00.0000
                          CLOSE TYPE: NORMAL   ASSOCIATION: RELEASE   DISPOSITION: BLOCKEXIT
                          TRANSACTION COUNT = 1        PHYSICAL (READ COUNT=0, WRITE COUNT=0)
                          BUFFER CATEGORY: BUFFERGOAL NOT APPLICABLE    BUFFERSIZE: 0
                          BLOCKSIZE      = 80          MIN/MAXRECSIZE    = 0/ 80         INTMODE = EBCDIC
   16:07:01    EOT  5972  *OBJECT/CHECKMAIL ON PACK.
                          STACK NUMBER: 04D4           USERCODE: WJW.
                          PROCESSOR TIME: 00:00:00.1411 CHARGECODE: 6751.
                          I/O       TIME: 00:00:00.1392 AVERAGE MEMORY USAGE: CODE=721, DATA=2109
                          READYQ    TIME: 00:00:00.0130 MEMORY INTEGRAL: CODE=0.202, DATA=0.591
                          INITPBIT  TIME: 00:00:00.0280 INITIAL PBITS: 91.
                          ELAPSED   TIME: 00:00:00.3310 MAXIMUM NUMBER OF ASDS USED: 28.
                                                        MAXIMUM SAVE MEMORY USED: 2183.

NORMAL TERMINATION FOR LOGANALYZER PROGRAM.
```

Figure 7–34. SESSION Option Output

```
LOGANALYZER VERSION: 39.023.045,   SDASUPPORT VERSION: 39.023.375,   JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 18:28:57

REQUEST:  PRINTER TASK 6697 USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000781 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/03/1990 15:53:36
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 37475 RECORDS FROM 04/03/1990 15:54:00 TO 04/03/1990 18:30:00

Tuesday, April 3, 1990
   18:19:40     BOT  6697   *SYSTEM/FILEDATA.
                            STACK NUMBER: 0B6C
                            CODE COMPILED: 03/19/1990 22:37:27 BY DCALGOL 39.23
                            TASK TYPE: DEPENDENT TASK(PROCESS)
                            PRIORITY: 50
                            USERCODE: SLADO.
                            CHARGECODE: 6251.
                            SOURCENAME: TB235/CANDE/2.
                            INITIATING MCS: SYSTEM/CANDE.
   18:19:40     OPEN 6697   EXT NAME: LINE.
                            INT NAME: LINE.
                            FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK B6C, JOB 6692, TASK 6697)
                            USE = CLOSED KIND=REMOTE
                            OPENTYPE: WAIT, POSITION: AT FRONT, MOTION: NONE
   18:19:40     OPEN 6697   EXT NAME: (SLADO)JOB6697.
                            INT NAME: DATABASE.
                            FAMILY NAME: DCCP
                            FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK B6C, JOB 6692, TASK 6697)
                            USE = CLOSED KIND=PACK   FILEKIND= DATA     FILE SIZE 0 SEGS.  UNIT NUMBER 45
                            OPENTYPE: AVAILABLE, POSITION: 0, MOTION: NONE
   18:19:40    DEIMP 6697   *SYSTEM/FILEDATA.
                            COMPILER:  DCALGOL 39.23
                            ORIGINATING LSN:  500 MCS:  2
                            USERCODE:  SLADO.
                            WARNING 77: GETSTATUS DIRECTORY REQUEST MASK BIT 14 WILL BE DEIMPLEMENTED ON MARK 4.0. SEE THE GETSTATUS/
                            SETSTATUS MANUAL.
   18:19:40    CLOSE 6697   EXT NAME: (SLADO)JOB6697.
                            INT NAME: DATABASE.
                            FAMILY NAME: DCCP
                            FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK B6C, JOB 6692, TASK 6697)
                            IO TIME       : 00:00:00.0189
                            CLOSE TYPE: NORMAL   ASSOCIATION: RELEASE   DISPOSITION: REWIND
                            TRANSACTION COUNT = 31        PHYSICAL (READ COUNT=1, WRITE COUNT=1)
                            FILE STRUCTURE: ALIGNED180
                            BUFFER CATEGORY: BUFFERGOAL NOT APPLICABLE    BUFFERSIZE: 900
                            BLOCKSIZE       = 900          MIN/MAXRECSIZE    = 0/ 30          INTMODE = SINGLE
                            UPDATED  AREASIZE 600 SECTORS   FILE SIZE 600 SECTORS
                            SERIAL NO 659045   CREATION DATE 90093 CYCLE 1 VERSION 0 SAVEFACTOR 0
   18:19:40    CLOSE 6697   EXT NAME: LINE.
                            INT NAME: LINE.
                            FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK B6C, JOB 6692, TASK 6697)
                            IO TIME       : 00:00:00.0000
                            CLOSE TYPE: NORMAL   ASSOCIATION: RELEASE   DISPOSITION: BLOCKEXIT
                            TRANSACTION COUNT = 1         PHYSICAL (READ COUNT=0, WRITE COUNT=0)
                            BUFFER CATEGORY: BUFFERGOAL NOT APPLICABLE    BUFFERSIZE: 0
                            BLOCKSIZE       = 1863         MIN/MAXRECSIZE    = 0/ 1863        INTMODE = EBCDIC
   18:19:41     EOT  6697   *SYSTEM/FILEDATA.
                            STACK NUMBER: 0B6C           AVERAGE DISK SECTORS IN USE BY TEMPORARY FILES: 148.
```

Figure 7–35. TASK Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 10:18:31

REQUEST:  PRINTER COMMENT USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000785 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 06:01:43
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 72061 RECORDS FROM 04/04/1990 06:02:00 TO 04/04/1990 10:20:00

Wednesday, April 4, 1990
   10:15:33       ->8106  OPERATOR ENTERED: LC EVENT 0828 REASON 2 COMMENT DMP285.
   11:23:15       ->8106  OPERATOR ENTERED: LC EVENT DUMPTAPE=289 SERIALNO=DMP289 COMMENT=FATAL DUMP, UNPLANNED.

NORMAL TERMINATION FOR LOGANALYZER PROGRAM.
```

Figure 7–36. COMMENT Option Output

```
LOGANALYZER VERSION: 39.022.045,  SDASUPPORT VERSION: 39.020.035,  JOBFORMATTER VERSION: 39.022.324.
MCP *SYSTEM/MCP/39022C VERSION: 39.022 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 14:15:57

REQUEST:  P /000705 1000 TO 1005 DEIMPLEMENTATION USERCODE. (UNSORTED BY DEFAULT)

SUMLOG #000705 CREATED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 09:33:46
TITLE = *SUMLOG/8/031490/000705 ON PACK.

FILE CONTAINS 88014 RECORDS FROM 03/14/1990 09:34:00 TO 03/14/1990 13:47:00

Wednesday, March 14, 1990
   10:00:02   DEIMP 3580   *FILERECOVERY ON PACK.
                           COMPILER:  ALGOL 36.191
                           ORIGINATING UNIT:  51
                           CODEFILE:  *FILERECOVERY ON PACK.
                           USERCODE:  JJBACKUP.
                           WARNING 8: THIS CODE FILE CAN'T BE RUN ON THE NEXT RELEASE
   10:00:03   DEIMP 3582   (JJBACKUP)OBJECT/COMMONLIBRARY ON PACK.
                           COMPILER:  DCALGOL 36.191
                           ORIGINATING UNIT:  51
                           USERCODE:  JJBACKUP.
                           WARNING 8: THIS CODE FILE CAN'T BE RUN ON THE NEXT RELEASE
   10:00:06   DEIMP 3578   (ALIA)OBJECT/DIRDUMP ON ORV.
                           COMPILER:  DCALGOL 38.83
                           ORIGINATING LSN:  634 MCS:  2
                           USERCODE:  ORV.
                           WARNING 79: ATTRIBSEARCHER CANNOT BE USED FOR ATTRIBUTES ADDED AFTER MARK 3.8.
   10:00:08   DEIMP 3586   (JJBACKUP)JJBACKUPDB.
                           COMPILER:  DCALGOL 39.22
                           ORIGINATING UNIT:  0
                           USERCODE:  JJBACKUP.
                           WARNING 74: ON MARK 4.2, THE SUBSYSTEM TASK ATTRIBUTE WILL BE DEIMPLEMENTED
   10:00:09   DEIMP 3580   (JJBACKUP)JJBACKUPDB.
                           COMPILER:  DCALGOL 39.22
                           ORIGINATING UNIT:  0
                           USERCODE:  JJBACKUP.
                           WARNING 74: ON MARK 4.2, THE SUBSYSTEM TASK ATTRIBUTE WILL BE DEIMPLEMENTED
   10:00:22   DEIMP 3594   *FILERECOVERY ON PACK.
                           COMPILER:  ALGOL 36.191
                           ORIGINATING UNIT:  51
                           CODEFILE:  *FILERECOVERY ON PACK.
                           USERCODE:  JJBACKUP.
                           WARNING 8: THIS CODE FILE CAN'T BE RUN ON THE NEXT RELEASE
   10:00:22   DEIMP 3595   (JJBACKUP)OBJECT/COMMONLIBRARY ON PACK.
                           COMPILER:  DCALGOL 36.191
                           ORIGINATING UNIT:  51
                           USERCODE:  JJBACKUP.
                           WARNING 8: THIS CODE FILE CAN'T BE RUN ON THE NEXT RELEASE
   10:00:25   DEIMP 3597   (JJBACKUP)JJBACKUPDB.
                           COMPILER:  DCALGOL 39.22
                           ORIGINATING UNIT:  0
                           USERCODE:  JJBACKUP.
```

Figure 7-37. DEIMPLEMENTATION Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 10:38:22

REQUEST:  PRINTER MSG USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000785 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 06:01:43
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 79304 RECORDS FROM 04/04/1990 06:02:00 TO 04/04/1990 10:39:00

Wednesday, April 4, 1990
    06:01:46         7297    MSRDISP3: DISPLAY:FLOG file not found:  *SYSTEM/FLOG..
    06:01:57         2562    MSRPREFMT34:  WARNING:HOST IRV # 325 MAY HAVE STATE MISMATCH.
    06:01:57         2562    MSRPREFMT34: HOST IRV # 325 IS STILL ESTABLISHING
    06:02:01         2543    MSRWARNG46: WARNING 46: LOGICAL FILE BLOCKSIZE INCONSISTENT WITH PHYSICAL FILE BLOCKSIZE @ (63836500)
    06:02:33         2543    MSRWARNG46: WARNING 46: LOGICAL FILE BLOCKSIZE INCONSISTENT WITH PHYSICAL FILE BLOCKSIZE @ (63836500)
    06:02:55         2562    MSRPREFMT34: HOST CAND553B # 162 IS NOW QUIET
    06:02:55         2562    MSRPREFMT34: HOST CAND553B # 162 IS NOW ESTABLISHING
    06:03:19         2537    MSRPREFMT34: HOST IRV # 325 IS NOW INTERRUPTED
    06:03:32         2543    MSRWARNG46: WARNING 46: LOGICAL FILE BLOCKSIZE INCONSISTENT WITH PHYSICAL FILE BLOCKSIZE @ (63836500)
    06:03:59         2537    MSRPREFMT34: HOST IRV # 325 IS NOW QUIET
    06:03:59         2562    MSRPREFMT34: HOST IRV # 325 IS NOW ESTABLISHING
    06:04:40         2562    MSRPREFMT34:  WARNING:HOST CAND553B # 162 MAY HAVE STATE MISMATCH.
    06:04:40         2562    MSRPREFMT34: HOST CAND553B # 162 IS STILL ESTABLISHING
    06:04:58         2537    MSRPREFMT34: HOST CAND553B # 162 IS NOW INTERRUPTED
    06:05:19         2537    MSRPREFMT34: HOST CAND553B # 162 IS NOW QUIET
    06:05:19         2562    MSRPREFMT34: HOST CAND553B # 162 IS NOW ESTABLISHING
    06:05:42         2562    MSRPREFMT34:  WARNING:HOST IRV # 325 MAY HAVE STATE MISMATCH.
    06:05:42         2562    MSRPREFMT34: HOST IRV # 325 IS STILL ESTABLISHING
    06:06:59         2562    MSRPREFMT34:  WARNING:HOST CAND553B # 162 MAY HAVE STATE MISMATCH.
    06:06:59         2562    MSRPREFMT34: HOST CAND553B # 162 IS STILL ESTABLISHING
    06:07:25         2562    MSRPREFMT34:  WARNING:HOST IRV # 325 MAY HAVE STATE MISMATCH.
    06:07:25         2562    MSRPREFMT34: HOST IRV # 325 IS STILL ESTABLISHING
    06:08:43         2562    MSRPREFMT34:  WARNING:HOST CAND553B # 162 MAY HAVE STATE MISMATCH.
    06:08:43         2562    MSRPREFMT34: HOST CAND553B # 162 IS STILL ESTABLISHING
    06:09:05         2562    MSRPREFMT34: HOST IRV # 325 IS NOW QUIET
    06:09:05         2562    MSRPREFMT34: HOST IRV # 325 IS NOW ESTABLISHING
    06:09:18         2537    MSRPREFMT34: HOST CAND553B # 162 IS NOW ACTIVE
    06:09:37         2543    MSRWARNG46: WARNING 46: LOGICAL FILE BLOCKSIZE INCONSISTENT WITH PHYSICAL FILE BLOCKSIZE @ (63836500)
    06:10:48         2562    MSRPREFMT34:  WARNING:HOST IRV # 325 MAY HAVE STATE MISMATCH.
    06:10:48         2562    MSRPREFMT34: HOST IRV # 325 IS STILL ESTABLISHING
    06:11:04         2537    MSRPREFMT34: HOST IRV # 325 IS NOW ACTIVE
    06:12:48         2537    MSRPREFMT34: HOST IRV # 325 IS NOW QUIET
    06:12:48         2543    MSRPREFMT34: HOST IRV # 325 IS NOW ESTABLISHING
    06:14:19         2562    MSRPREFMT34: HOST CAND553B # 162 IS NOW QUIET
    06:14:19         2543    MSRPREFMT34: HOST CAND553B # 162 IS NOW ESTABLISHING
    06:14:32         2562    MSRPREFMT34:  WARNING:HOST IRV # 325 MAY HAVE STATE MISMATCH.
    06:14:32         2562    MSRPREFMT34: HOST IRV # 325 IS STILL ESTABLISHING
    06:14:40         2537    MSRPREFMT34: HOST CAND553B # 162 IS NOW ACTIVE
    06:16:17         2562    MSRPREFMT34:  WARNING:HOST IRV # 325 MAY HAVE STATE MISMATCH.
    06:16:17         2562    MSRPREFMT34: HOST IRV # 325 IS STILL ESTABLISHING
    06:18:01         2562    MSRPREFMT34: HOST IRV # 325 IS NOW QUIET
    06:18:01         2562    MSRPREFMT34: HOST IRV # 325 IS NOW ESTABLISHING
    06:19:21         2537    MSRPREFMT34: HOST IRV # 325 IS NOW ACTIVE
    06:21:36         2557    MSRPREFMT34: COMS: 06:21 MONITOR >>> DCERROR: 06:21:35 TA270(333) TERMINATE ERROR: LASTFLAG=VERTICAL PARI
                             TY(CHAR),FLAGS=STATION/LINE NOT RDY;
    06:21:40         2557    MSRPREFMT34: COMS: 06:21 MONITOR >>> DCERROR: 06:21:39 TA242(327) TERMINATE ERROR: LASTFLAG=VERTICAL PARI
                             TY(CHAR),FLAGS=STATION/LINE NOT RDY;
    06:21:44         2557    MSRPREFMT34: COMS: 06:21 MONITOR >>> DCERROR: 06:21:43 TA242(327) TERMINATE ERROR: LASTFLAG=VERTICAL PARI
```

Figure 7-38. MSG Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFO
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 10:35:58

REQUEST:  PRINTER OPERATOR USERCODE . (UNSORTED BY DEFAULT)

SUMLOG #000785 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 06:01:43
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 78606 RECORDS FROM 04/04/1990 06:02:00 TO 04/04/1990 10:37

Wednesday, April 4, 1990
   06:29:03          ->2538   OPERATOR ENTERED:2538 SM: TO ELISE REMINDER: YOU HAVE 2 NEW MAIL MESSAGES.
   06:37:44          ->0002   OPERATOR ENTERED: ID 11 : QUIT
   06:37:48          ->0002   OPERATOR ENTERED: ID 111 : QUIT
   06:38:06          ->0002   OPERATOR ENTERED: ID 111 : CLEAR
   06:41:27          ->0002   OPERATOR ENTERED:2554 SM:STATUS 328-339.
   06:42:11          ->2557   OPERATOR ENTERED:2557 SM:READY  328-339.
   07:04:01          ->7304   OPERATOR ENTERED:7304LP
   07:04:01          ->2577   OPERATOR ENTERED:2577LP
   07:04:01          ->2554   OPERATOR ENTERED:2554LP
   07:04:02          ->7346   OPERATOR ENTERED:7304,2577,2554,2538,2529,2526,2525,2517LP
   07:04:02          ->2529   OPERATOR ENTERED:2529 SM: MAXTASKS =150.
   07:04:02          ->2529   OPERATOR ENTERED:2529 SM: MAXSTATIONS = 250.
   07:04:02          ->2529   OPERATOR ENTERED:2529 SM: MAXGRINDS = 13.
   07:04:02          ->2529   OPERATOR ENTERED:2529 SM: GRINDLIMIT = 5.
   07:04:02          ->2529   OPERATOR ENTERED:2529 SM: FACTOR WORK = 80 QUIT = 19.
   07:04:02          ->2529   OPERATOR ENTERED:2529 SM: DEPTH = 10.
   07:04:02          ->2529   OPERATOR ENTERED:2529 SM: NEWS.
   07:04:03          ->2529   OPERATOR ENTERED:2529 SM: SCHEDULE USERLIMIT 2.
   07:04:03          ->2529   OPERATOR ENTERED:2529 SM: SCHEDULE    5.
   07:04:03          ->2529   OPERATOR ENTERED:2529 SM: LAISSEZFILE = 4.
   07:04:03          ->2529   OPERATOR ENTERED:2529 SM: LGSTA M333.
   07:04:03          ->2529   OPERATOR ENTERED:2529 SM: LGOP + M333: LGOFF, LGERR, AUTOINFO.
   07:04:03          ->2529   OPERATOR ENTERED:2529 SM: OP + ALLMSG, SWAPALL, DIALLOGIN, SCATTER.
   07:04:05          ->7353   OPERATOR ENTERED: BEGIN JOB NEWS;  STARTTIME = 15:30; QUEUE = 5;  RUN *SUPPRESS/MIX/NOS ON PACK
   07:04:25          ->7370   OPERATOR ENTERED: MQ 0  MIXLIMIT= 8
   07:04:29          ->7371   OPERATOR ENTERED: MQ 4  MIXLIMIT= 2
   07:04:29          ->7370   OPERATOR ENTERED: MQ 0  DEF(PRI=50), LIM (PRI=99)
   07:04:29          ->7374   OPERATOR ENTERED: MQ 30 MIXLIMIT= 2, DEF(PRI=40)
   07:04:29          ->2523   OPERATOR ENTERED: PS DEV+ LP5
   07:04:33          ->7377   OPERATOR ENTERED: DQO
   07:04:34          ->2523   OPERATOR ENTERED: PS DEVICES+LP4
   07:04:37          ->7379   OPERATOR ENTERED: SB PACK =DLBACKUP TAPE7 =TAPE7 TAPE9 =TAPE9 PETAPE =PETAPE TAPE =TAPE DISK =DLBACKUP
   07:04:37          ->7380   OPERATOR ENTERED: HU SITE.
   07:04:38          ->7382   OPERATOR ENTERED: COMPILERTARGET = LEVELO
   07:04:42          ->7385   OPERATOR ENTERED: ID 108 : OPTION = NONE
   07:04:42          ->7378   OPERATOR ENTERED: OP- 0  (ERROR: VALUE=),1,3,6,7,9,10,11,13,15,17,18,22,29,47
   07:04:47          ->7388   OPERATOR ENTERED: ID 109 : OPTION = NONE
   07:04:47          ->7374   OPERATOR ENTERED: MQ 60 MIXLIMIT= 0, DEF(PRI=50)
   07:04:47          ->7378   OPERATOR ENTERED: CS *SUPPRESS/MIX/NOS.
   08:30:24          ->7502   OPERATOR ENTERED:7502LP-
   08:31:48          ->7508   OPERATOR ENTERED:7508 IL MT 28
   08:32:21          ->7508   OPERATOR ENTERED:7508 FA:TITLE=MB/SYSTEM/GAMMA/STANDALONE ON ASD.
   08:32:26          ->7508   OPERATOR ENTERED:7508 AX:OK.
   08:38:53          ->7530   OPERATOR ENTERED:7530LP-
```

Figure 7-39. OPERATOR Option Output

```
LOGANALYZER VERSION: 39.022.045,  SDASUPPORT VERSION: 39.020.035,  JOBFORMATTER VERSION: 39.022.324.
MCP *SYSTEM/MCP/39022C VERSION: 39.022 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 14:18:05

REQUEST:  P /000705 1330 TO 1400 SECURITY USERCODE. (UNSORTED BY DEFAULT)

SUMLOG #000705 CREATED BY A15 (SYSTEM SERIAL = 8) ON 03/14/1990 09:33:46
TITLE = *SUMLOG/8/031490/000705 ON PACK.

FILE CONTAINS 88014 RECORDS FROM 03/14/1990 09:34:00 TO 03/14/1990 13:47:00

Wednesday, March 14, 1990
    13:30:24            8289   SECURITY VIOLATION - INCORRECT PASSWORD
                               VIOLATION CODE: 10
                               ERROR ITEM: KAY.
                               ORIGINATING UNIT NUMBER: 0

    13:30:24            8289   MCS SECURITY VIOLATION:
                               INVALID USERCODE/PASSWORD AT LOG ON
                               ORIGINATING LSN: 688   ORIGINATING MCS: 2
                               STATION NAME: TC103/CANDE/7
                               USERCODE: KAY
                               ERROR ITEM: KAY

    13:41:16            8289   SECURITY VIOLATION - INCORRECT USERCODE
                               VIOLATION CODE: 9
                               ERROR ITEM: SABE.
                               ORIGINATING UNIT NUMBER: 0

    13:41:16            8289   MCS SECURITY VIOLATION:
                               INVALID USERCODE/PASSWORD AT LOG ON
                               ORIGINATING LSN: 671   ORIGINATING MCS: 2
                               STATION NAME: TB125/CANDE/3
                               USERCODE: SABE
                               ERROR ITEM: SABE

    13:41:23            8289   SECURITY VIOLATION - INCORRECT PASSWORD
                               VIOLATION CODE: 10
                               ERROR ITEM: KEIFFER.
                               ORIGINATING UNIT NUMBER: 0

    13:41:24            8289   MCS SECURITY VIOLATION:
                               INVALID USERCODE/PASSWORD AT LOG ON
                               ORIGINATING LSN: 671   ORIGINATING MCS: 2
                               STATION NAME: TB125/CANDE/3
                               USERCODE: KEIFFER
                               ERROR ITEM: KEIFFER

NORMAL TERMINATION FOR LOGANALYZER PROGRAM.
```

Figure 7–40. SECURITY Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 11:20:57

REQUEST:  PRINTER EOT DUMP USERCODE. (UNSORTED BY DEFAULT)

SUMLOG #000786 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 10:55:49
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 11015 RECORDS FROM 04/04/1990 10:56:00 TO 04/04/1990 11:22:00


04/04/1990 10:55:49.9252   8442/8442  TYPE: 1.2  LENGTH: 53 (47 FIXED)
         010220FA20FA  000000015FEE  0003D1441654  BC3500010002  00000000004B  ......  ....^.  ..J...  ......  ......
         000000000000  00000040002F  000000200033  000000000000  000000000000  ......  ......  ......  ......  ......
         000000000000  00000001A31E  000000033C25  000000000000  000000000000  ......  ....T.  ......  ......  ......
         000000000000  20B8992B1F45  00004602541F  000000000005  000000000000  ......  ..R...  ......  ......  ......
         000000000000  23AE7795E721  24F2F468E7AF  000000275569  000000000000  ......  ...NX.  .24.X.  ......  ......
         AA4008270017  000000000000  000000000000  000000000000  000000019D20  ......  ......  ......  ......  ......
         000000005A2F  00000000009A  000000000000  000000000000  4E7E3D11CC06  ....].  ......  ......  ......  +=....
         000000000000  000000000000  000000000000  000000000000  000000000000  ......  ......  ......  ......  ......
         000000000023  000000000000  000000000000  2515461187E8  000000000000  ......  ......  ......  ....GY  ......
         000000000EA9  00000000024F  18010304C6C9  D3C507C8C1D5  C4D3C5D907D3  .....Z  .....!  ....FI  LE.HAN  DLER.L
         C6C1F1F5C3C4  08010104E2D7  C1E900000000

Wednesday, April 4, 1990
    10:55:49     EOJ  8442   FILE/HANDLER/SFA12AB.
                             STACK NUMBER: 024F            AVERAGE DISK SEC
                             PROCESSOR TIME: 00:00:00.2575 USERCODE: SASHA.
                             I/O       TIME: 00:00:00.5088 AVERAGE MEMORY USAGE: CODE=95165, DATA=3679
                             READYQ    TIME: 00:00:00.2538 MEMORY INTEGRAL: CODE=72.926, DATA=2.819
                             INITPBIT  TIME: 00:00:00.0554 INITIAL PBITS: 154.
                             ELAPSED   TIME: 00:00:06.1866 MAXIMUM NUMBER OF ASDS USED: 35.
                                                           MAXIMUM SAVE MEMORY USED: 3753.


04/04/1990 10:55:52.2672   8446/8446  TYPE: 1.2  LENGTH: 51 (47 FIXED)
         010220FE20FE  000000015FEE  0003D152FA24  BC3300010002  000000000050  ......  ....^.  ..J...  ......  .....&
         000000000002  00000040002F  000000000000  000000000000  000000000000  ......  ......  ......  ......  ......
         000000000000  0000000023DB  00000000B0A6  000000000000  000000000000  ......  ......  .....W  ......  ......
         000000000000  000000000000  000002469EB7  000000000005  000000000000  ......  ......  ......  ......  ......
         000000000000  000000000000  2495EEFB8659  000000022D21  000000000000  ......  ......  .N..F.  ......  ......
         000000000000  000000000000  000000000000  000000000000  0000000136E8  ......  ......  ......  ......  .....Y
         00000000057B  00000000000B  000000000000  000000000000  4E7E3D150CC7  .....#  ......  ......  ......  +=...G
         000000000000  000000000000  000000000000  000000000000  000000000000  ......  ......  ......  ......  ......
         000000000005  000000000000  000000000000  000000000000  000000000000  ......  ......  ......  ......  ......
         000000000288  000000000255  1301010FE4D7  C4C1E3C56DC6  C1E4D3E3D3D6  ......  ......  ....UP  DATE_F  AULTLO
         C70000000000                                                          G.....

    10:55:52     EOJ  8446   "UPDATE FAULTLOG".
                             STACK NUMBER: 0255            AVERAGE MEMORY USAGE: CODE=0, DATA=702
                             PROCESSOR TIME: 00:00:00.0220 MEMORY INTEGRAL: CODE=0.000, DATA=0.092
                             I/O       TIME: 00:00:00.1085 INITIAL PBITS: 11.
                             READYQ    TIME: 00:00:00.1910 MAXIMUM NUMBER OF ASDS USED: 5.
                             INITPBIT  TIME: 00:00:00.0033 MAXIMUM SAVE MEMORY USED: 696.
                             ELAPSED   TIME: 00:00:00.3423

04/04/1990 10:55:54.9856   8450/8450  TYPE: 1.2  LENGTH: 53 (47 FIXED)
         010221022102  000000015FEE  0003D1644284  BC3500010002  00000000004B  ......  ....^.  ..J...  ......  ......
```

Figure 7–41. DUMP Option Output

```
LOGANALYZER VERSION: 39.023.045,  SDASUPPORT VERSION: 39.023.375,  JOBFORMATTER VERSION: 39.023.328.
MCP *SYSTEM/MCP/39023G VERSION: 39.024 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 11:24:04

REQUEST:  PRINTER 1100 TO 1110 BOT EOT UNSORTED USERCODE. (UNSORTED IS NOW DEFAULT) .

SUMLOG #000786 CREATED BY A15 (SYSTEM SERIAL = 8) ON 04/04/1990 10:55:49
TITLE = *SYSTEM/SUMLOG ON PACK (CURRENT SUMLOG).

FILE CONTAINS 12210 RECORDS FROM 04/04/1990 10:56:00 TO 04/04/1990 11:25:00

Wednesday, April 4, 1990
   11:00:05    EOT  8481   MAIL/REMINDER.
                           STACK NUMBER: 0293                AVERAGE MEMORY USAGE: CODE=103172, DATA=3760
                           PROCESSOR TIME: 00:00:10.3289 MEMORY INTEGRAL: CODE=1100.138, DATA=40.092
                           I/O       TIME: 00:00:00.3341 INITIAL PBITS: 6651, OTHER PBITS: 62.
                           READYQ    TIME: 00:00:02.4548 MAXIMUM NUMBER OF ASDS USED: 24.
                           INITPBIT  TIME: 00:00:01.8576 MAXIMUM SAVE MEMORY USED: 2569.
                           OTHERPBIT TIME: 00:00:00.1137
                           ELAPSED   TIME: 00:00:56.3818

   11:00:17    BOJ  8483   (RASHID)OBJECT/PATCH/NED ON FIRE.
                           STACK NUMBER: 029A
                           CODE COMPILED: 10/19/1989 14:21:20 BY DCALGOL 37.303
                           QUEUE: 0
                           ORIGINATING LSN: 464 MCS: 1
                           PRIORITY: 80
                           USERCODE: CHANDU.
                           CHARGECODE: 6731.
                           SOURCENAME: TB243.
                           INITIATING MCS: *SYSTEM/COMS.
   11:00:20    BOT  8484   *OBJECT/ED ON MCPMAST.
                           STACK NUMBER: 029C
                           CODE COMPILED: 03/19/1990 23:08:01 BY DCALGOL 39.23
                           TASK TYPE: DEPENDENT TASK(PROCESS)
                           PRIORITY: 50
                           USERCODE: LOCARNA.
                           CHARGECODE: 6685.
                           SOURCENAME: TB109/CANDE/3.
                           INITIATING MCS: SYSTEM/CANDE.
   11:00:25    BOJ  8485   (RASHID)OBJECT/PATCH/NED/PARSER ON FIRE.
                           STACK NUMBER: 029D
                           CODE COMPILED: 10/05/1989 13:51:31 BY DCALGOL 37.303
                           QUEUE: 0
                           ORIGINATING LSN: 464 MCS: 1
                           PRIORITY: 50
                           USERCODE: CHANDU.
                           CHARGECODE: 6731.
                           SOURCENAME: TB243.
                           INITIATING MCS: *SYSTEM/COMS.
   11:00:37    SNTX 8327   *SYSTEM/NEWP ON SYS39.
                           STACK NUMBER: 019C            AVERAGE DISK SECTORS IN USE BY TEMPORARY FILES: 184940.
                           PROCESSOR TIME: 00:03:36.0610 AVERAGE DISK SECTORS IN USE BY PERMANENT FILES: 569455.
                           I/O       TIME: 00:05:10.7478 USERCODE: X.
                           READYQ    TIME: 00:01:27.1873 CHARGECODE: 6685.
                           INITPBIT  TIME: 00:00:01.8125 LINES PRINTED: 122.
                           OTHERPBIT TIME: 00:00:00.9429 AVERAGE MEMORY USAGE: CODE=63864, DATA=672092
                           ELAPSED   TIME: 00:13:19.0179 MEMORY INTEGRAL: CODE=33644.095, DATA=354064.143
                                                         INITIAL PBITS: 3611, OTHER PBITS: 377.
                                                         MAXIMUM NUMBER OF ASDS USED: 3279.
```

Figure 7–42. UNSORTED Option Output

```
LOGANALYZER VERSION: 39.031.052,  SDASUPPORT VERSION: 39.030.042,  JOBFORMATTER VERSION: 39.031.392.
MCP *SYSTEM/MCP/39031G VERSION: 39.032 (MCP/AS)
ANALYZED BY A15 (SYSTEM SERIAL = 8) ON 07/24/1990 18:39:06

REQUEST:  PRINTER 1800 TO 1810 BOT FILE USERCODE . SORTED

SUMLOG #001182 CREATED BY A15 (SYSTEM SERIAL = 8) ON 07/24/1990 15:17:29
TITLE = *SYSTEM/SUMLOG ON DISK (CURRENT SUMLOG).

FILE CONTAINS 83929 RECORDS FROM 07/24/1990 15:17:00 TO 07/24/1990 18:40:00

Tuesday, July 24, 1990
    18:01:07    BOJ  0111  MCP/HOST.
                           JOB ENTERED SYSTEM: 07/24/1990 18:01:07 FROM WFL 39.31
                           QUEUE: 30, ORIGINATING LSN: 589 MCS: 2
                           STACK NUMBER: 07C0, PRIORITY: 40, SOURCENAME: TB131/CANDE/1.
                           USERCODE: WONG.
                           CHARGECODE: 1835.
                           INITIATING MCS: SYSTEM/CANDE.
    18:01:07    BOT  0112  *SYSTEM/PATCH ON SYSPK.
                           CODE COMPILED: 06/27/1990 11:59:46 BY ALGOL 39.30
                           TASK TYPE: DEPENDENT TASK(PROCESS)
                           STACK NUMBER: 07C3, PRIORITY: 40, SOURCENAME: TB131/CANDE/1.
                           USERCODE: WONG.
                           CHARGECODE: 1835.
                           INITIATING MCS: SYSTEM/CANDE.
    18:01:07    OPEN 0112  EXT NAME: CARD.
                           INT NAME: CARD.
                           FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK 7C3, JOB 0111, TASK 0112)
                           USE = IN KIND=READER (PSEUDOREADER)
                           OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
    18:01:09    BOT  0113  *SYSTEM/NEWP ON SYSPK.
                           CODE COMPILED: 06/27/1990 12:31:47 BY NEWP 39.30
                           TASK TYPE: COROUTINE(CALL)
                           STACK NUMBER: 07C7, PRIORITY: 40, SOURCENAME: TB131/CANDE/1.
                           CODEFILE: MCP/CPFIX/U ON ACCT.
                           USERCODE: WONG.
                           CHARGECODE: 1835.
                           INITIATING MCS: SYSTEM/CANDE.
    18:01:10    OPEN 0113  EXT NAME: (WONG)MCP/CPFIX/PATCHOUTPUT.
                           INT NAME: CARD.
                           FAMILY NAME: MCPMAST
                           FILE ACCESS RULE = DECLARER   (ACTOR = DECLARER = STACK 7C7, JOB 0111, TASK 0113)
                           USE = CLOSED KIND=PACK   FILEKIND= DATA     FILE SIZE 28 SEGS.  UNIT NUMBER 72
                           OPENTYPE: WAIT, POSITION: 0, MOTION: NONE
    18:02:53    BOJ  0115  SERVER/LP5.
                           QUEUE: 0, ORIGINATING UNIT: 0
                           STACK NUMBER: 082C, PRIORITY: 80
    18:03:26    BOJ  0117  ICONFIGCOMP.
                           JOB ENTERED SYSTEM: 07/24/1990 18:03:25 FROM WFL 39.31
                           QUEUE: 15, ORIGINATING LSN: 555 MCS: 2
                           STACK NUMBER: 084F, PRIORITY: 45, SOURCENAME: TB154/CANDE/4.
```

Figure 7–43. SORTED Option Output

Section 8
LOGGER

This section describes the LOGGER utility program, which enables a user to obtain reports that aid in the analysis of system performance and utilization. LOGGER also serves as a basis for the installation's billing system. The following are two important features of this log analysis program:

- LOGGER generates a wide variety of reports, depending on individual installation requirements.
- LOGGER combines data over various time intervals in order to generate long-term reports.

LOGGER accepts a set of report-specification commands from the user and interprets these specifications. These commands can specify information such as which data from each log file is to be included in the report, sorting by a particular data item, and data items to be used as control breaks. Usually, a set of report specifications consists of only six or seven commands. Therefore, the setup time involved in running LOGGER is minimal. If an installation has several different reports that it periodically wishes to obtain, the reports can be kept in one disk file. At run time, LOGGER reads the appropriate report specifications from this file.

Original data for LOGGER is stored in either the current log file (SYSTEM/SUMLOG) or in one or more of the SUMLOGS created with the TL (Transfer Log) system command. LOGGER can also generate reports from files that it saves each time it is run, thereby avoiding the need to load large numbers of SUMLOG files in order to generate weekly or monthly reports.

When run, LOGGER can be instructed to obtain the data necessary for a specified report from one of four sources:

- SYSTEM/SUMLOG. Refer to Section 12, "SUMLOG," for information about this file.
- SUMLOG files for either one date or a range of dates.
- Files generated and saved from a previous run of LOGGER.
- The disk resource control (DRC) system resource usage contained in the USERDATAFILE.

Each time LOGGER is run using SUMLOG files as the source, it creates files titled JOBSUMMARY/<mmddyy> and STATISTICS/<mmddyy> and, optionally, a file titled FILEIODATA/<mmddyy>. These files contain the saved data referred to previously along with the information needed to generate additional reports. Subsequently, LOGGER can be run with specifications to generate a report from one of these files or from files covering a range of dates.

Thus, long-term reports can be generated without having all of the original log files present. The saved files are much smaller than the original log files. For example, LOGGER could be run each day of the week and the log files removed. At the end of the week, LOGGER could also be used to generate a weekly report using the files that it created daily.

LOGGER can generate detailed and summary reports. A detailed report consists of one line for each record in the file; a summary report consists of one line for a particular group of records.

LOGGER Operation

LOGGER operates in two main phases. During the first phase, LOGGER creates data files containing the information necessary to generate a report. During the second phase, LOGGER reads the files and generates a report. The first phase is bypassed if LOGGER is instructed to use existing data files to generate a report.

LOGGER creates the files JOBSUMMARY, STATISTICS, DRCDATA, and FILEIODATA during the first phase of operation. These files contain the data necessary to generate reports. The JOBSUMMARY file contains data on each job, task, and message control system (MCS) session. The STATISTICS file contains overall system statistics such as number of jobs run and number of halt/loads. This data is grouped by 15-minute intervals. The FILEIODATA file contains data on file usage obtained from file open and close log entries. The JOBSUMMARY and STATISTICS files are always created; however, the DRCDATA file is created only when the *OPTION WRITEDRCDATA* command is used, and the FILEIODATA file is created only when the *OPTION WRITEIODATA* command is used.

LOGGER generates reports during the second phase of operation by reading data from the files created in the first phase. During any run, LOGGER generates a report based on only one of the files. When instructed, LOGGER can bypass the first operational phase by generating reports using files created during previous runs. LOGGER can also generate reports using files covering a range of dates (for example, all JOBSUMMARY files created during one week).

The following WFL job can be used to run LOGGER:

```
BEGIN JOB;
  RUN SYSTEM/LOGGER;
  <input records>
END JOB
```

```
<input records>

──┬──────────────────┬┬───────────────────┬┬──────────────────┬────────┤
  │ ┌←─ <in spec> ─┐ ││ ┌←─ <rep spec> ─┐ ││ ┌←─ <in spec> ─┐ │
  └─┴──<in spec>───┴─┘└─┴──<rep spec>───┴─┘└─┴──<in spec>───┴─┘

<in spec>

──<input-specification command>────────────────────────────────────────┤

<rep spec>

──<report-specification command>───────────────────────────────────────┤
```

The following text describes the meaning of each variable:

<input records>	Determines which information is used to generate a report and where the information comes from. If <input records> is not specified, a report is not generated. However, LOGGER still creates the JOBSUMMARY and STATISTICS files.
<input-specification command>	Described in "Input-Specification Commands" in this section.
<report-specification command>	Described in "Report-Specification Commands" in this section.

Example

The following WFL job generates a report based on SYSTEM/SUMLOG because a SOURCE command is not present. Refer to "SOURCE Command" in this section for more information.

```
BEGIN JOB;
  RUN SYSTEM/LOGGER; DATA CARD
  REPORT 1
END JOB
```

LOGGER Input

LOGGER input consists of input-specification and report-specification commands.

Each of the LOGGER command listings in this section features the following four levels of description:

- A brief introductory description explaining the function of the command and each of its specific applications
- A syntax diagram outlining the various options of the command
- An explanation section describing each of the syntax options
- Examples showing possible command combinations and the results that combination will produce

Input-Specification Commands

The following input-specification commands can be used as input to LOGGER.

```
<input-specification command>

──┬─<correction command>────────┬──────────────────────────────────────────┤
  ├─<maxrecords command>────────┤
  ├─<option command>────────────┤
  ├─<sort parameters command>───┤
  ├─<stop command>──────────────┤
  └─<use command>───────────────┘
```

CORRECTION Command

Use the CORRECTION command to correct values in a report if the log contains erroneous data for an entry. Only reports generated from the JOBSUMMARY file can have corrections applied to them.

This command must follow any report-specification commands.

```
<correction command>

── CORRECTION ──<mm/dd>──<time>──<mix number>──<item>── = ──<number>───┤

<mix number>

   ┌←──────────────┐
───┴─/4\─<integer>─┴──────────────────────────────────────────────────┤
```

The following text describes the meaning of each variable:

<mm/dd>	Specifies the date. *mm* specifies the month in a 2-digit form; *dd* specifies the day in a 2-digit form. The delimiter must be a slash (/).
<time>	Specifies the time to the previous hour in the form of <hh>. This option is required because the same mix number can occur more than once in the same day.
<mix number>	Any valid mix number.
<item> = <number>	Any valid numeric JOBSUMMARY file data item. Refer to "File Data Items" in this section for more information.

Examples

The following command replaces the processor time for task 1234 dated 09/01 with the value 2.54:

```
CORRECTION Ø9/Ø1 Ø9 1234 PROCESSTIME = 2.54.
```

The following command replaces the I/O time for task 7245 dated 08/24 with the value 1.72:

```
CORRECTION Ø8/24 ØØ 7245 IOTIME = 1.72.
```

MAXRECORDS Command

Use the MAXRECORDS command to specify the maximum number of log records to be read. This command can appear before or after a report-specification command. If no MAXRECORDS command is entered, the maximum number of log records to be read is 1,000,000,000.

```
<maxrecords command>

— MAXRECORDS —<integer>————————————————————————————————————|
```

The following text describes the meaning of the variable:

<integer>	Specifies the number of log records to be read.

Example

The following example specifies that a maximum of 45,000 log records are to be read:

```
MAXRECORDS 45ØØØ
```

OPTION Command

Use the OPTION command to enable certain options. All options are disenabled by default. This command can appear before or after the report-specification commands.

```
<option command>

           ┌←─────────────────────┐
— OPTION ──┴┬─/1\─ WRITEIODATA ───┴──────────────────────────|
            ├─/1\─ WRITEDRCDATA ─┤
            ├─/1\─ DRCONLY ──────┤
            ├─/1\─ RESETDRC ─────┤
            ├─/1\─ DEBUG ────────┤
            ├─/1\─ UPDATE ───────┤
            └─/1\─ YEAR ─────────┘
```

The following text describes the meaning of each variable:

WRITEIODATA	Creates the FILEIODATA file.
	WRITEIODATA is also TRUE if the SOURCE FILE IS FILEIODATA command appears in the report-specification commands. Refer to "SOURCE Command" for more information. The FILEIODATA file is not created if the USE CURRENT command appears in the input-specification commands. Refer to "USE Command" for more information.
WRITEDRCDATA	Creates the DRCDATA file. LOGGER creates this file using the disk resource control (DRC) information contained in the USERDATAFILE. The DRCDATA file is also created if the SOURCE report-specification command specifies that the DRCDATA file is to be used to generate the report.
DRCONLY	Prevents LOGGER from processing any system log files. LOGGER collects only disk resource control (DRC) data when this option is used, and any requests for JOBSUMMARY, STATISTICS, or FILEIODATA reports generate syntax errors.

continued

continued

RESETDRC	Causes LOGGER to reset the MBYTEDAYS value to zero for all disk resource control (DRC) system users who have DRC entries in the USERDATAFILE. MBYTEDAYS is explained in Table 8–4 "DRCDATA File Data Items" later in this section.

Note *If LOGGER is run with the RESETDRC option set, the value of MBYTEDAYS is set to zero for every packname entry currently in use by every usercode. This causes MBYTEDAYS to start accumulating statistics from zero again. For example, if LOGGER is run weekly with the RESETDRC option set, each run reports the disk space usage for the previous week, sets the value of MBYTEDAYS to zero, and then accumulates statistics until the next LOGGER run.*

When LOGGER has finished generating the reports and files, it sets its TASKVALUE task attribute to a value of 1. If the RESETDRC option was specified, the TASKVALUE task attribute is set before LOGGER starts setting the MBYTEDAYS values back to 0. You can use the TASKVALUE task attribute in a WFL job to recover properly the data generated by LOGGER should your system fail. The job can prevent LOGGER from restarting from the beginning and reporting MBYTEDAYS values that have already been reset to zero. An example of a WFL job that uses the TASKVALUE task attribute in recovery is given later under this heading.

DEBUG	Prints certain debugging information.
UPDATE	Updates the year-to-date file with the results of the current run. If no year-to-date file exists, UPDATE creates one. Report-specification commands must be present when this option is used.
YEAR	Generates a year-to-date report from the year-to-date file. After producing this report, LOGGER terminates the run and performs no other LOGGER updates or report generation. Because the format of the year-to-date report is inherent in the file, report-specification commands are not needed when using this option.

Examples

The following command enables the DEBUG and UPDATE options:

```
OPTION DEBUG UPDATE
```

The following is an example of a WFL job that can be used to restart LOGGER for a DRC report with the RESETDRC option set. This example is not complete and must be modified for individual systems. For information about issues involved in recovery, refer to the discussion of the RESETDRC option earlier under this heading.

```
BEGIN JOB LOGGER/DRC;
TASK T;
BOOLEAN RESETONLY;
SUBROUTINE SETDRC;
BEGIN
RUN SYSTEM/LOGGER;
DATA
OPTION DRCONLY RESETDRC
?%END
END; %SUBROUTINE
ON RESTART,
BEGIN
     IF RESETONLY THEN
        BEGIN
        SETDRC;
        GO ENDOFJOB;
        END
     ELSE
        GO JOBSTART;
END;
JOBSTART:
PROCESS RUN SYSTEM/LOGGER[T];
DATA
OPTION RESETDRC
SOURCE IS DRCDATA
REPORT USERCODE, PACKNAME, DISKINUSE, MBYTEDAYS
SORT BY USERCODE ASCENDING
REPORTS ARE SUMMARY 1
?%END DATA

WAIT(T(VALUE)=1);
RESETONLY := TRUE;
ENDOFJOB:
END JOB
```

SORT PARAMETERS Command

Use the SORT PARAMETERS command to change the parameters used by the SORT command when sorting the entries.

<sort parameters command>

```
                        ┌←─────────────────────────────┐
— SORT — PARAMETERS ────┬─/1\─ DISKSIZE ─<integer>─┬─┴──────────────┤
                        └─/1\─ TAPES ──<integer>───┘
```

The following text describes the meaning of each variable:

DISKSIZE <integer> — Specifies the maximum amount of disk to be used for the sort. The default is 180 million words.

Note: *The <integer> in the DISKSIZE option specifies the size in units of words, not sectors. A sector typically contains 30 words.*

continued

continued

TAPES <integer>	Specifies the number of tapes to be used for the sort. The default is 3. The number of tapes specified can range from 0 through 8. However, if the number of tapes is 1, 2, or 3, three tapes are used.

Example

The following command sets the disk size to 3,600,000 words and specifies that no tapes are to be used:

```
SORT PARAMETERS DISKSIZE 3600000 TAPES 0
```

STOP Command

Use the STOP command to indicate the end of the LOGGER input. LOGGER does not read any input-specification or report-specification commands specified after the STOP command. This command should be the last entry in the LOGGER input.

```
<stop command>

— STOP ————————————————————————————————————————————————————|
```

USE Command

The USE command specifies which source files are to be used for the LOGGER report. If you do not specify the USE command, LOGGER uses the current SYSTEM/SUMLOG on the current log family as the source file. LOGGER can access a SUMLOG file only if the log file title has the form SUMLOG/<system serial number>/<date>/<log number>, which is the form assigned to it automatically by the system during a log release process.

LOGGER accesses SUMLOG files through the SDASUPPORT library. Depending on the security status of the SUMLOG file and the privilege status of the usercode running LOGGER, the SDASUPPORT library might perform log filtering. For further information about log filtering, refer to Section 12, "SUMLOG."

The USE command can only appear once in an input deck. This command can appear before or after a report-specification command.

```
<use command>

— USE ─┬──────────┬─┬─ CURRENT ─────────────┬──────────────────────|
       ├─ SUMLOG ─┘ └─<mmddyy>─┬────────────┤
       │                       └─<mmddyy>───┤
       └─ SYSTEM/SUMLOG ────────────────────┘
```

The following text describes the meaning of each variable:

SUMLOG <mmddyy>	
SUMLOG <mmddyy> <mmddyy>	Indicates that system logs released through a TL (Transfer Log) system command are to be used as the source files. Refer to the System Commands Reference Manual for additional information about the TL command. If two <mmddyy> dates are specified, the files between those dates, inclusive, are used.
SUMLOG CURRENT	Same as SUMLOG <mmddyy>, where <mmddyy> is the current date.
CURRENT	Indicates that the JOBSUMMARY, STATISTICS, DRCDATA, or FILEIODATA file with the current date is to be used as the source file. New files are not created.
<mmddyy>	
<mmddyy> <mmddyy>	Use any valid date where mm is the month, dd is the day, and yy is the year. This form of the command specifies that the JOBSUMMARY, STATISTICS, DRCDATA, or FILEIODATA file from the specified date is used. If two <mmddyy> dates are specified, the files between those dates, inclusive, are used.
SYSTEM/SUMLOG	Indicates SYSTEM/SUMLOG is to be used as the source file.

Example

The following command specifies that SUMLOG files dated with 030281 are to be used as source files:

```
USE SUMLOG Ø3Ø281
```

Report-Specification Commands

The report-specification commands specify the format and content of reports.

Only the first 72 characters of an input record are used for report-specification commands. A hyphen (-) used after text and before column 72 indicates that the following input record is a continuation of the current command.

Additional examples that demonstrate the effect of combining certain report-specification commands, are provided in “Report-Specification Command Examples” in this section.

The following report-specification commands can be used as input to LOGGER.

```
<report-specification command>

──┬─<break command>──────┬──────────────────────────────────────────┤
  ├─<end command>────────┤
  ├─<exclude command>────┤
  ├─<heading command>────┤
  ├─<include command>────┤
  ├─<output command>─────┤
  ├─<page size command>──┤
  ├─<report command>─────┤
  ├─<reports command>────┤
  ├─<sort command>───────┤
  └─<source command>─────┘
```

BREAK Command

Use the BREAK command to specify where to break to calculate and print the requested totals and averages.

```
<break command>

                     ┌←── , ──┐
── BREAK ─┬──────┬───┴─<item>─┴─┬────────────────────┬──────────────┤
          └─ ON ─┘              └─<break options>────┘

<break options>

  ┌←─/1\──────────────────────────────────┐
──┴─┬─/1\─ TOTALING ──┬──┬←── , ──┐───────┴───────────────────────────┤
    └─/1\─ AVERAGING ─┘  └─<item>─┘
```

The following text describes the meaning of each variable:

<item>	Indicates a valid JOBSUMMARY, STATISTICS, DRCDATA, or FILEIODATA file data item. Refer to "File Data Items" in this section for more information. When more than one file data item is specified, the items are taken in order from left to right with the rightmost item specifying the innermost break.
<break options>	The specified totals or averages or both are accumulated and printed each time the value of the break item changes. Both totaling and averaging can be performed on the same item in the same detail report. Either totaling or averaging can be performed on an item in a SUMMARY or YTD report. If both totaling and averaging are specified for the same item in a SUMMARY or YTD report, only the totals are printed.

Examples

The following command specifies the NAME file data item as a control break:

```
BREAK ON NAME
```

The following command specifies totals for TASKS, JOBS, and HL to be printed for the DATE file data item each time the value of the item changes:

```
BREAK ON DATE TOTALING TASKS, JOBS, HL
```

The following command specifies totals for PROCESSTIME and IOTIME and averages for IOTIME be printed for the USERCODE file data item each time the value of the item changes:

```
BREAK ON USERCODE TOTALING PROCESSTIME, IOTIME AVERAGING IOTIME
```

END Command

Use the END command to indicate the end of the report-specification commands. This command must appear at the end of the report-specification commands.

```
<end command>

— END ————————————————————————————————————————————————————————|
```

EXCLUDE Command

Use the EXCLUDE command to specify the records that are to be excluded from the report. As many EXCLUDE commands as desired can appear in the report specifications. Excluded items are ORed. When the EXCLUDE and INCLUDE commands are used together, the results are ANDed. Refer to "INCLUDE Command" for more information.

```
<exclude command>

— EXCLUDE ─┬──────────┬─┬──────┬─<item>─┬─ = ───┬─┬─<integer>───────┬─|
           └─ RECORD ─┘ └─ IF ─┘        ├─ > ───┤ └─<quoted string>─┘
                                        ├─ < ───┤
                                        ├─ NEQ ─┤
                                        ├─ LEQ ─┤
                                        └─ GEQ ─┘
```

The following text describes the meaning of each variable:

<item>	Specifies a valid JOBSUMMARY, STATISTICS, DRCDATA, or FILEIODATA file data item. Refer to "File Data Items" in this section for more information. For items of type string, the length of the comparison made to select records is the minimum of the length of the quoted string and the actual length of the item.
=	Excludes all items equal to the integer or quoted string from the report.
>	Excludes all items greater than the integer or quoted string from the report.
<	Excludes all items less than the integer or quoted string from the report.

continued

continued

NEQ	Excludes all items not equal to the integer or quoted string from the report.
LEQ	Excludes all items less than or equal to the integer or quoted string from the report.
GEQ	Excludes all items greater than or equal to the integer or quoted string from the report.
<integer>	Any valid real or integer number corresponding to a numeric file data item.
<quoted string>	A quoted string is data enclosed in quotation marks. The syntax is similar to an ALGOL quoted string not a WFL quoted string. The data can be null, and the data can contain quotation marks. Two successive quotation marks signify a null string, and three quotation marks signify a quotation mark within a quoted string. Quotation marks can be single (') or double (") as long as the beginning and end pair match. Embedded quotations must use quotation marks that are not the same as the delimiter quotation marks. For example, "*A 'quotation' in the string*" is valid. Quoted strings cannot be continued to the next input record.

Examples

The following command specifies that only records with the usercode MAVO are to be printed in the report:

```
EXCLUDE RECORD IF USERCODE NEQ "MAVO"
```

The following command specifies that all records with the chargecode SKIP are to be excluded from the report:

```
EXCLUDE IF CHARGECODE = "SKIP"
```

The following command specifies that all records with the queue number 7 are to be excluded from the report. Note that the QUEUE data item is padded with leading zeros to make its length three digits.

```
EXCLUDE IF QUEUE = "ØØ7"
```

HEADING Command

Use the HEADING command to specify the heading that appears at the top center of each page. If more than one HEADING command is present in the report specifications, only the last one is used.

```
<heading command>

— HEADING ─┬──────┬─ <quoted string> ──────────────────────────┤
           └─ IS ─┘
```

The following text describes the meaning of the variable:

<quoted string>	A quoted string is data enclosed in quotation marks. The syntax is similar to an ALGOL quoted string, not a WFL quoted string. The data can be null, and the data can contain quotation marks. Two successive quotation marks signify a null string, and three quotation marks signify a quotation mark within a quoted string. Quotation marks can be single (') or double (") as long as the beginning and end pair match. Embedded quotations must use quotation marks that are not the same as the delimiter quotation marks. For example, "*A 'quotation' in the string*" is valid. Quoted strings cannot be continued to the next input record.

Example

The following command places the heading JOB-TASK-SESSION BY USERCODE at the top center of each page of the report:

```
HEADING IS "JOB-TASK-SESSION BY USERCODE"
```

INCLUDE Command

Use the INCLUDE command to indicate what records are to be included in the report. Any item not specified in this command is excluded from the report. As many INCLUDE commands as desired can appear in the report-specification commands. Items that are included are ORed. When INCLUDE and EXCLUDE commands are used together, the results are ANDed. Refer to "EXCLUDE Command" for more information.

```
<include command>

— INCLUDE ──┬──────────┬─┬──────┬─<item>─┬─ =  ──┬─┬─<integer>───────┬─┤
            └─ RECORD ─┘ └─ IF ─┘        ├─ >  ──┤ └─<quoted string>─┘
                                         ├─ <  ──┤
                                         ├─ NEQ ─┤
                                         ├─ LEQ ─┤
                                         └─ GEQ ─┘
```

The following text describes the meaning of each variable:

<item>	Specifies a valid JOBSUMMARY, STATISTICS, DRCDATA, or FILEIODATA file data item. Refer to "File Data Items" in this section for more information. For items of type string, the length of the comparison made to select records is the minimum of the length of the quoted string and the actual length of the item.
=	Includes the items equal to the integer or quoted string.
>	Includes the items greater than the integer or quoted string.
<	Includes the items less than the integer or quoted string.
NEQ	Includes the items not equal to the integer or quoted string.
LEQ	Includes the items less than or equal to the integer or quoted string.

continued

continued

GEQ	Includes the items greater than or equal to the integer or quoted string.
<integer>	Any valid real or integer number corresponding to numeric file data items.
<quoted string>	A quoted string is data enclosed in quotation marks. The syntax is similar to an ALGOL quoted string, not a WFL quoted string. The data can be null, and the data can contain quotation marks. Two successive quotation marks signify a null string, and three quotation marks signify a quotation mark within a quoted string. Quotation marks can be single (') or double (") as long as the beginning and end pair match. Embedded quotations must use quotation marks that are not the same as the delimiter quotation marks. For example, "*A 'quotation' in the string*" is valid. Quoted strings cannot be continued to the next input record.

Examples

The following command specifies that only records with the date 02/23/82 are to be included in the report:

```
INCLUDE RECORD IF DATE = "Ø2/23/82"
```

The following command specifies that only records with the type J are to be included in the report:

```
INCLUDE IF TYPE = "J"
```

The following command specifies that only records with the queue number 7 are to be included in the report. Note that the QUEUE data item is padded with leading zeros to make its length three digits.

```
INCLUDE IF QUEUE = "ØØ7"
```

OUTPUT Command

Use the OUTPUT command to specify items that are to appear in the report.

```
<output command>

                                      ┌←─── , ───┐
── OUTPUT ─┬─────────┬─┬───────┬──────┴─ <item> ─┴──────────────────────┤
           └─ ITEMS ─┘ └─ ARE ─┘
```

The following text describes the meaning of the variable:

<item>	Specifies a valid JOBSUMMARY, STATISTICS, DRCDATA, or FILEIODATA file data item. Refer to "File Data Items" in this section for more information.

Example

The following command specifies that the TYPE, MIXNO, JOBNO, and PROCESSTIME file data items are to appear in the report. Because these items are all valid JOBSUMMARY file data items, it is assumed that a SOURCE FILE IS JOBSUMMARY command is present in the report specifications. Refer to "SOURCE Command" for more information.

```
OUTPUT ITEMS ARE TYPE, MIXNO, JOBNO, PROCESSTIME
```

PAGE SIZE Command

Use the PAGE SIZE command to specify the number of lines per page for the report. If no PAGE SIZE command is entered, the number of lines per page is 60.

```
<page size command>

— PAGE ┬────────┬┬──────┬<integer>─────────────────────────┤
       └─ SIZE ─┘└─ IS ─┘
```

The following text describes the meaning of the variable:

<integer>	Indicates the number of lines on a page.

Example

The following command specifies that each page of the report will contain 72 lines:

```
PAGE SIZE IS 72
```

REPORT Command

Use the REPORT command to indicate the beginning of the report specifications. This command can only appear once in the input deck.

```
<report command>

— REPORT ┬───────────┬─────────────────────────────────────┤
         └─<integer>─┘
```

The following text describes the meaning of the variable:

<integer>	Identifies the number of the report in the LOGREPORTS file. The LOGREPORTS file can contain many report specifications and is used for installation convenience. Refer to "REPORT Commands and LOGREPORTS File" in this section for more information.

Example

The following command specifies that the report specifications for the report are to be found in the LOGREPORTS file under report number 4:

```
REPORT 4
```

REPORTS Command

Use the REPORTS command to specify whether the type of report to be printed is a detailed report, summary report, or both.

If this command is not present in the report specifications, a detailed report is produced.

```
<reports command>

— REPORTS ─┬───────┬─┬────────────┬─┬─────────────────────────────┬──────┤
           └─ ARE ─┘ └─ DETAILED ─┘ │ ┌←────────────────────────┐ │
                                    └─┴─ SUMMARY ─<integer>─────┴─┘
```

The following text describes the meaning of each variable:

DETAILED	Specifies one detailed report. A detailed report includes one line for each record in the file.
SUMMARY <integer>	Specifies one summary report. A summary report consists of one line for a particular group of records in the file. If summary reports are requested, the REPORTS command must be preceded by at least as many BREAK commands as there are summary reports to be printed. Refer to "BREAK Command" in this section for more information. The integer indicates the control break item to be summarized when more than one BREAK command is specified in the report specifications.

Examples

The following command specifies a summary report with totals to be generated for the control break item specified in the second BREAK command:

```
REPORTS ARE DETAILED SUMMARY 2
```

The following command specifies a summary report to be generated for the control break items specified in the first and second BREAK commands:

```
REPORTS ARE DETAILED SUMMARY 1 SUMMARY 2
```

SORT Command

Use the SORT command to specify the sort sequence for the output items.

```
<sort command>

                     ┌←──────────── , ──────────────┐
— SORT ─┬──────┬─────┴─ <item> ─┬──────────────┬────┴───────────────────┤
        └─ BY ─┘                ├─ ASCENDING ──┤
                                └─ DESCENDING ─┘
```

The following text describes the meaning of each variable:

<item>	Specifies a valid JOBSUMMARY, STATISTICS, DRCDATA, or FILEIODATA file data item. Refer to "File Data Items" in this section for more information. If more than one item is specified, sorting begins with the first item. When the first item is equal, the sort moves to the next item and continues in order until all items are sorted.
ASCENDING	Sorts the items from the lowest to the highest order. The default order is ASCENDING.
DESCENDING	Sorts the items from the highest to the lowest order.

Examples

The following command indicates the report is to be sorted by job number in ascending order:

```
SORT BY JOBNO ASCENDING
```

The following command indicates the report is to be sorted in in ascending order first by usercode, and then by mix number, within each usercode:

```
SORT BY USERCODE, MIXNO
```

SOURCE Command

Use the SOURCE command to specify whether a JOBSUMMARY, STATISTICS, DRCDATA, or FILEIODATA file is to be used to generate the report. Only one SOURCE command can appear in the input deck. This command must precede all other report-specification commands except the REPORT command.

```
<source command>

— SOURCE ─┬────────┬┬──────┬┬─ JOBSUMMARY ─┬──────────────────────┤
          └─ FILE ─┘└─ IS ─┘├─ STATISTICS ─┤
                            ├─ DRCDATA ────┤
                            └─ FILEIODATA ─┘
```

The following text describes the meaning of each variable:

JOBSUMMARY	Indicates a JOBSUMMARY file is to be used to generate the report.
STATISTICS	Indicates a STATISTICS file is to be used to generate the report.
DRCDATA	Indicates a DRCDATA file is to be used to generate the report.
FILEIODATA	Indicates a FILEIODATA file is to be used to generate the report.

Example

The following command indicates the FILEIODATA file is to be used to generate the report. This command also enables the WRITEIODATA option. Refer to "OPTION Command" in this section for more information.

```
SOURCE FILE IS FILEIODATA
```

Report-Specification Command Examples

The following examples illustrate the use of report-specification commands. Detailed explanations of the individual commands are in "Report-Specification Commands" in this section. A sample of the report produced by each example is shown in Figures 8–1 through 8–8.

Example 1

In the following example, the REPORT command signifies the beginning of the report specifications. The SOURCE command selects the JOBSUMMARY file as the source of information for the report. The OUTPUT command selects the TYPE, MIXNO, JOBNO, NAME, PRIORITY, ORGUNIT, CHARGECODE, PROCESSTIME, and IOTIME file data items to be printed. The END command signifies the end of the report-specification commands.

```
REPORT
SOURCE IS JOBSUMMARY
OUTPUT ITEMS ARE TYPE, MIXNO, JOBNO, NAME, PRIORITY, ORGUNIT,-
  CHARGECODE, PROCESSTIME, IOTIME
END
```

Figure 8–1 shows the resulting report.

04/04/90

TYPE	MIXNO	JOBNO	NAME	PRIORITY	ORGUNIT	CHARGECODE	PROCESSTIME	IOTIME
T	9927	9851	(WBJ)OBJECT/S/TASK/CHARGECODE	50	LSN:00602	CHARGE.	0.01	0.00
J	9923	9923	(WBJ)OBJECT/S/TASK/CHARGECODE/	50	LSN:00602	CHARGE.	0.01	0.05
T	9928	9851	(WBJ)OBJECT/S/TASK/CHARGECODE	50	LSN:00602	CHARGE.	0.01	0.00
J	9922	9922	"UPDATE_FAULTLOG".	80	UNIT:00000		0.02	0.07
J	9926	9926	"UPDATE_FAULTLOG".	80	UNIT:00000		0.02	0.00
T	9931	9851	(WBJ)OBJECT/S/TASK/CHARGECODE/	50	LSN:00602	CHARGE.	0.01	0.03
T	9932	9851	(WBJ)OBJECT/S/TASK/CHARGECODE/	50	LSN:00602	CHARGE.	0.01	0.00
T	9924	9851	(WBJ)OBJECT/S/TASK/UTIL/LIBRAR	50	LSN:00602	CHARGE.	0.08	0.07
J	9933	9933	(WBJ)OBJECT/S/TASK/CHARGECODE/	50	LSN:00602	CHARGE.	0.01	0.03
T	9934	9851	(WBJ)OBJECT/S/TASK/UTIL/LIBRAR	50	LSN:00602	6685.	0.04	0.00
T	9935	9851	(WBJ)OBJECT/S/TASK/UTIL/LIBRAR	50	LSN:00602	6685.	0.04	0.00
T	9938	9936	SYSTEM/FAULTLOGGER.	50	UNIT:00000		0.04	0.05
J	9936	9936	"BEGIN JOB;RUN SYS".	50	UNIT:00000		0.03	0.10
T	9939	9937	SYSTEM/FAULTLOGGER.	50	UNIT:00000		0.03	0.05
J	9937	9937	"BEGIN JOB;RUN SYS".	50	UNIT:00000		0.03	0.11
T	9916	9851	(WBJ)OBJECT/S/TASK/CHARGECODE	50	LSN:00602	6685.	0.25	0.13
T	9940	9851	SYSTEM/PATCH.	50	LSN:00602	6685.	0.30	1.24
T	9941	9851	SYSTEM/ALGOL.	50	LSN:00602	6685.	0.85	1.08
T	9942	9851	SYSTEM/PATCH.	50	LSN:00602	6685.	0.30	1.19
T	9943	9851	SYSTEM/ALGOL.	50	LSN:00602	6685.	0.67	0.91
T	9945	9851	(WBJ)OBJECT/S/TASK/CONVENTION	50	LSN:00602	6685.	0.01	0.00
T	9946	9851	(WBJ)OBJECT/S/TASK/CONVENTION	50	LSN:00602	6685.	0.01	0.00
T	9947	9851	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.00
T	9948	9851	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.00
J	9949	9949	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.03
T	9951	9851	(WBJ)OBJECT/S/TASK/CONVENTION	50	LSN:00602	6685.	0.01	0.00
T	9952	9851	(WBJ)OBJECT/S/TASK/CONVENTION	50	LSN:00602	6685.	0.01	0.00
T	9953	9851	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.00
T	9954	9851	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.00
T	9950	9851	(WBJ)OBJECT/S/TASK/UTIL/LIBRAR	50	LSN:00602	6685.	0.08	0.04
J	9955	9955	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.09
T	9944	9851	(WBJ)OBJECT/S/TASK/CONVENTION	50	LSN:00602	6685.	0.40	0.19
T	9957	9851	(WBJ)OBJECT/S/TASK/CONVENTION	50	LSN:00602	6685.	0.01	0.05
T	9958	9851	(WBJ)OBJECT/S/TASK/CONVENTION	50	LSN:00602	6685.	0.01	0.04
T	9959	9851	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.05
T	9960	9851	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.04
J	9961	9961	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.03
T	9963	9851	(WBJ)OBJECT/S/TASK/CONVENTION	50	LSN:00602	6685.	0.01	0.04
T	9964	9851	(WBJ)OBJECT/S/TASK/CONVENTION	50	LSN:00602	6685.	0.01	0.04
T	9965	9851	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.01
T	9966	9851	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.01
T	9962	9851	(WBJ)OBJECT/S/TASK/UTIL/LIBRAR	50	LSN:00602	6685.	0.08	0.14
J	9967	9967	(WBJ)OBJECT/S/TASK/CONVENTION/	50	LSN:00602	6685.	0.01	0.04
T	9956	9851	(WBJ)OBJECT/S/TASK/CONVENTION	50	LSN:00602	6685.	0.43	0.97
T	9969	9851	SYSTEM/PATCH.	50	LSN:00602	6685.	0.30	1.28
T	9970	9851	SYSTEM/ALGOL.	50	LSN:00602	6685.	0.80	1.14
T	9971	9851	SYSTEM/PATCH.	50	LSN:00602	6685.	0.30	1.17
T	9975	9851	SYSTEM/ALGOL.	50	LSN:00602	6685.	0.67	1.04
T	9977	9851	(WBJ)OBJECT/S/TASK/DESTNAME ON	50	LSN:00602	6685.	0.01	0.00
T	9978	9851	(WBJ)OBJECT/S/TASK/DESTNAME ON	50	LSN:00602	6685.	0.01	0.00
T	9979	9851	(WBJ)OBJECT/S/TASK/DESTNAME/EX	50	LSN:00602	6685.	0.01	0.00
T	9980	9851	(WBJ)OBJECT/S/TASK/DESTNAME/EX	50	LSN:00602	6685.	0.01	0.00
J	9981	9981	(WBJ)OBJECT/S/TASK/DESTNAME/EX	50	LSN:00602	6685.	0.02	0.05
T	9983	9851	(WBJ)OBJECT/S/TASK/DESTNAME ON	50	LSN:00602	6685.	0.01	0.00
T	9984	9851	(WBJ)OBJECT/S/TASK/DESTNAME ON	50	LSN:00602	6685.	0.01	0.00

Figure 8–1. LOGGER Example 1 Output

Example 2

In the following example, the REPORT command signifies the beginning of the report specifications. The SOURCE command selects the JOBSUMMARY file as the source of information for the report. The HEADING command specifies *LOG SORTED BY TIME* to be used as the heading for the report.

The SORT command specifies that the report is sorted by the STARTTIME file data item. The OUTPUT command selects the MIXNO, TYPE, STARTTIME, NAME, USERCODE, PRIORITY, ELAPSEDTIME, and TERMCOND file data items to be printed. The END command signifies the end of the report-specification commands.

```
REPORT
SOURCE FILE IS JOBSUMMARY
HEADING IS "LOG SORTED BY TIME"
SORT BY STARTTIME
OUTPUT ITEMS ARE MIXNO, TYPE, STARTTIME, NAME, USERCODE,-
  PRIORITY, ELAPSEDTIME, TERMCOND
END
```

Figure 8–2 shows the resulting report.

LOG SORTED BY TIME
04/04/90

MIXNO	TYPE	STARTTIME	NAME	USERCODE	PRIORITY	ELAPSEDTIME	TERMCOND
7444	J	08:12:39	(LEEL)SYSTEM/RUNMAIL ON PACK.	JBCOXX.			
8281	S	10:41:41	-MCS SESSION-	PWTODD.	00	291.49	
9714	T	14:40:41	OBJECT/ED.	ACWAN.			
9828	T	15:07:25	STATUS/CHANGE/LFA15CD.		75	18.94	
9849	T	15:12:56	OBJECT/ED ON MCPMAST.	DUNKINS.			
9851	J	15:13:07	TASKTEST.	WBF.	50	20.96	
9857	T	15:13:15	(ACWAN)OBJECT/TESTDCS ON DC.	ACWAN.	50	10.98	
9859	T	15:13:16	(ACWAN)DATACOMSUPPORT/9.	ACWAN.	50	10.96	
9858	J	15:13:16	(ACWAN)OBJECT/DCS/PATCH ON DC.	ACWAN.	50	10.97	
9916	T	15:14:45	(WBF)OBJECT/S/TASK/CHARGECODE	WBF.	50	0.32	
9924	T	15:14:52	(WBF)OBJECT/S/TASK/UTIL/LIBRAR	WBF.	50	0.04	
9926	J	15:14:52	"UPDATE_FAULTLOG".		80	0.00	
9922	J	15:14:52	"UPDATE_FAULTLOG".		80	0.00	
9928	T	15:14:52	(WBF)OBJECT/S/TASK/CHARGECODE	WBF.	50	0.00	
9923	J	15:14:52	(WBF)OBJECT/S/TASK/CHARGECODE/	WBF.	50	0.00	
9927	T	15:14:52	(WBF)OBJECT/S/TASK/CHARGECODE	WBF.	50	0.00	
9931	T	15:14:53	(WBF)OBJECT/S/TASK/CHARGECODE/	WBF.	50	0.00	
9932	T	15:14:53	(WBF)OBJECT/S/TASK/CHARGECODE/	WBF.	50	0.00	
9935	T	15:14:54	(WBF)OBJECT/S/TASK/UTIL/LIBRAR	WBF.	50	0.00	P-DSED
9934	T	15:14:54	(WBF)OBJECT/S/TASK/UTIL/LIBRAR	WBF.	50	0.00	P-DSED
9933	J	15:14:54	(WBF)OBJECT/S/TASK/CHARGECODE/	WBF.	50	0.00	
9938	T	15:14:55	SYSTEM/FAULTLOGGER.		50	0.00	
9937	J	15:14:55	"BEGIN JOB;RUN SYS".		50	0.01	
9939	T	15:14:55	SYSTEM/FAULTLOGGER.		50	0.00	
9936	J	15:14:55	"BEGIN JOB;RUN SYS".		50	0.01	
9940	T	15:15:06	SYSTEM/PATCH.	WBF.	50	0.03	
9941	T	15:15:08	SYSTEM/ALGOL.	WBF.	50	0.04	
9942	T	15:15:11	SYSTEM/PATCH.	WBF.	50	0.03	
9943	T	15:15:13	SYSTEM/ALGOL.	WBF.	50	0.03	
9944	T	15:15:15	(WBF)OBJECT/S/TASK/CONVENTION	WBF.	50	0.26	
9946	T	15:15:16	(WBF)OBJECT/S/TASK/CONVENTION	WBF.	50	0.00	
9945	T	15:15:16	(WBF)OBJECT/S/TASK/CONVENTION	WBF.	50	0.00	
9948	T	15:15:17	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9947	T	15:15:17	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9950	T	15:15:18	(WBF)OBJECT/S/TASK/UTIL/LIBRAR	WBF.	50	0.04	
9952	T	15:15:18	(WBF)OBJECT/S/TASK/CONVENTION	WBF.	50	0.00	
9951	T	15:15:18	(WBF)OBJECT/S/TASK/CONVENTION	WBF.	50	0.00	
9949	J	15:15:18	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9953	T	15:15:19	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9954	T	15:15:19	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9955	J	15:15:20	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9956	T	15:15:31	(WBF)OBJECT/S/TASK/CONVENTION	WBF.	50	0.29	
9958	T	15:15:32	(WBF)OBJECT/S/TASK/CONVENTION	WBF.	50	0.00	
9957	T	15:15:32	(WBF)OBJECT/S/TASK/CONVENTION	WBF.	50	0.00	
9960	T	15:15:33	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9959	T	15:15:33	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9964	T	15:15:35	(WBF)OBJECT/S/TASK/CONVENTION	WBF.	50	0.00	
9963	T	15:15:35	(WBF)OBJECT/S/TASK/CONVENTION	WBF.	50	0.00	
9961	J	15:15:35	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9962	T	15:15:35	(WBF)OBJECT/S/TASK/UTIL/LIBRAR	WBF.	50	0.05	
9966	T	15:15:36	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9965	T	15:15:36	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9967	J	15:15:37	(WBF)OBJECT/S/TASK/CONVENTION/	WBF.	50	0.00	
9969	T	15:15:49	SYSTEM/PATCH.	WBF.	50	0.03	
9970	T	15:15:51	SYSTEM/ALGOL.	WBF.	50	0.04	

Figure 8–2. LOGGER Example 2 Output

Example 3

In the following example, the REPORT command signifies the beginning of the report specifications. The SOURCE command selects the JOBSUMMARY file as the source of information for the report.

The BREAK command specifies the USERCODE file data item to act as a control break. Each usercode is printed on a separate line every time the value of the control break item changes.

The SORT command specifies sorting by more than one file data item. The output is sorted in ascending order first by USERCODE and then by MIXNO.

The OUTPUT command selects the TYPE, MIXNO, NAME, LINES, AVGCORECODE, AVGCOREDATA, DATE, PRIORITY, and ORGUNIT file data items to be printed.

The HEADING command causes the heading *JOB/TASK/SESSION SUMMARY BY USERCODE* to be printed at the top of every page. The PAGE SIZE command specifies that 56 lines are to be printed on each page. The END command signifies the end of the report-specification commands.

```
REPORT
SOURCE FILE IS JOBSUMMARY
BREAK ON USERCODE
SORT BY USERCODE, MIXNO ASCENDING
OUTPUT ITEMS ARE TYPE, MIXNO, NAME, LINES, AVGCORECODE,-
  AVGCOREDATA, DATE, PRIORITY, ORGUNIT
HEADING IS "JOB/TASK/SESSION SUMMARY BY USERCODE"
PAGE SIZE IS 56
END
```

Figure 8–3 shows the resulting report.

JOB/TASK/SESSION SUMMARY BY USERCODE
04/04/90

USERCODE:

TYPE	MIXNO	NAME	LINES	AVGCORECODE	AVGCOREDATA	DATE	PRIORITY	ORGUNIT
J	0108	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	114	984	04/04/90	50	LSN:00488
J	0109	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	135	994	04/04/90	50	LSN:00488
J	0120	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	142	995	04/04/90	50	LSN:00488
J	0123	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	143	990	04/04/90	50	LSN:00488
J	0162	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	120	995	04/04/90	50	LSN:00488
J	0183	"BEGIN JOB:RUN SYS".	0	75	1201	04/04/90	50	UNIT:00002
T	0184	SYSTEM/FILEDATA.	39	4025	9028	04/04/90	50	UNIT:00002
J	0186	SERVER/LP5/"R#7511"/"J#0183".	0	0	4609	04/04/90	80	UNIT:00000
J	0203	SERVER/LP5/"R#7512"/"J#9989".	0	0	5010	04/04/90	80	UNIT:00000
J	0221	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	128	999	04/04/90	50	LSN:00488
J	0234	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	144	991	04/04/90	50	LSN:00488
J	0277	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	127	994	04/04/90	50	LSN:00488
J	0282	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	144	998	04/04/90	50	LSN:00488
J	0292	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	999	04/04/90	50	LSN:00488
J	0296	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	979	04/04/90	50	LSN:00488
J	0299	"BEGIN JOB:COPY = ".	0	53	910	04/04/90	50	UNIT:00002
T	0300	LIBRARY/MAINTENANCE.	0	0	22312	04/04/90	50	UNIT:00002
J	0313	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	983	04/04/90	50	LSN:00488
J	0317	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	995	04/04/90	50	LSN:00488
J	0327	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	992	04/04/90	50	LSN:00488
J	0333	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	995	04/04/90	50	LSN:00488
J	0352	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	997	04/04/90	50	LSN:00488
J	0365	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	990	04/04/90	50	LSN:00488
J	0380	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	998	04/04/90	50	LSN:00488
J	0383	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	984	04/04/90	50	LSN:00488
J	0394	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	985	04/04/90	50	LSN:00488
J	0398	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	989	04/04/90	50	LSN:00488
J	0415	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	997	04/04/90	50	LSN:00488
J	0428	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	994	04/04/90	50	LSN:00488
J	0458	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	997	04/04/90	50	LSN:00488
J	0462	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	992	04/04/90	50	LSN:00488
J	0473	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	987	04/04/90	50	LSN:00488
J	0478	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	982	04/04/90	50	LSN:00488
J	0484	R/UTILITY/PHONE.	0	2945	7211	04/04/90	50	LSN:00488
J	0495	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	992	04/04/90	50	LSN:00488
J	0498	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	994	04/04/90	50	LSN:00488
J	0500	"BEGIN JOB:RUN SYS".	0	75	1235	04/04/90	50	UNIT:00001
T	0501	SYSTEM/FILEDATA.	75	3667	9485	04/04/90	50	UNIT:00001
J	0503	SERVER/LP5/"R#7537"/"J#0500".	0	0	4735	04/04/90	80	UNIT:00000
J	0515	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	990	04/04/90	50	LSN:00488
J	0519	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	993	04/04/90	50	LSN:00488
J	0539	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	145	989	04/04/90	50	LSN:00488
T	0601	MAIL/REMINDER.	0	101552	2883	04/04/90	60	UNIT:00000
T	0613	SUPPRESS/MIX/NOS ON PACK.	0	157	1451	04/04/90	50	UNIT:00000
J	0630	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	129	988	04/04/90	50	LSN:00488
J	0712	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	131	999	04/04/90	50	LSN:00488
J	0716	(KLM)OBJECT/SYMBOL/OKI/TRANSFO	0	143	998	04/04/90	50	LSN:00488
J	0741	SERVER/LP5/"R#7545"/"J#8281".	0	0	5132	04/04/90	80	UNIT:00000

Figure 8–3. LOGGER Example 3 Output

Example 4

In the following example, the REPORT command signifies the beginning of the report specifications. The SOURCE command selects the JOBSUMMARY file as the source of information for the report.

The SORT command specifies sorting by more than one file data item. The output is sorted in ascending order by NAME, USERCODE, and STARTTIME.

The BREAK command specifies the NAME file data item to act as a control break. Each name is printed on a separate line each time the value of the control break item changes.

The INCLUDE and EXCLUDE commands specify that only MCS sessions and jobs of user SITE that do not have a chargecode of SKIP are processed for the report. The HEADING command specifies *REPORT FOR USERCODE DBH* to be used as the heading for the report.

The OUTPUT command selects the TYPE, MIXNO, NAME, LINES, AVGCORECODE, AVGCOREDATA, DATE, PRIORITY, and ORGUNIT file data items to be printed. The END command signifies the end of the report-specification commands.

```
REPORT
SOURCE JOBSUMMARY
SORT BY NAME, USERCODE, STARTTIME
BREAK ON NAME
INCLUDE RECORD IF TYPE = "S"
INCLUDE RECORD IF TYPE = "J"
EXCLUDE RECORD IF USERCODE NEQ "SITE"
EXCLUDE RECORD IF CHARGECODE = "SKIP"
HEADING IS "REPORT FOR USERCODE DBH"
OUTPUT ITEMS ARE TYPE, MIXNO, NAME, STARTTIME, ELAPSEDTIME
END
```

Figure 8–4 shows the resulting report.

```
                                                        REPORT FOR USERCODE DBH
                                                                04/04/90
TYPE   MIXNO   NAME                                     STARTTIME      ELAPSEDTIME

  NAME:  (DBH)OBJECT/S/TASK/CHARGECODE/

J       9923   (DBH)OBJECT/S/TASK/CHARGECODE/           15:14:52             0.00
J       9933   (DBH)OBJECT/S/TASK/CHARGECODE/           15:14:54             0.00
  NAME:  (DBH)OBJECT/S/TASK/CONVENTION/

J       9949   (DBH)OBJECT/S/TASK/CONVENTION/           15:15:18             0.00
J       9955   (DBH)OBJECT/S/TASK/CONVENTION/           15:15:20             0.00
J       9961   (DBH)OBJECT/S/TASK/CONVENTION/           15:15:35             0.00
J       9967   (DBH)OBJECT/S/TASK/CONVENTION/           15:15:37             0.00
  NAME:  (DBH)OBJECT/S/TASK/DESTNAME/EX

J       9981   (DBH)OBJECT/S/TASK/DESTNAME/EX           15:16:01             0.00
J       9987   (DBH)OBJECT/S/TASK/DESTNAME/EX           15:16:03             0.00
  NAME:  (DBH)OBJECT/S/TASK/DISKLIMIT/E

J       0100   (DBH)OBJECT/S/TASK/DISKLIMIT/E           15:16:32             0.01
J       0107   (DBH)OBJECT/S/TASK/DISKLIMIT/E           15:16:37             0.02
J       0115   (DBH)OBJECT/S/TASK/DISKLIMIT/E           15:16:50             0.01
J       0122   (DBH)OBJECT/S/TASK/DISKLIMIT/E           15:16:57             0.02
  NAME:  (DBH)OBJECT/S/TASK/DISPLAYONLY

J       0137   (DBH)OBJECT/S/TASK/DISPLAYONLY           15:17:31             0.00
J       0143   (DBH)OBJECT/S/TASK/DISPLAYONLY           15:17:34             0.00
  NAME:  (DBH)OBJECT/S/TASK/ELAPSEDLIMI

J       0155   (DBH)OBJECT/S/TASK/ELAPSEDLIMI           15:18:08             0.02
J       0163   (DBH)OBJECT/S/TASK/ELAPSEDLIMI           15:18:10             0.00
  NAME:  (DBH)OBJECT/S/TASK/ELAPSEDTIME

J       0175   (DBH)OBJECT/S/TASK/ELAPSEDTIME           15:18:42             0.03
J       0181   (DBH)OBJECT/S/TASK/ELAPSEDTIME           15:18:53             0.03
  NAME:  (DBH)OBJECT/S/TASK/FAMILY/EXTE

J       0197   (DBH)OBJECT/S/TASK/FAMILY/EXTE           15:19:18             0.00
J       0204   (DBH)OBJECT/S/TASK/FAMILY/EXTE           15:19:20             0.00
  NAME:  (DBH)OBJECT/S/TASK/FILEACCESSR

J       0214   (DBH)OBJECT/S/TASK/FILEACCESSR           15:19:44             0.00
J       0220   (DBH)OBJECT/S/TASK/FILEACCESSR           15:19:47             0.00
J       0227   (DBH)OBJECT/S/TASK/FILEACCESSR           15:20:00             0.00
J       0233   (DBH)OBJECT/S/TASK/FILEACCESSR           15:20:03             0.01
  NAME:  (DBH)OBJECT/S/TASK/IOTIME/EXTE

J       0244   (DBH)OBJECT/S/TASK/IOTIME/EXTE           15:20:26             0.01
```

Figure 8–4. LOGGER Example 4 Output

Example 5

In the following example, the REPORT command signifies the beginning of the report specifications. The SOURCE command selects the JOBSUMMARY file as the source of information for the report.

The INCLUDE command specifies that only MCS sessions are processed for the report. The SORT command specifies the output to be sorted by the USERCODE file data item.

The BREAK command specifies that totals be accumulated and printed for PROCESSTIME and IOTIME each time the value of the control break item (DATE) changes. The HEADING command specifies *MCS SESSION SUMMARY* to be used as the heading for the report.

The OUTPUT command selects the MIXNO, MCSNAME, LSN, STANAME, LOGONREASON, LOGOFFREASON, PROCESSTIME, IOTIME, and STARTTIME file data items to be printed. The END command signifies the end of the report-specification commands.

```
REPORT
SOURCE IS JOBSUMMARY
INCLUDE RECORD IF TYPE = "S"
SORT BY USERCODE
BREAK ON USERCODE TOTALING PROCESSTIME, IOTIME
HEADING "MCS SESSION SUMMARY"
OUTPUT ITEMS ARE MIXNO, MCSNAME, LSN, STANAME, LOGONREASON,-
  LOGOFFREASON, PROCESSTIME, IOTIME, STARTTIME
END
```

Figure 8–5 shows the resulting report.

```
                                    MCS SESSION SUMMARY
                                         04/04/90
MIXNO   MCSNAME   LSN   STANAME                 LOGONREASON       LOGOFFREASON        PROCESSTIME         IOTIME    STARTTIME
  USERCODE:  CHANG.
0566          1   634   SF79CD/"SERV1_CD"       UNSWITCHED        NORMAL LOG-OFF             0.00           0.00    15:28:48
- - - - TOTALS FOR CHANG. - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                                                             0.00           0.00
  USERCODE:  SASHA.
8281          2   568   TB154/CANDE/1.          NEW LOG ON        SPLIT SESSION              0.60           0.12    10:41:41
- - - - TOTALS FOR SASHA. - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                                                             0.60           0.12
```

Figure 8–5. LOGGER Example 5 Output

Example 6

In the following example, the REPORT command signifies the beginning of the report specifications. The SOURCE command selects the JOBSUMMARY file as the source of information for the report. The SORT command specifies the output to be sorted first in descending order by the USERCODE file data item, and then in ascending order by the NAME file data item.

The first BREAK command specifies that totals be accumulated and printed for PROCESSTIME and IOTIME each time the value of the control break item (USERCODE) changes. An average for the elapsed time is also printed. The second BREAK command specifies that totals be accumulated and printed for PROCESSTIME, IOTIME, MEMINTCODE, and MEMINTDATA each time the value of the control break item (NAME) changes.

The INCLUDE command specifies that only tasks are processed for the report. The OUTPUT command selects the MIXNO, PROCESSTIME, IOTIME, CARDSREAD, LINES, MEMINTCODE, MEMINTDATA, STARTTIME, and ELAPSEDTIME file data items to be printed. The HEADING command specifies *PROGRAM SUMMARY BY USERCODE* to be used as the heading for the report.

SUMMARY 1 in the REPORTS command causes totals to be printed for the control break item specified in the first BREAK command (USERCODE). SUMMARY 2 in the REPORTS command causes totals to be printed for the control break item specified in the second BREAK command (NAME). The totals generated by the REPORTS command are printed immediately after a corresponding dashed line is printed. The END command signifies the end of the report-specification commands.

```
REPORT
SOURCE FILE IS JOBSUMMARY
SORT BY USERCODE DESCENDING, NAME ASCENDING
BREAK ON USERCODE TOTALING PROCESSTIME, IOTIME AVERAGING-
  ELAPSEDTIME
BREAK ON NAME TOTALING PROCESSTIME, IOTIME, MEMINTCODE,-
  MEMINTDATA
INCLUDE RECORD IF TYPE  = "T"
OUTPUT ITEMS ARE MIXNO, PROCESSTIME, IOTIME, CARDSREAD, LINES,-
  MEMINTCODE, MEMINTDATA, STARTTIME, ELAPSEDTIME
HEADING IS "PROGRAM SUMMARY BY USERCODE"
REPORTS ARE DETAILED SUMMARY 1 SUMMARY 2
END
```

Figure 8–6 shows the resulting report.

```
                                                  PROGRAM SUMMARY BY USERCODE
                                                           08/07/90
MIXNO      PROCESSTIME          IOTIME    CARDSREAD       LINES   MEMINTCODE    MEMINTDATA    STARTTIME       ELAPSEDTIME
  USERCODE:  JOANNE.
  NAME:  OBJECT/ED ON SYSPK.
9561            0.63                2.34            0           0        68.58          87.72   12:39:29              26.69
- - - - TOTALS FOR OBJECT/ED ON SYSPK. - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                0.63                2.34                                 68.58          87.72
- - - - TOTALS FOR JOANNE. - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                0.63                2.34
- - - - AVERAGES FOR JOANNE. - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                                                                                  26.69
  USERCODE:  MARTEN.
  NAME:  SYSTEM/DUMPANALYZER ON SYSPK.
9939           11.49                2.15            0         711       366.67         672.58   13:57:51              23.16
- - - - TOTALS FOR SYSTEM/DUMPANALYZER ON SYSPK. - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
               11.49                2.15                                366.67         672.58
  NAME:  SYSTEM/FILEDATA ON SYSPK.
0342            0.65                0.54            0           0         7.07          14.74   14:39:16               0.76
0339            0.50                0.43            0           0         5.38          11.47   14:38:28               0.48
- - - - TOTALS FOR SYSTEM/FILEDATA ON SYSPK. - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                1.15                0.97                                 12.45          26.21
- - - - TOTALS FOR MARTEN. - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
               12.64                3.12
- - - - AVERAGES FOR MARTEN. - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                                                                                   8.13

                                                  PROGRAM SUMMARY BY USERCODE
                                                           08/07/90
                                                  SUMMARY REPORT BY USERCODE
                           TOTAL                 TOTAL     AVERAGE
USERCODE                   PROCESSTIME           IOTIME    ELAPSEDTIME   # OF RUNS
JOANNE.                           0.63             2.34          26.69   1
MARTEN.                         360.53            74.37           2.16   47

                                                  PROGRAM SUMMARY BY USERCODE
                                                           08/07/90
                                                     SUMMARY REPORT BY NAME
                                  TOTAL                       TOTAL     TOTAL          TOTAL
NAME                              PROCESSTIME                 IOTIME    MEMINTCODE     MEMINTDATA     # OF RUNS
  USERCODE: JOANNE.
OBJECT/ED ON SYSPK.                          0.63               2.34         68.58          87.72   1
  USERCODE: MARTEN.
SYSTEM/DUMPANALYZER ON SYSPK.               11.49               2.15        366.67         672.58   1
SYSTEM/FILEDATA ON SYSPK.                    1.15               0.97         12.45          26.21   2
```

Figure 8–6. LOGGER Example 6 Output

Example 7

In the following example, the REPORT command signifies the beginning of the report specifications. The SOURCE command selects the STATISTICS file as the source of information for the report.

The HEADING command specifies *SUMMARY STATISTICS* to be used as the heading for the report. The OUTPUT command selects the TIME, DISK, TAPE, READER, REMOTE, MISCFILES, TASKS, JOBS, HL, MAXTASKS, and MAXJOBS file data items to be printed.

The BREAK command specifies that totals be accumulated and printed for DISK, TAPE, READER, REMOTE, TASKS, JOBS, and HL each time the value of the control break item (DATE) changes. The END command signifies the end of the report-specification commands.

```
REPORT
SOURCE FILE IS STATISTICS
HEADING IS "SUMMARY STATISTICS"
OUTPUT ITEMS ARE TIME, DISK, TAPE, REMOTE, MISCFILES,-
  TASKS, JOBS, HL, MAXTASKS, MAXJOBS
BREAK ON DATE TOTALING DISK, TAPE, REMOTE, TASKS,-
  JOBS, HL
END
```

Figure 8–7 shows the resulting report.

SUMMARY STATISTICS
04/04/90

TIME	DISK	TAPE	REMOTE	MISCFILES	TASKS	JOBS	HL	MAXTASKS	MAXJOBS
DATE: 04/04/90									
15:00:00	0	0	0	0	9	6	0	2	3
15:15:00	0	90	10	2	440	110	0	6	4
15:30:00	0	26	22	3	153	30	0	10	5
15:45:00	0	11	17	1	24	9	0	3	3
TOTALS FOR 04/04/90	0	127	49		626	155	0		

Figure 8–7. LOGGER Example 7 Output

Example 8

In the following example, the REPORT command signifies the beginning of the report specifications. The SOURCE command selects the FILEIODATA file as the source of information for the report.

The OUTPUT command selects the MIXNO, INTNAME, EXTNAME, USE, KIND, FILEKIND, RETENTION, and IOTIME file data items to be printed. The HEADING command specifies *FILE USAGE REPORT* to be used as the heading for the report. The END command signifies the end of the report-specification commands.

```
REPORT
SOURCE FILE IS FILEIODATA
OUTPUT ITEMS ARE MIXNO, INTNAME, EXTNAME, USE, KIND, FILEKIND,-
  RETENTION, IOTIME
HEADING IS "FILE USAGE REPORT"
END
```

Figure 8–8 shows the resulting report.

FILE USAGE REPORT
10/09/90

MIXNO	INTNAME	EXTNAME	USE	KIND	FILEKIND	RETENTION	IOTIME
6605	DATABASE.	(ASTROX)JOB6605.	I/O	PACK	DATA	SCRATCH	0.00
6605	LINE.	LINE.	OUT	REMOTE	DATA		0.00
6609	LOG.	SUMLOG/8/100990/001419.	IN	PACK	DATA	PERMANENT	0.04
6608	LOG.	SUMLOG/8/100990/001419.	IN	PACK	DATA	PERMANENT	0.04
6609	LOG.	SUMLOG/8/100990/001419.	IN	PACK	DATA	PERMANENT	2.32
6608	LOG.	SUMLOG/8/100990/001419.	IN	PACK	DATA	PERMANENT	2.32
6608	FLOG.	SYSTEM/FAULTLOG.	I/O	PACK	DATA	PERMANENT	0.78
5577	INBOX.	(EMAJ)NEWS/NET/"PC SOFTWARE".	IN	PACK	DATA	PERMANENT	0.24
5577	INBOXINX.	(EMAJ)NEWS/INDEXF/NET/"PC_SOFT	IN	PACK	DATA	PERMANENT	0.03
6610	WFLCODE.	(JADROWS)WFLCODE.	OUT	PACK	CODE	SCRATCH	0.55
6610	LFYLE.	(JADROWS)WFL/NSP/MAKEFW.	IN	PACK	SYMBOL	PERMANENT	0.12
6612	FYLE.	(JADROWS)DECK/EMSCEXEC.	IN	PACK	SYMBOL	PERMANENT	0.00
6612	FYLE.	(JADROWS)PATCH/NSP/EMSCEXEC.	IN	PACK	DATA	PERMANENT	0.00
6612	FYLE.	(JADROWS)DECK/EMSCEXEC.	IN	PACK	SYMBOL	PERMANENT	0.00
6612	CARD.	(JADROWS)PATCHESFOR/EMSCEXEC.	IN	PACK	DATA	PERMANENT	0.00
6612	PATCH.	(JADROWS)MERGED/FWREL/EMSCEXEC	OUT	PACK	DATA	SCRATCH	0.01
6612	LINE.	(JADROWS)BD/0006611/0006612/00	OUT	PACK	BACKUP	PERMANENT	0.07
6612	TEMPFILE.	(JADROWS)TEMPFILE.	OUT	PACK	DATA	SCRATCH	0.02
6614	NEWWORKSOURCE.	(LAWRI)CANDE/TEXT6310.	OUT	PACK	SYMBOL	SCRATCH	0.00
6614	WORKSOURCE.	(LAWRI)DONP/NPC/INSERT/TEST.	IN	PACK	SYMBOL	PERMANENT	0.00
6614	NEWWORKSOURCE.	(LAWRI)CANDE/TEXT6310.	OUT	PACK	SYMBOL	SCRATCH	0.00
6614	MYOPTIONS.	(LAWRI)EDITOR/OPTIONS.	IN	PACK	DATA	PERMANENT	0.08
0898	INBOXINX.	(EMAJ)NEWS/INDEXF/NET/DAILY.	IN	PACK	DATA	PERMANENT	0.04
0898	"SUB_FILE".	(ROHBER)MAIL/NEWSVIEWED.	I/O	PACK	DATA	PERMANENT	0.02
9211	F05A.	CANDE/STARTUP.	IN	PACK	DATA	PERMANENT	0.05
6616	REMOTE.	REMOTE.	OUT	REMOTE	DATA		0.00
6821	"TEMPFILE'O".	(EMAJ)MAIL/OUTGOING/SFA17A/NEW	I/O	PACK	DATA	SCRATCH	0.00
6826	"TEMPFILE'O".	(EMAJ)MAIL/OUTGOING/SFA17A/NEW	I/O	PACK	DATA	SCRATCH	0.00
6821	"TEMPFILE'O".	(EMAJ)MAIL/OUTGOING/SFA17A/NEW	I/O	PACK	DATA	SCRATCH	0.02
6826	"TEMPFILE'O".	(EMAJ)MAIL/OUTGOING/SFA17A/NEW	I/O	PACK	DATA	SCRATCH	0.02
9210	F05B.	(SMTH)CANDE/RECV5670.	IN	PACK	CODE	PERMANENT	0.07
9210	F05B.	(SMTH)CANDE/RECV6420.	IN	PACK	CODE	PERMANENT	0.06
9211	F05A.	(SMTH)CANDE/STARTUP.	IN	PACK	DATA	PERMANENT	0.07
6821	USERMBOX.	(EMAJ)MAIL/NEWS.	I/O	PACK	DATA	PERMANENT	0.06
6826	USERMBOX.	(EMAJ)MAIL/NEWS.	I/O	PACK	DATA	PERMANENT	0.06
6821	"TEMPFILE'O".	(EMAJ)MAIL/OUTGOING/SFA17A/NEW	I/O	PACK	DATA	SCRATCH	0.02
6826	"TEMPFILE'O".	(EMAJ)MAIL/OUTGOING/SFA17A/NEW	I/O	PACK	DATA	SCRATCH	0.02
7595	TFOLD.	(EMAJ)NEWS/TEMPFOLDER/NEWS/INC	IN	PACK	DATA	PERMANENT	0.04
7595	TFOLDX.	(EMAJ)NEWS/INDEXF/TEMPFOLDER/N	IN	PACK	DATA	PERMANENT	0.06
7595	NEWS.	(EMAJ)MAIL/NEWS.	IN	PACK	DATA	PERMANENT	0.03
7595	TFOLDX.	(EMAJ)NEWS/INDEXF/TEMPFOLDER/N	IN	PACK	DATA	PERMANENT	0.02
7595	TFOLD.	(EMAJ)NEWS/TEMPFOLDER/NEWS/INC	IN	PACK	DATA	PERMANENT	0.03
6619	WFLCODE.	(ASTROX)WFLCODE.	OUT	PACK	CODE	SCRATCH	0.06
7595	NFOLD.	(EMAJ)NEWS/NET/POLITICS.	IN	PACK	DATA	PERMANENT	0.15
7595	NFOLDX.	(EMAJ)NEWS/INDEXF/NET/POLITICS	IN	PACK	DATA	PERMANENT	0.13
7595	TFOLDX.	(EMAJ)NEWS/INDEXF/TEMPFOLDER/N	IN	PACK	DATA	PERMANENT	0.03
7595	TFOLD.	(EMAJ)NEWS/TEMPFOLDER/NEWS/INC	IN	PACK	DATA	PERMANENT	0.04
7595	TFOLDX.	(EMAJ)NEWS/INDEXF/TEMPFOLDER/N	IN	PACK	DATA	PERMANENT	0.02
9211	F05A.	(SMTH)DO/ACL.	IN	PACK	DATA	PERMANENT	0.06
6622	NEWWORKSOURCE.	(SMTH)CANDE/TEXT7010.	OUT	PACK	SYMBOL	SCRATCH	0.00
6622	SYMBOL.	(KLMOR)SYMBOL/ACLSUPPORT.	IN	PACK	SYMBOL	PERMANENT	0.00
6622	NEWWORKSOURCE.	(SMTH)CANDE/TEXT7010.	OUT	PACK	SYMBOL	SCRATCH	0.00
6622	MYOPTIONS.	(SMTH)EDITOR/OPTIONS.	IN	PACK	DATA	PERMANENT	0.14
6613	NEWTAPE.	(JADROWS)SYMBOL/FWREL/EMSCEXEC	OUT	PACK	DATA	SCRATCH	13.12
6613	CODE.	(JADROWS)SYSTEM/FWREL/EMSCEXEC	OUT	PACK	DATA	SCRATCH	1.83

Figure 8–8. LOGGER Example 8 Output

Example 9

In the following example, the DRCONLY option indicates that only DRC data is to be collected, the REPORT command signifies the beginning of the report specifications, and the SOURCE command selects the DRCDATA files as the source of information for the report.

The SORT command selects the USERCODE, CHARGECODE, and ASCENDING options. The OUTPUT command selects the USERCODE, PACKNAME, DISKINUSE, MBYTEDAYS, USEDATE, USETIME, and CHARGECODE options as the data items to be printed.

The BREAK command specifies that totals be accumulated and printed for DISKINUSE and MBYTEDAYS each time the value of the CHARGECODE and USERCODE control breaks change. The totals generated by the REPORTS command are printed immediately after a corresponding line of hyphens (-) is printed.

The END command indicates the end of the report-specification commands.

```
OPTION DRCONLY
REPORT
SOURCE IS DRCDATA
SORT BY USERCODE, CHARGECODE, ASCENDING
OUTPUT ITEMS ARE USERCODE, PACKNAME, DISKINUSE, MBYTEDAYS, USEDATE,
USETIME, CHARGECODE
BREAK ON USERCODE, CHARGECODE TOTALING DISKINUSE, MBYTEDAYS
REPORTS ARE DETAILED SUMMARY 1 SUMMARY 2
END
```

Figure 8–9 shows the resulting report.

```
                                                                   06/06/91
USERCODE              PACKNAME                       DISKINUSE     MBYTEDAYS  USEDATE    USETIME    CHARGECODE

  USERCODE:  PDQ

  CHARGECODE:  1130

PDQ                   NEWMCPS                             0.00         91.72  05/03/91   09:32:00   1130
PDQ                   NEWPACK                             0.00         88.42  05/03/91   09:32:00   1130
PDQ                   SCRATCH249                          0.00          0.00  05/03/91   09:32:00   1130
PDQ                   SCRATCH67                           0.00       3261.77  05/03/91   09:32:00   1130
PDQ                   DISK                                0.00          0.69  05/03/91   09:32:00   1130
PDQ                   TESTMCPS                            0.00          0.00  05/03/91   09:32:00   1130
PDQ                   SYS39                               0.00          0.00  05/03/91   09:32:00   1130
PDQ                   OLDMCPS                            91.07       6621.74  05/03/91   09:32:00   1130
PDQ                   MCPS                               30.85       3681.94  05/03/91   09:32:00   1130
PDQ                   PACK                               32.20       2818.45  05/03/91   09:32:00   1130
PDQ                   MCPMAST                            50.33       2822.64  05/03/91   09:32:00   1130
- - - - TOTALS FOR 1130          - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                        204.45      19387.37
- - - - TOTALS FOR PDQ           - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                        204.45      19387.37

  USERCODE:  VIOLAN

  CHARGECODE:  1278

VIOLAN                PACK                                0.00          0.21  03/27/91   13:15:02   1278
VIOLAN                DISK                                0.00          0.00  03/27/91   13:15:02   1278
- - - - TOTALS FOR 1278          - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                          0.00          0.21
- - - - TOTALS FOR VIOLAN        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                          0.00          0.21

  USERCODE:  MERVYN

  CHARGECODE:  2046

MERVYN                MCPS                                0.00        633.59  04/08/91   20:01:25   2046
MERVYN                DISK                                0.00          0.00  04/08/91   20:01:25   2046
MERVYN                PACK                                0.00          0.61  04/08/91   20:01:25   2046
MERVYN                MCPMAST                             2.35        163.45  04/08/91   20:01:25   2046
- - - - TOTALS FOR 2046          - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                          2.35        797.65
- - - - TOTALS FOR MERVYN        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                          2.35        797.65

  USERCODE:  SMYTHE

  CHARGECODE:  1263

SMYTHE                NEWPACK                             0.00          0.02  05/03/91   06:28:05   1263
SMYTHE                OLDPACK                             0.02          0.87  05/03/91   06:28:05   1263
SMYTHE                OLDOPS                              0.00          0.10  05/03/91   06:28:05   1263
SMYTHE                FIRE                                0.38         22.51  05/03/91   06:28:05   1263
SMYTHE                DISK                                0.00          0.00  05/03/91   06:28:05   1263
SMYTHE                OPS                                 0.00          0.16  05/03/91   06:28:05   1263
```

Figure 8–9. LOGGER Example 9 Output

File Data Items

Tables 8–1 through 8–3 describe all the file data items present in the JOBSUMMARY, STATISTICS, and FILEIODATA files. Table 8–4 describes the file data items present in the DRCDATA files.

Table 8–1. JOBSUMMARY File Data Items

Item	Type	Description
ACCESSCODE	string	Accesscode (without password) of the entry
AVGCORECODE	integer	Average core usage for job or task (code)
AVGCOREDATA	integer	Average core usage for job or task (data)
CARDSPUNCHED	integer	Number of cards punched
CARDSREAD	integer	Number of cards read
CHARGECODE	string	Chargecode for this entry (limited to 10 characters)
CHARGES	charge	Billing charges for this entry
CODEFILE	string	Code file name for job or task
DATE	string	Date of entry in form mm/dd/yy
DESTMCS	integer	Destination MCS number for job or task
DESTUNIT	string	Destination unit for job or task
ELAPSEDTIME	real	Elapsed time in minutes
INITPBITS	integer	Number of initial presence bit operations (job or task only)
INITPBITTIME	real	Process time for initial presence bit operations
IOTIME	real	I/O time in seconds
JOBENTRYTIME	string	Time job entered system
JOBNO	string	Job number of entry. The JOBNO data item is always four digits long, and is padded with leading zeros if necessary.
JOBQUEUEDTIME	real	Time job in queue
LINES	integer	Number of lines printed
LOGOFFREASON	string	Reason for log-off (session only)
LOGONREASON	string	Reason for log-on such as hello or split (session only)

continued

Table 8–1. JOBSUMMARY File Data Items (cont.)

Item	Type	Description
LSN	string	Station number (session only). The LSN data item is always three digits long, and is padded with leading zeros if necessary.
MCSNAME	string	MCS number (session only)
MEMINTCODE	real	Memory integral for job or task (code)
MEMINTDATA	real	Memory integral for job or task (data)
MIXNO	string	Mix number of entry. The MIXNO data item is always four digits long, and is padded with leading zeros if necessary.
NAME	string	Job or task name. *-MCS SESSION-* for MCS session
ORGMCS	integer	Originating MCS number for job or task
ORGUNIT	string	Originating unit number for job or task
OTHERPBITS	integer	Number of other presence-bit operations (job or task only)
OTHERPBITTIME	real	Process time for other presence bit operations
PRIORITY	integer	Priority
PROCESSTIME	real	Processor time in seconds
QUEUE	string	Queue number. The QUEUE data item is always three digits long, and is padded with leading zeros if necessary.
STANAME	string	Station name (session only)
STARTTIME	string	Time of program initiation in the form hh:mm:ss
STOPTIME	string	Time of program termination in the form hh:mm:ss
TERMCOND	string	Termination condition for job or task
TYPE	string	J for job, T for task, and S for MCS session
USERCODE	string	Usercode

Notes:

- *String items cannot be totaled or averaged; however, all other items can.*
- *LOGGER computes the ELAPSEDTIME value for RESTART MCS sessions from the LOGON and LOGOFF log records instead of using the ELAPSEDTIME value in the LOGOFF record.*
- *The CHARGECODE as reported by LOGGER is limited to 10 characters. The first 10 characters are used. CHARGECODEs longer than 10 characters are truncated in the printed report.*
- *LOGGER computes the USERCODE value from the Major Type 1, Minor Type 2 (EOJ) or 4 (EOT) log entry for the process.*

Table 8–2. STATISTICS File Data Items

Item	Type	Description
DATE	string	Date of log entry in the form mm/dd/yy
DISK	integer	Number or disk file opens during interval
HL	integer	Number of halt/loads
JOBS	integer	Number of jobs initiated
MAXJOBS	integer	Maximum number of jobs running at one time
MAXTASKS	integer	Maximum number of tasks running at one time
MISCFILES	integer	Number of file opens not included in the other file open counts
PACK	integer	Number of pack file opens
PUNCH	integer	Number of punch file opens
READER	integer	Number of card reader file opens
REMOTE	integer	Number of remote file opens
SESSIONS	integer	Number of MCS sessions initiated
TAPE	integer	Number of tape file opens
TASKS	integer	Number of tasks initiated
TIME	string	Time of day in the form hh:mm:ss at the beginning of the 15-minute interval

Notes:

- *String items cannot be totaled or averaged; however, all other items can.*
- *The MAXJOBS and MAXTASKS items counts are obtained by counting beginning of tasks (BOTs) and beginning of jobs (BOJs) and subtracting end of tasks (EOTs) and end of jobs (EOJs). Therefore, missing log entries cause errors in these data.*

Table 8–3. FILEIODATA File Data Items

Item	Type	Description
ACCESSCODE	string	Accesscode (without password) for this entry
AREASIZE	integer	Area size
ASSOCIATION	string	ASSOCIATION of file close
BLOCKSIZE	integer	Block size in words (value is rounded up if units equal characters)
BUFFERSIZE	integer	BUFFERSIZE file attribute
BUFFSOURCE	string	Buffer size source. The possible values are • BLOCKSIZE, indicating that the buffer size of the file was based on the BLOCKSIZE file attribute because the BUFFERSIZE file attribute is not applicable to this file. • BUFFERSIZE, indicating that the buffer size of the file was specified through an assignment to the BUFFERSIZE file attribute. • BUFFERGOAL, indicating that the buffer size of the file was based on the BUFFERGOAL memory management parameter of the SF (Set Factors) system command.
CCSVERSION	string	CCSVERSION file attribute (disk and CD files)
CHARGECODE	string	Chargecode for this entry (limited to 10 characters)
CHARGES	charge	Billing charges for this entry
CLOSETYPE	string	Type of close
CREATIONDATE	string	Creation date in yyddd
DENSITY	string	DENSITY file attribute

continued

Table 8–3. FILEIODATA File Data Items (cont.)

Item	Type	Description
DISPOSITION	string	DISPOSITION of file close
DUMMYFILE	string	DUMMYFILE if attribute is true, otherwise blank
EXTMODE	string	EXTMODE file attribute
EXTNAME	string	External name of the file
FAMILYNAME	string	FAMILYNAME file attribute (disk files)
FILEKIND	string	FILEKIND file attribute
FILESTRUCTURE	string	FILESTRUCTURE file attribute (disk files)
FRAMESIZE	integer	FRAMESIZE file attribute
INTMODE	string	INTMODE file attribute
IOTIME	real	I/O time used in seconds
INTNAME	string	Internal name of the file
JOBNO	string	Job number. The JOBNO data item is always four digits long, and is padded with leading zeros if necessary.
KIND	string	KIND file attribute
MAXRECSIZE	integer	Maximum record size in words (value is rounded up if units equal characters).
MIXNO	string	Mix number. The MIXNO data item is always four digits long, and is padded with leading zeros if necessary.
REELNO	string	Reel number for tape files. The REELNO data item is always six digits long, and is padded with leading zeros if necessary.
RETENTION	string	SCRATCH, PERMANENT, or none
SAVEFACTOR	string	SAVEFACTOR
SERIALNO	string	Serial number
TIME	string	Time of file close as hh:mm:ss
UNITNO	string	Unit number. The UNITNO data item is always five digits long, and is padded with leading zeros if necessary.

continued

Table 8–3. FILEIODATA File Data Items (cont.)

Item	Type	Description
UNITS	string	Units ***Note:*** *The UNITS item is scheduled for deimplementation in a future release. The FRAMESIZE and INTMODE items can be used instead to provide the same information.*
USE	string	IN, OUT, or I/O
USERCODE	string	Usercode

Notes:

- *String items cannot be totaled or averaged; however, all other items can.*
- *For information about the file attributes corresponding to FILEIODATA items, refer to the* A Series File Attributes Programming Reference Manual.

Table 8–4. DRCDATA File Data Items

Item	Type	Description
ACCESSCODE	string	The first of the values defined for the ACCESSCODELIST usercode attribute in the USERDATAFILE
CHARGECODE	string	Chargecode for this entry
CHARGES	charge	Billing charges for this entry
DISKINUSE	real	Disk space currently in use by the specified user on the specified pack. This value is reported in megabytes.

continued

Table 8–4. DRCDATA File Data Items (cont.)

Item	Type	Description
MBYTEDAYS	real	The disk integral value for the specified user on the specified pack. This value is reported in megabyte-days, which is the number of megabytes used multiplied by the number of days the pack is used. For example, if a user stores one megabyte on a pack for one day, the MBYTEDAYS value is one. The value is recalculated by the DRC system every time the amount of space in use by the specified user is changed.
PACKNAME	string	The family name of the disk for this entry
USEDATE	real	The date that the system last updated the DRC disk usage for this entry
USERCODE	string	The usercode for this DRC entry
USETIME	real	The time that the system last updated the DRC disk usage for this entry

Long-Term Report Generation

LOGGER is able to generate long-term reports. These reports can cover more than one day or more than one log file. The following types of long-term reports can be generated:

- Reports based on data files or log files accumulated over a certain period.
- Reports based on year-to-date totals.

Extended Time Period Reports

By default, LOGGER bases its report on data found in the current log file (SYSTEM/SUMLOG). Refer to Section 12, "SUMLOG," for information about SYSTEM/SUMLOG. The SUMLOG is found on the current log family, as specified by use of the DL (Disk Location) system command. Refer to the *A Series System Commands Operations Reference Manual* for information about the DL command. However, the USE command can be supplied to the program to specify that its source information is elsewhere. Refer to "USE Command" for more information.

Only one USE command can appear in an input deck. If none appears, the current SYSTEM/SUMLOG is used. The USE SYSTEM/SUMLOG command explicitly specifies that the current SYSTEM/SUMLOG is to be used.

The following commands indicate that system logs released through a TL (Transfer Log) system command are to be used. Refer to the *A Series System Commands Operations Reference Manual* for information about the TL command.

Command	Description
USE SUMLOG CURRENT	SUMLOG files with the current date in their titles are to be used.
USE SUMLOG <mmddyy>	SUMLOG files with the given date in their titles are to be used.
USE SUMLOG <mmddyy mmddyy>	SUMLOG files whose titles contain date between the two given dates, inclusive, are used.

The following commands indicate that existing JOBSUMMARY, STATISTICS, and FILEIODATA files are to be used to generate reports:

Command	Description
USE CURRENT	Files with the current date in their titles are used.
USE <mmddyy>	Files with the given date in their titles are used.
USE <mmddyy mmddyy>	Files whose titles contain dates between the two given dates, inclusive, are used.

Year-to-Date Totals Reports

LOGGER can create, update, and generate reports from a year-to-date totals file. These options are invoked by specifications in the OPTION command. Refer to "OPTION Command" for more information.

The year-to-date totals file has a structure that is implicitly defined by the report specifications used to create it. Each record corresponds to a change in a control break item and contains those totals or averages that were generated at that point.

For example, if the year-to-date file was created by a report that specified USERCODE and CHARGECODE as control break items and totaled processor time, I/O time, elapsed time, and charges, each record in the year-to-date totals file would correspond to one combination of usercode and chargecode and would contain the previously mentioned totals.

Because the structure of the report determines the structure of the file, any report can be used to create the file initially. However, after this initial file creation, all updates must be performed by either the same report or a report with the same control break items and totals. If the program detects that the report being used to update the year-to-date file has a different structure than that file, it does not perform the update and issues an error message.

The internal name of the year-to-date file is YTDFILE. The YTDFILE file can be file-equated. When LOGGER is run with the UPDATE option enabled, it first checks to see if the file already exists. Refer to "OPTION Command" for information about the UPDATE option. If the file exists, it is updated; otherwise, a new file is created.

Only the totals for the last specified control break item are kept in the file. For example, if the report used to create the file had the following BREAK commands, then only totals for ELAPSEDTIME would be kept in the year-to-date totals because ELAPSEDTIME is the only item specified at the innermost level:

```
BREAK ON USERCODE TOTALING PROCESSTIME, IOTIME
BREAK ON CHARGECODE TOTALING ELAPSEDTIME
```

Year-to-Date File Updates

When the year-to-date file is updated, no existing records are modified, and new records are added at the end of the file. At the time that the year-to-date report is generated, the program locates all records concerning the same combination of control break items and totals all of the appropriate totaled items at that time. Each record in the file contains the date on which that record was added to the file, which makes it possible to determine what changes were made on each update to the year-to-date totals. The date stored in the YTDFILE record is in Julian form (yyddd) to enable the year-to-date file to run longer than one calendar year.

Year-to-Date Totals File Format

Figures 8–10 through 8–12 illustrate the file format of record numbers 1 through 3.

Record number 1 contains 60 words of control break item descriptions, as shown in Figure 8–10.

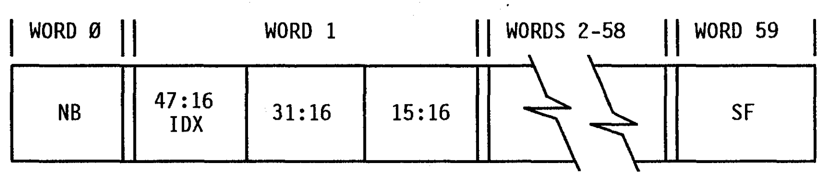

Figure 8–10. Year-to-Date Record 1 Format

Word	Information
0	NB (Number of control break items)
1 through 58	IDX (IDTABLE index)
	LEN (Length of item in characters)
	STRT (Starting character position in the file)
59	SF (Source file)
	0 (JOBSUMMARY file)
	1 (STATISTICS file)
	2 (FILEIODATA file)

Record number 2 contains the totaled item descriptions shown in Figure 8–11.

WORD Ø	WORD 1				WORDS 2-58	WORD 59
NB	47:16 IDX	31:16 LEN	15:16 STRT	7:8 TAF		SF

Figure 8–11. Year-to-Date Record 2 Format

Word	Information
0	NT (Number of totaled items)
1 through 58	IDX (IDTABLE index)
	LEN (Not used)
	STRT (Start position or word index)
	TAF (0 if item is totaled, 1 if averaged)

Record number 3 contains the item descriptions shown in Figure 8–12.

BI(1)	BI(2)	BI(3)	...BI(NB)	TI(1)	TI(2)	TI(3)	...TI(NT)	C	D

Figure 8–12. Year-to-Date Record 3 Format

Word	Information
n	BI (Control break items—character strings)
	TI (Totaled control break item binary values)
	C (Count number of entries from which totals were obtained)
	D (Date)

Program Operation Characteristics

The following variations in the operating characteristics can affect LOGGER output.

REPORT Commands and LOGREPORTS File

A REPORT command indicates to LOGGER that it is to read a set of report specifications. Refer to "REPORT Command" for more information.

If an integer is present in the REPORT command, the report is located in the LOGREPORTS file, and the integer is the identifying number of the report.

Report-specification commands are read until an END command is found, at which time reading of input records is resumed. If no integer is present in the REPORT command, the report-specification commands are assumed to be on input records immediately following the REPORT command. Commands are scanned up to column 72. Columns 73 through the end can be used for sequence numbers or comments.

A LOGREPORTS file can contain as many report specifications as are needed. Each command has an identifying number associated with it.

In the following example, the LOGREPORTS file contains only two reports. The records containing a number sign (#) as the first character are the report identification records, and the number following the number sign is the identifying number of the report that follows it.

Record Number	Contents
1	#1
2	SOURCE FILE IS JOBSUMMARY
3	OUTPUT ITEMS ARE NAME, USERCODE, STARTTIME
4	PAGE SIZE IS 57
5	HEADING IS "SYSTEM USAGE REPORT"
6	END
7	#2
8	SOURCE FILE IS STATISTICS
9	OUTPUT ITEMS ARE TIME, TASKS, JOBS, DISK, PACK,
10	BREAK ON DATE
11	SORT BY DATE, TIME
12	END

LOGGER does not provide facilities for updating the LOGREPORTS file; however, it can be updated using CANDE. The file can also be maintained as a card deck and put on disk by using the WFL *DECK* statement. Refer to the *A Series Work Flow Language (WFL) Programming Reference Manual* for information about the DECK statement.

Calculation of Charges

The JOBSUMMARY, FILEIODATA, and DRCDATA files include a CHARGES data item, which returns the dollar amount charged for various user activities on the system. LOGGER invokes a procedure in the BILLINGSUPPORT library to calculate the value of the CHARGES data item. LOGGER also passes the ACCESSCODE data item to BILLINGSUPPORT for use in its CHARGES calculations. BILLINGSUPPORT is a support library that is designed to be rewritten by the installation to meet its special billing needs. For further information about BILLINGSUPPORT, refer to the BILLINGSUPPORT symbolic and to the *A Series System Administration Guide*.

Corrections

If the log contains erroneous data for an entry, the CORRECTION command allows the correct value to be given on the report. Refer to "CORRECTION Command" in this section for more information.

If the results of the run are used to update the year-to-date totals file, the corrected value is used. If the report is being generated from log files instead of from an existing JOBSUMMARY file, the resulting JOBSUMMARY file contains the corrected value. If the report is generated from an existing JOBSUMMARY file, the report shows the corrected value; however, the file is not changed.

When correcting the CHARGES field, the value is an integer value in cents with no dollar sign ($). Refer to "Calculation of Charges" earlier in this section for information about the CHARGES field.

Files and File Equation

The following text describes how LOGGER files can be file-equated:

CARD LINE	The input card reader file or line printer file, or both, can be file-equated as remote files to run from a CANDE terminal. The CARD file does not need to exist if input cards are not supplied.
LINE	The following syntax should be used for installations with CATALOGING set to TRUE that require the LINE file to be file-equated to BACKUP TAPE: `FILE LINE(KIND=PRINTER,BACKUPKIND=TAPE,LABELTYPE=STANDARD)` The LABELTYPE=STANDARD specification allows a volumed printer backup tape to be created.
LOGREPORTS	The file from which report specifications are read. No restrictions exist on file-equation.
JOBSUMMARY STATISTICS DRCDATA FILEIODATA	The titles of these files cannot be changed through file-equation because the program modifies the file titles in order to put the date in them.
YTDFILE	The year-to-date totals file. No restrictions on file-equation.

$NODUMP Compile-Time Option

SYMBOL/LOGGER contains an ONANYFAULT statement in the procedure that reads the log file to prevent the program from being terminated in the event of bad data in the log entry. If the program encounters a fault, it takes a program dump in case the fault was caused by a program error. The $NODUMP compile-time option, which can be used to recompile SYMBOL/LOGGER, suppresses program dumps. This option can be used when data in the log that causes the program to receive a fault is known to be present and program dumps are not wanted.

Program Information

The following programming information is intended for those who wish to modify the program for use at a particular installation. However, this information is not necessary for understanding the use of the program.

Overall Organization

LOGGER performs the following steps:

1. LOGGER reads input-specification and report-specification commands, checks syntax of input, and builds arrays for use by later steps. The main procedures involved are PROCESSINPUTCARDS and PROCESSREPORTSPECIFICATIONS.
2. LOGGER reads the log file and creates the JOBSUMMARY, STATISTICS, DRCDATA and, optionally, any required FILEIODATA files. This step is omitted if data is obtained from an existing JOBSUMMARY, STATISTICS, DRCDATA, or FILEIODATA file. The main procedures involved are WRITEITEM, WRITEJOBSUMMARY, and WRITEIODATAFILE.
3. LOGGER reads the JOBSUMMARY, STATISTICS, DRCDATA, or FILEIODATA file. LOGGER extracts appropriate items, and sorts by appropriate items as specified in the report specifications. LOGGER then prints a report. The main procedure involved is EDITOR.
4. LOGGER writes a summary report if SUMMARY is specified in the REPORTS command. Refer to "REPORTS Command" for more information. The procedure involved is SUMMARYREPORT.
5. LOGGER initializes a year-to-date totals file, if this file is requested. The procedure involved is INITIALIZEYTDFILE.
6. LOGGER generates a report from the year-to-date totals file. The procedure involved is YTDREPORT. This procedure is executed if the OPTION YEAR command is specified. Refer to "OPTION Command" in this section for more information.

Structure of Program Files

The program files contain the results of processing of the log entries and are passed to the EDITOR report-generation procedure. The JOBSUMMARY file has one record for each job, task, and MCS session found in the log. The STATISTICS file has one record for each 15 minutes of data in the log. The FILEIODATA file has one record for each file-close record found in the log. All data in these files, including numeric items, is in EBCDIC form.

The following four value arrays are associated with each file:

- IDTABLE
- NAMEINFOTABLE

- SHORTNAMES table
- NAMETABLE

The first three value arrays are parallel tables (the nth entry in one corresponds to the nth entry in the others), and one entry exists in the table for each item in the files.

The IDTABLE specifies where the item occurs in each record of the appropriate file, the length of the item, and its type of data. For example, taking the first item from the JOBIDTABLE (which is the IDTABLE for the JOBSUMMARY file), the listing shows a declaration of PLF(001,04,0). The first number is the starting character position in each record of the file where this item is found. Therefore, this item starts in character position 1. The second number is the length in characters; the item is four characters long. The third number is the data type. Zero specifies alphanumeric, one specifies a real number; two specifies an integer, and three specifies a field containing a floating dollar sign ($) in the CHARGES field. Refer to “Calculation of Charges” earlier in this section for additional information about the CHARGES field. All items are actually stored in EBCDIC characters, but the data-type field is used to decide what to do with the item when it must be totaled or averaged.

The name of the totaled or averaged item must be found in an appropriate NAMEINFOTABLE (in this case, the JOBNAMEINFOTABLE). From the listing, JOBNAMEINFOTABLE has a value of PL(000,05), which means that the name of the item is found in the NAMETABLE starting at character 0 for five characters.

The SHORTNAMES table, which contains the first six characters of each name, is used by the procedure scanning the report specifications. This procedure performs a MASKSEARCH of the the report specifications. SHORTNAMES table then goes to the parallel NAMEINFOTABLE, takes that information to locate the full name in the NAMETABLE, and takes the corresponding entry from the IDTABLE to determine the location of the data.

Four array-reference variables are set to the appropriate arrays when the SOURCE command is processed. Refer to “SOURCE Command” for more information. When the EDITOR procedure is called to print the report, the procedure is passed one of the three files as a parameter. Because EDITOR uses the array-reference variables, it need not be aware of which file it is processing. The process is identical for each file.

Tables Used by the EDITOR Procedure

The procedure PROCESSREPORTSPECIFICATIONS builds several tables from the report-specification commands. These tables are then used by the EDITOR procedure. A brief description of each table is presented here, followed by Figures 8–13 through 8–16 showing the fields in each table.

BREAKINFO	Contains information from the BREAK commands. Refer to "BREAK Command" for more information.
EDITORINFO	Contains an entry for each output item.
INCLCHECK	

continued

continued

EXCLCHECK	Contains information from the INCLUDE and EXCLUDE commands. Refer to "INCLUDE Command" and "EXCLUDE Command" for more information.
TAITEMS	Contains information on the items specified for totaling and averaging.

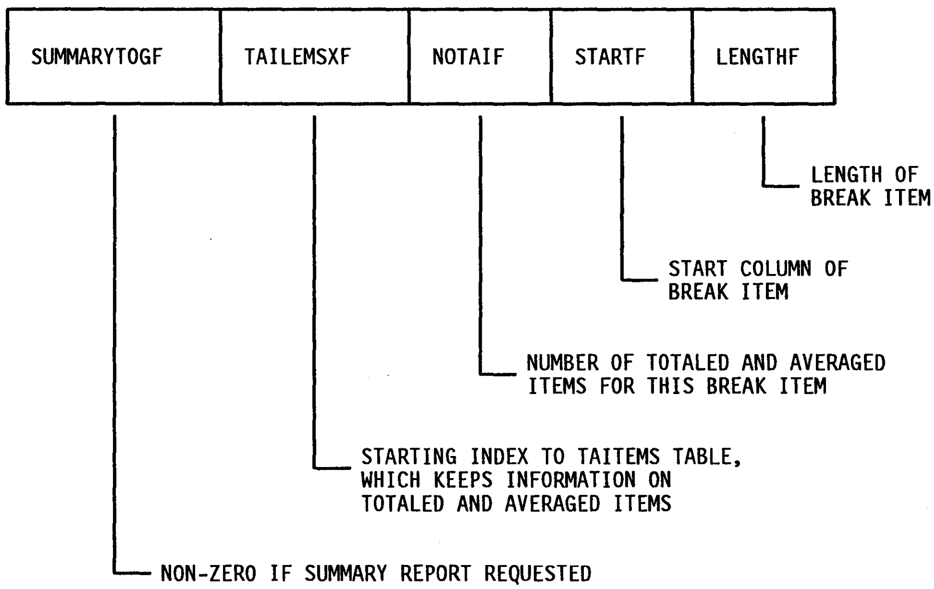

Figure 8–13. BREAKINFO Table

Note: *Items STARTF and LENGTHF are duplicates of information kept in IDTABLE and are used to identify the item. Refer to "Structure of Program Files" earlier in this section for additional information about IDTABLE.*

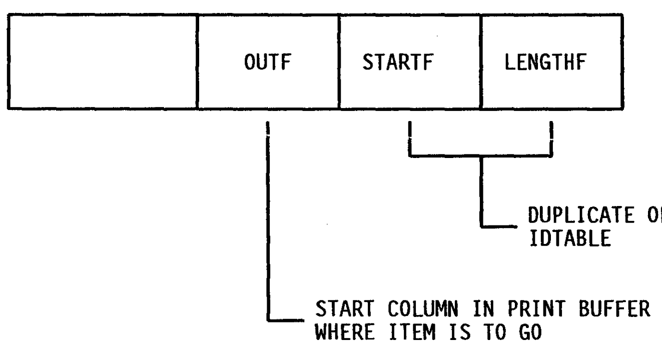

Figure 8–14. EDITORINFO Table

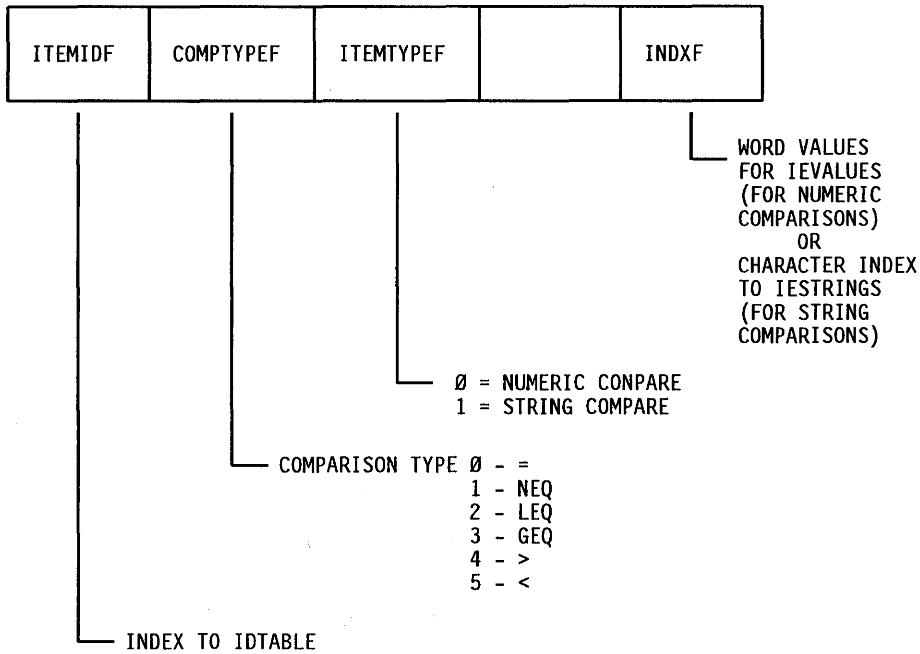

Figure 8–15. INCLCHECK and EXCLCHECK Tables

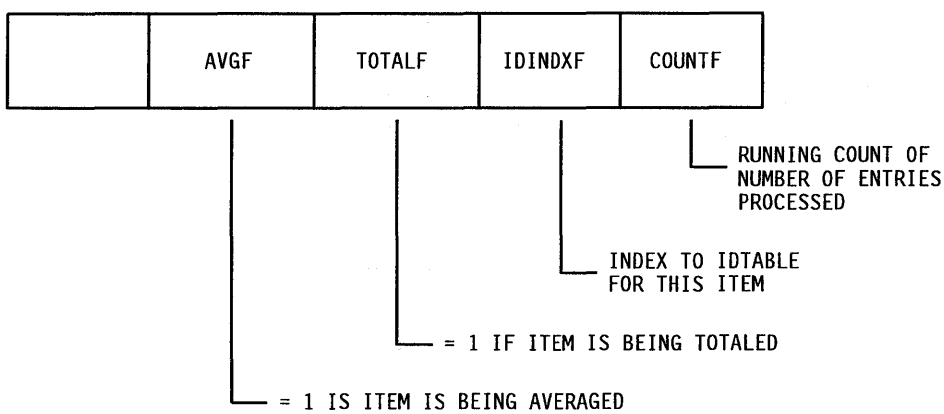

Figure 8–16. TAITEMS Table

Note: *A parallel table, TA, is used to hold the current total value for the corresponding TAITEMS entry.*

Files Used by the Program

The following types of files are utilized by LOGGER.

Table 8–5. Files Used by LOGGER

File Type	Name	Description
Global	LOG	The input log file. By default, the title is SYSTEM/SUMLOG; however, it can be changed to another file if a USE command is specified. Refer to "USE Command" for more information.
	DRCAUDIT	This file is created when the RESETDRC option is specified. DRCAUDIT contains all the records generated by LOGGER. The records are in the format used by the BILLINGSUPPORT library. The records contain the values accumulated before the MBYTEDAYS field was set to zero (0). If any billing information is lost, a program can read the DRCAUDIT file and then call the BILLINGSUPPORT library to calculate the charges.
	JOBSUMMARY STATISTICS DRCDATA FILEIODATA	Refer to "LOGGER Operation" earlier in this section for information about the JOBSUMMARY, STATISTICS, DRCDATA, and FILEIODATA files.
	OUTFILE	Used by the procedure EDITOR when a SORT command is included in the report specifications. Refer to "SORT Command" for more information. EDITOR performs a sort whose output is OUTFILE. This output is then read in to print the report.
	SUMMARY	If a REPORTS command specifies a summary report, the procedure BREAKCHECK saves information in this file each time a break occurs. Refer to "REPORTS Command" for more information. The file is read in later by the procedure SUMMARYREPORT.
	YTDFILE	The year-to-date totals file
	PRNT	The printer file. Its INTNAME is LINE.
Local To PROCESSIN-PUTCARDS	CARD	The card reader file

continued

Table 8–5. Files Used by LOGGER (cont.)

File Type	Name	Description
	LOGREPORTS	The file from which the report specifications are read
Local To INITIALIZEYTDFILE	NEWYTDFILE	When updating an existing YTDFILE file, all records prior to the current day are copied into NEWYTDFILE, and the old YTDFILE is then removed. Refer to to YTDFILE in this table for more information. The title of NEWYTDFILE is changed to that of the old YTDFILE so that the updated file has the same name.

Section 9
Peripheral Test Driver (PTD)

The peripheral test driver (PTD) is an interpreter program that is part of the master control program (MCP). It interprets s-ops that are found in test-case semicompiled code (s-code) files that were created to test peripheral equipment on a system. PTD is available on all A Series systems.

A test case is written with either a physical device or a BADDISK file as its target. The connection between the test case and test object is established by way of

- A programmatic test case data structure called a DEVICESTRUCTURE
- The OPEN s-op
- A device mnemonic or disk file name entered by the user

The test case s-code files and descriptions of the peripheral tests that can be performed are found on the PTDTESTS TAPE.

Test cases are organized into numbered "sections" that perform certain operations. Those sections that fall into logical categories are called blocks. For example, block 100 can consist of sections 101, 102, and 103, where those particular sections test the TESTID, the READBUFFER, and the WRITEBUFFER operations respectively. Other blocks can consist of sections designed to perform complex sequences of data transfer that drive a peripheral or data link processor (DLP) at a certain level of tolerance. Sections numbered from 3000 above are special sections that are not performed unless they are explicitly invoked by the user.

Refer to the RUN and REPEAT directives in this manual for information on explicitly invoking certain sections. Section 4000 lists what the other sections do and is particularly useful. Refer to the PTDTESTS TAPE for each test case's organization.

The following steps are necessary to execute a peripheral test:

1. Execute PTD.
2. Select the directives necessary for the test.
3. Select the directives unique to IOP PTD.
4. Reserve the peripheral to be tested, if necessary.

Descriptions of these procedures follow.

PTD Statement

The PTD statement is recognized by the Work Flow Language (WFL) compiler, and starts SYSTEM/MAINTENANCE as an independent runner.

To initiate a test on a particular type of peripheral, DLP, Control (CTL), or data communications data link processor (DCDLP) the statement must also contain file-equate syntax. That syntax should file-equate the PTD internal s-code file to the appropriate test case code file.

The A Series systems allow PTD to operate from either an operator display terminal (ODT) or a data comm terminal. In either case, all PTD input and displays are done through a logical file titled PTDSPO. When using the ODT, all input to PTD must have a triangular GS (group-separator) in column 1. This character directs the entry to PTD instead of to the CONTROLLER. To direct the operator dialog to a data comm station, the user should change the KIND of PTDSPO to REMOTE and the TITLE of PTDSPO to the station name; GS is not used with this medium.

For users primarily debugging test case code or PTD itself, PTD can be executed with a VALUE clause that sets certain of its options immediately at run time. The relevant bits are

Bit	Option Set	Function
[2:1]	TRACE	Test-case s-ops in execution
[1:1]	DUMP	Test-case virtual memory dump
[0:1]	PGMDUMP	Program dump of PTD

Hence, executing PTD with VALUE = 2 causes the DUMP option to be set, VALUE = 3 sets both DUMP and PGMDUMP, and so on.

General Execution Syntax

In general, PTD is executed by entering a statement in the following form, where <s-code file name> is the name of the test case code file found on the PTDTESTS TAPE:

```
PTD; FILE SCODE(TITLE = <s-code file name>)
```

Other items are entered depending on the user's choice of dialog device and the user's rare decision to add the VALUE clause. Actual s-code file names are used in the following examples. Refer to the PTDTESTS Tape for a description of the release tape, file naming conventions, and individual test case documentation.

ODT Execution Examples

In the following examples, PTD is executed at an ODT, and all subsequent operator dialog occurs at that ODT:

- To execute PTD in conjunction with the test case s-code file named PTD/MAINT/CP, enter the following command:

```
PTD; FILE SCODE(TITLE = PTD/MAINT/CP)
```

- To execute PTD in conjunction with the test case s-code file named PTD/CONF/CR found on the family disk named DMS and to request a virtual memory dump prior to end-of-job (EOJ), enter the following command:

```
PTD; FILE SCODE(TITLE = PTD/CONF/CR ON DMS); VALUE = 2
```

- To execute PTD in conjunction with the PTD/MAINT/PE test case s-code file and to request a virtual memory dump prior to EOJ, enter the following command:

```
BEGIN JOB; PTD; FILE SCODE(TITLE = PTD/MAINT/PE); VALUE = 2
```

ODT Initiation, Remote (Data Comm) Dialog Device Examples

In the following examples PTD is executed at an ODT but all further dialog occurs at the specified data comm terminal:

- To execute PTD in conjunction with the test case s-code file named PTD/MAINT/HTS found on the family disk named DISK from a remote terminal with a station number of TD450365, enter

```
PTD; FILE SCODE(TITLE = PTD/MAINT/HTS ON DISK);
     FILE PTDSPO(KIND = REMOTE, TITLE = TD450365)
```

- To execute PTD in conjunction with the test case s-code file named PTD/CONF/PSS from a remote terminal station number 55, and to request a virtual memory dump as well as a PTD program dump prior to EOJ, enter

```
PTD; FILE SCODE(TITLE = PTD/CONF/PSS); STATION = 55; VALUE = 3

BEGIN JOB; PTD; FILE SCODE(TITLE = PTD/CONF/PE); STATION = 55
```

Remote Execution Examples

In the following examples PTD is executed at a logged-on Command and Edit (CANDE) terminal and all subsequent operator dialog occurs at that terminal.

To execute PTD from a logged-on CANDE terminal in conjunction with the test case s-code file named PTD/MAINT/PE, enter one of the following commands:

```
WFL BEGIN JOB; PTD; FILE SCODE(TITLE = PTD/MAINT/PE);
                    VALUE = 1

WFL BEGIN JOB; PTD; FILE SCODE (TITLE = PTD/MAINT/PE)

WFL BEGIN JOB; PTD; FILE SCODE(TITLE = PTD/MAINT/PE ON TESTPACK);
```

All subsequent operator dialog occurs at that terminal.

PTD Directives

When PTD is first executed, it performs some internal initialization steps that can take a few seconds to accomplish depending on the size of the test case code file and the number of other jobs in the mix. PTD displays the test case file name and release-level data, and then displays the following message:

```
AWAITING DIRECTIVE
```

PTD idles waiting for a directive to be entered.

Any valid PTD directive can now be entered, one directive per transmission. The RUN, REPEAT, and GO directives place PTD into immediate execution. When execution of the specified section or sections has been completed, PTD again displays the following message and idles until a new directive is entered:

```
AWAITING DIRECTIVE
```

Remember that if the ODT is used, GS must appear in column 1.

The following are the possible directives.

```
──┬─ DISPLAY ─┬─┬─────────────────────────┬────────────────────┤
  ├─ DUMP ────┤ └─ OFF ───────────────────┤
  ├─ IOTRACE ─┤                           │
  ├─ MONITOR ─┤                           │
  ├─ PGMDUMP ─┤                           │
  ├─ PRINT ───┤                           │
  ├─ TRACE ───┘                           │
  ├─ ERROR ─┬─ HALT ──────────────────────┤
  │         └─ IGNORE ────────────────────┤
  ├─ STATUS ──────────────────────────────┤
  ├─ RUN ─┬───────────────────────────────┤
  │       └─<section list>────────────────┤
  ├─ REPEAT ─┬────────────────┬─┬─────────┤
  │          └─<section list>─┘ └─ FOR ─<count>─┤
  ├─ GO ──────────────────────────────────┤
  └─ QUIT ────────────────────────────────┘
```

<section list>

```
 ┌←─/9\──────────────── , ──────────────────┐
─┴─<section number>─┬───────────────────────┴──────────────┤
                    ├─ TO ─┬─┬─<section number>─┤
                    └─ - ──┘ └─ END ────────────┘
```

The following text describes the meaning of each option:

DISPLAY

The DISPLAY directive displays messages on the ODT or remote terminal. The DISPLAY OFF form disallows such messages. When an ODT is being used, the DISPLAY OFF form causes PTD to close its PTDSPO file so that normal system display traffic can resume. By default, PTD displays messages on the ODT or remote terminal.

DUMP

The DUMP directive dumps the test case's stack and data memory to a printer file prior to EOJ. The DUMP OFF form stops such a dump. By default, PTD does not create this printer file unless a fatal execution error occurs. In that instance, the data is dumped unconditionally by PTD. That output should be submitted when reporting any test case or PTD problems.

IOTRACE

The IOTRACE directive displays input/output (I/O) information before and after every test case I/O operation. Prior to initiating the I/O, the MLI REQUEST and IOLENGTH are shown. After the I/O has completed, PTD displays the MLI RESULT, the number of data bytes transferred, and an IOP result.

The user should be aware that whenever the words IOP RESULT are used in the context of PTD or a Unisys test case, they refer to a value constructed strictly for the purposes of a programmatic, host-independent result descriptor that summarizes errors that occur within the host to MLI interface. The true IOP result descriptor can be obtained only by inspecting the test case I/O control block or blocks (IOCBs) as they complete. The address of the IOCB is made available through the use of the special I/O options, described in "PTD Operator Interruption" in this section.

The IOTRACE OFF form disallows the displaying of I/O information. By default, PTD does not display I/O information.

continued

continued

MONITOR	The MONITOR directive writes all display traffic to a printer file and places an asterisk (*) in column one of all operator entries. The MONITOR OFF form disallows the writing of all display traffic to a printer file. By default, PTD does not print display traffic.
PGMDUMP	The PGMDUMP directive allows PTD to dump itself via PROGRAMDUMP to a printer file prior to EOJ. The PGMDUMP OFF form disallows such a dump. By default, PTD does not allow a dump. However, any fatal execution error that occurs causes an unconditional program dump. The output from that dump should be submitted when reporting any test case or PTD problems.
PRINT	The PRINT directive causes PTD to print test-case-specified text to a printer file when the PRNT s-op is encountered. The PRINT OFF form disallows such printing. By default, no text is sent to a printer file.
TRACE	The TRACE directive allows PTD to write each test case s-op to a printer file after the s-op is executed. It is used for test case and PTD debugging. Since vast numbers of s-ops are executed in the usual test case, tracing them is not economical. The TRACE OFF form disallows such writing. By default, PTD does not write each test case s-op to a printer file.
ERROR	The test case notifies PTD of any I/O error via the EROP s-op. PTD then stops and idles or ignores it, depending on the setting of the ERROR directive. If the ERROR HALT form is used, PTD halts on any nonfatal I/O error and displays "STOPPED ON ERROR." The user is expected to enter any valid PTD directive at this time. If the GO directive is entered, PTD continues execution at the next s-op. By default, the ERROR form is used. The ERROR IGNORE form causes PTD to treat all nonfatal test case I/O errors as no-ops and execution continues uninterrupted. ***Note:*** *The test case can retry an I/O as part of its testing algorithm, but no test I/O is ever automatically retried by PTD or the operating system.*
STATUS	The STATUS directive enables PTD to display its status. Information included is the number of the section currently in execution, the toggles currently set, and the current setting of the ERROR option.

continued

continued

RUN

The RUN directive causes PTD to execute sections 100 to 2999.

The RUN <section lists> form allows the user to specify the sections or ranges of sections of a test case to be run. If a range of sections is specified, the second number must be greater than the first number. The END option can only appear as the last item in a <section list> and must be preceded by a starting section number. The use of END also causes PTD to go to EOJ when the execution of the final section has finished.

The following are examples:

```
RUN
RUN 3999
RUN 200
RUN 101, 201
RUN 203 TO 207,301-302,4000-END
```

REPEAT

The REPEAT directive executes sections 100 to 2999.

The REPEAT <section list> form repeats the section or range of sections an infinite number of times. If the range of sections is specified, the second number of the range must be greater than the first number. The END option can only appear as the last item in a <section list> and must be preceded by a starting section number. The use of END does not cause PTD to go to EOJ when the execution of the final section has finished.

The REPEAT <section list> FOR <count> form of the directive executes the specified section or range of sections a specified number of times. The maximum value of <count> is 999999999.

The following are examples:

```
REPEAT
REP F 3
REPEAT 102 FOR 10
REPEAT 101, 301, 102, 302
REPEAT 207-209, 101, 305 - 400
```

GO

The GO directive allows PTD to resume execution at the same point it was stopped. It could have stopped because of an error condition or an operator interruption. Refer to "PTD Operator Interruption"later in this section for more information. If PTD had finished executing a section list, the following message is displayed: "SECTION LIST COMPLETED. USE 'RUN' OR 'REPEAT'".

QUIT

The QUIT directive terminates PTD.

IOP PTD Directives

The IOP PTD directives are used primarily as aids in developing and debugging PTD and test cases. They also are useful in observing the flow of test case I/O operations or in stepping an I/O through the processor using the IOSTOP directive.

Selecting any of these directives, with the exception of IOSTOP and LINES, causes PTD to analyze the test case I/O currently in process at a selectable point in the execution of the test. The appropriate DEVICESTRUCTURE information is displayed on the ODT or remote terminal. DEVICESTRUCTURE is the programmatic I/O interface between the test case and PTD and is analogous to a file information block (FIB). Up to ten separate DEVICESTRUCTUREs can be declared in the test case program in the Peripheral Test Language (PTL). Each DEVICESTRUCTURE carries all information relating to the I/O device, the I/O request, and the result of the operation.

By default, these directives are disabled. When entering any of these directives an asterisk must precede them.

```
──┬─ HELP ── IODEBUG ─┬─┬───────┬──────────────────────────────────┤
  ├─ BOJ ─────────────┤ └─ OFF ─┘
  ├─ BOPEN ───────────┤
  ├─ AOPEN ───────────┤
  ├─ BIIO ────────────┤
  ├─ AIIO ────────────┤
  ├─ BCLOSE ──────────┤
  ├─ ACLOSE ──────────┤
  ├─ EOJ ─────────────┤
  ├─ IOSTOP ──────────┤
  ├─ ALL ─────────────┤
  ├─ IODEBUG ─────────┤
  └─ LINES ──<n>──────┘
```

The following text describes the meaning of each option:

*HELP IODEBUG	The *HELP IODEBUG directive displays an explanation of special I/O options. The *HELP IODEBUG OFF form disallows the display of these options.
*BOJ	The *BOJ directive displays each DEVICESTRUCTURE the first time it is touched after this directive has been set. The *BOJ OFF form disallows this display.
*BOPEN	The *BOPEN directive displays each DEVICESTRUCTURE before it is opened. The *BOPEN OFF form disallows this display.
*AOPEN	The *AOPEN directive displays each DEVICESTRUCTURE after it is opened. The *AOPEN OFF form disallows this display.
*BIIO	The *BIIO directive causes a display before the I/O is initiated out of each DEVICESTRUCTURE. The *BIIO OFF form disallows this display.
*AIIO	The *AIIO directive causes a display after the I/O has completed on each DEVICESTRUCTURE. The *AIIO OFF form disallows this display.
*BCLOSE	The *BCLOSE directive causes a display before each DEVICESTRUCTURE is closed. The *BCLOSE OFF form disallows this display.
*ACLOSE	The *ACLOSE directive causes a display after each DEVICESTRUCTURE is closed. The *ACLOSE OFF form disallows this display.

continued

continued

*EOJ	The *EOJ directive causes a display of each DEVICESTRUCTURE immediately prior to EOJ. The *EOJ OFF form disallows this display.
*IOSTOP	The *IOSTOP directive analyzes at each user-selected point. PTD programmatically stops after the DEVICESTRUCTURE display; entering any input or null reactivates PTD. If *BIIO has been set in addition to *IOSTOP, then after the I/O information display and attendant stop/start, PTD executes a conditional halt prior to firing the I/O (CHLT must be set on the machine). 4"1OCB1OCB1OCB" is placed in the B-register and the address of the IOCB to be fired is placed in the A-register. The *IOSTOP OFF form disallows such an analysis.
*ALL	The *ALL directive selects all the directives described above. The *ALL OFF form turns all the directives off.
*IODEBUG	The *IODEBUG directive analyzes TEST case I/O operations at selectable phases of execution. This directive is automatically selected if any of the above directives are selected. The *IODEBUG OFF form stops this analysis without disturbing the other directives selected. If the *IODEBUG form is subsequently selected, the analysis resumes.
*LINES <n>	The *LINES <n> directive sets the number of lines of data displayed in the analysis output, thus allowing the user to view an entire data buffer. Forty HEX digits (10 MLI words) are displayed per line. The normal display is one line. The *LINES <n> OFF form returns the display to one line.

Selecting Test Devices

Once running, PTD eventually encounters an OPEN s-op and displays one of the following messages:

```
ENTER DEVICE FOR <peripheral designation>

ENTER FILE NAME FOR <BADDISK designation>
```

In these messages, the <peripheral designation> and the <BADDISK designation> are strings of characters that PTD finds in the DEVICESTRUCTURE and displays on the screen. For example, in the card reader test, the display looks like the following:

```
ENTER DEVICE FOR CARDREADER
```

In the case of a peripheral/DLP test on EMS systems, PTD expects the user to enter the following response. The underlines show the minimum abbreviations for the words in this command.

```
<unit> VIA PATHID <pathid>
       --- ----
```

In the case of a peripheral/DLP test on Large Systems (LS) systems, PTD expects the user to enter the following response:

```
<unit> VIA CTL <CTL id>
```

Examples of possible responses are

```
CR12
PK57 VIA PATHID 17
PK 100 VIA CTL 2001
```

For BADDISK testing, PTD expects the user to enter the following response:

```
<file title> ON <family name>
```

An example of a possible response is

```
BADDISK/FMLYINX1/UNIT192/AD4050000 ON PACK
```

Reserving a Unit and Selecting a Path to That Unit

With few exceptions, the unit being tested must be reserved using the UR (Unit Reserved) system command. The exceptions are as follows:

- The pack scanner test does not require a reserved unit.
- A unit cannot be reserved for the BADDISK confidence test.

The selection of a path that the test case I/O operations are channeled through occurs in one of three ways:

- Explicit specification by the user (using the VIA keyword)
- Automatic selection by PTD
- Dynamic selection by the operating system

Because each test case has different requirements, path selection rules are enforced by PTD on the basis of the type of test being executed. Examples of the different situations are described below. Some test cases issue operations that are dangerous in an online environment unless the path has been reserved through the UR system command. This command has the following form on EMS systems:

```
UR <unit> PATHID <pathid>
```

The UR command has the following form on LS systems:

```
UR CTL <CTL id>
```

Other test cases, such as the BADDISK test, are not as dangerous and do not require a reserved path. Finally, explicit path selection in the online environment can interfere with the operating system's path selection algorithms for other users of the unit. This problem does not exist if the unit is reserved.

Whenever a reserved path is selected by PTD or is specified by the user, PTD assigns itself to that path. This assignment is reflected in the response to any subsequent OL (Display Label and Paths) system command. On EMS systems, that display shows PATHSTATUS as either ASSIGNED RESERVED ONLINE or ASSIGNED RESERVED OFFLINE. As long as at least one test case DEVICESTRUCTURE is open to a unit through the reserved path, the path remains assigned to PTD, and no other invocation of PTD is able to use it until the path is unassigned. However, in any given invocation of PTD, one path can be used by more than one DEVICESTRUCTURE. Refer to "Path Selection Example 2" further on in this section.

The following information describes the unique characteristics of some test cases.

PACKSCANNER Tests (PTD/CONF/PS)

Path selection by either the operator or PTD is allowed only if the unit has been reserved. The following messages are displayed by PTD when a reservation cannot be done or reserving a unit is not possible:

- If the operator attempts path selection to an unreserved unit using commands such as PK 65 VIA PATHID 3 or PK 65 VIA, PTD responds with the following message:

  ```
  PATH SELECTION NOT ALLOWED UNLESS UNIT IS RESERVED.
  ```

  To correct this situation, the operator should reserve the unit and select a path again.

- If the operator attempts path selection to a reserved unit, the specified path must be online and not assigned to any other stack. If the specified path is offline, PTD responds with

  ```
  PATH IS OFFLINE - TEST REQUIRES AN ONLINE PATH.
  ```

  If the specified path has been assigned to another stack, PTD responds with

  ```
  PATH IS IN USE.
  ```

- If a path selection is not attempted and the unit is not reserved, the path is selected by the operating system. PTD, however, ensures that at least one path that is online and not reserved is available. If an available path cannot be found, PTD displays

  ```
  AT LEAST ONE PATH MUST BE ONLINE AND NOT RESERVED.
  ```

- If a path selection is not attempted and the unit is reserved, PTD automatically searches for an online path in the following order:
  1. The first search is for an online, unreserved path. On EMS systems, the path with the highest PATHID value is selected. For example, if PATHID values 11 and 10 are available, then 11 is selected.
  2. The second search is for an online, reserved, unassigned path. On EMS systems, the path with the highest PATHID value is selected.
  3. If a suitable path is not found, PTD responds with

     ```
     NO PATHS TO THE UNIT ARE ONLINE.
     ```

BADDISK Tests (PTD/CONF/BD20X and PTD/CONF/BD2X5)

Since the test case requires a valid BADDISK file, the disk file header must be in memory so that the specified file can be located. Therefore, the unit cannot be reserved, and path selection is always left to the operating system. PTD ensures that at least one path is online and not reserved.

- If the operator attempts to specify a path, PTD responds with

  ```
  PATH SELECTION NOT ALLOWED FOR THIS TEST.
  ```

- If PTD cannot see available paths to the unit holding the file, the following message is issued:

  ```
  AT LEAST ONE PATH MUST BE ONLINE AND NOT RESERVED.
  ```

All Other Confidence Tests (such as PTD/CONF/IVR and PTD/CONF/MT)

A path does not necessarily have to be reserved. Path selection by the user is always allowed and an online, unassigned path is required. If path selection is attempted the unit must be reserved.

Note: *PTD/CONF/IVR is not available on the Micro A. Use the IVR program available from the Administrator menu.*

- If the user specifies an offline path, PTD responds with

  ```
  PATH IS OFFLINE - TEST REQUIRES AN ONLINE PATH.
  ```

- If the user specifies an online, reserved path but the path is assigned to another stack, PTD responds with

  ```
  PATH IS IN USE.
  ```

- If a path selection is not attempted and the unit is reserved, PTD automatically searches for an online path in the following order:
  1. The first search is for an online, unreserved path. On EMS systems, the path with the highest PATHID value is selected. For example, if PATHID values 11 and 10 are available, then 11 is selected.
  2. The second search is for an online, reserved, unassigned path. On EMS systems, the path with the highest PATHID value is selected.
  3. If no suitable path is found, PTD responds with

     ```
     NO PATHS TO THE UNIT ARE ONLINE.
     ```

All Maintenance Tests (such as PTD/MAINT/=)

Path selection by the user and PTD is allowed; however, the path must be reserved and not assigned to another stack. Remember that a unit must be reserved to make a path selection.

- If an unreserved path is specified, PTD responds with

  ```
  PATH IS NOT RESERVED.
  ```

- If a reserved path is chosen, but that path is assigned to another stack by another invocation of PTD, PTD responds with

  ```
  PATH IS IN USE.
  ```

- If no path selection is attempted, PTD searches for a path in the following order:
  1. The first search is for a path that is already assigned to this invocation of PTD. In this way, test cases with more than one DEVICESTRUCTURE value always use the same path unless the user specifies otherwise. On EMS systems, the highest numbered PATHID that meets this criteria is selected.
  2. The second search is for a reserved, online, unassigned path. On EMS systems, the path with the highest PATHID value is selected.
  3. The third search is for a reserved, offline, unassigned path. On EMS systems, the path with the highest PATHID value is selected.
  4. If no reserved path is found, PTD responds with

     ```
     NO PATHS TO THE UNIT ARE RESERVED.
     ```

     If all paths are reserved and assigned to other stacks, PTD responds with

     ```
     ALL PATHS TO THE UNIT ARE IN USE.
     ```

EMS Path Selection Example 1

The following are valid responses for the device-request in the 206/207 disk pack maintenance test PTD/MAINT/HT20X. In this example there are four hypothetical paths to unit PK49. Those paths are as follows:

PATHID Value	Status
11	RESERVED OFFLINE
10	RESERVED ONLINE
09	ON-LINE
08	RESERVED ONLINE

To the PTD request "ENTER DEVICE FOR DISK", the user can give any of the following responses:

Response	Result
PK 49	PTD automatically selects PATHID 10 since it is the reserved, online path with the highest PATHID value.
PK 49 VIA	PTD displays all paths to the unit in a form similar to the response to an OL (Display Label and Paths) system command and then asks the user to select a reserved path from the list.
PK 49 VIA PATHID 11	Since this is a maintenance test and both the unit and path are reserved, this is an acceptable specification.

EMS Path Selection Example 2

Suppose that, in the maintenance test above, PATHID 10 was selected as the path for the first DEVICESTRUCTURE opened. The following information is displayed for any subsequent OL PK49 system command:

PATHID Value	Status
11	RESERVED OFFLINE
10	ASSIGNED RESERVED ONLINE
09	ONLINE
08	RESERVED ONLINE

Now suppose that a second DEVICESTRUCTURE is opened. To the PTD request "ENTER DEVICE FOR DISK", the user can give any of the following responses:

Response	Result
PK 49	PTD automatically selects PATHID 10, since it is the highest PATHID value that has already been assigned to an open DEVICESTRUCTURE.
PK 49 VIA	PTD displays all paths to the unit in a form similar to the response to an OL (Display Label and Paths) system command and then asks the user to select a reserved path from the list.

continued

continued

Response	Result
PK 49 VIA PATHID 11	Since this is a maintenance test and both the unit and path are reserved, this is an acceptable specification.

LS Path Selection Example

The following are valid responses for the device-request in the 206/207 disk pack maintenance test PTD/MAINT/HT20X. In this example, there are two hypothetical paths to unit PK 100. These paths are as follows:

CTL	Status
2000	ONLINE
2001	RESERVED

To the PTD request "ENTER DEVICE FOR DISK", the user can give any of the following responses:

Response	Result
PK 100	PTD automatically selects CTL 2001 since it is the only reserved, online path.
PK 100 VIA	PTD displays all paths to the unit in a form similar to the response to an OL (Display Label and Paths) system command and then asks the user to select a reserved path from the list.
PK 100 VIA CTL 2001	Since this is a maintenance test and both the unit and path are reserved, this is an acceptable specification.

PTD Operator Interruption

PTD operates in a *fetch s-op/execute s-op* cycle and is sensitive to an operator interrupt request before each fetch operation. An interrupt request is serviced only when all outstanding test case I/O operations have been completed. When the operator-dialog device is an ODT, the operator can interrupt PTD by entering the following:

```
<SYSTEM/MAINTENANCE mix number> HI
```

An example is as follows:

```
1127 HI
```

When the operator dialog is through a REMOTE station, the HI statement must be preceded by a question mark in column 1. An example is as follows:

```
?9142 HI
```

When interrupted, PTD responds with

```
AWAITING DIRECTIVE
```

Any valid PTD directive is accepted now.

Test Case Example

An example of the printed output of the PE tape DLP maintenance test showing the use of the RUN, REPEAT, and GO directives on EMS systems appears below. The example also illustrates one of the path-selection techniques. In this example, PTD was initiated at an ODT and was executed by the entry

```
PTD; FILE SCODE(TITLE=PTD/MAINT/PE)
```

After the AWAITING DIRECTIVE message appeared, the MONITOR directive was set so that subsequent operator entries are flagged with the asterisk in column 1. When the MT17 system command was first entered, PTD informed the operator that the unit was not reserved; the operator reserved it using the UR MT17 system command. To direct this command to the system, the operator omitted the GS character. Next, the MT 17 VIA system command was entered to PTD, resulting in the path information display and path selection query. When the device was successfully open, the test case later reported errors that were caused by the operator not having mounted a tape on the drive. With the exception of the operator dialog, all messages are produced by the test case. The dialog between operator and PTD follows. Operator responses are shown with asterisks(*).

```
 LISTING FOR TESTCASE=PTD/MAINT/PE
 AWAITING DIRECTIVE
*PRINT
 AWAITING DIRECTIVE
*RUN 1Ø1
 STARTING SECTION Ø1Ø1
 --> ECHO OP - ALL BITS OFF
 ENTER DEVICE FOR MAGTAPE
*MT 17
 UNIT MT 17 : NOT RESERVED.
 ENTER DEVICE FOR MAGTAPE
     [COMMENT:  operator reserved MT 17]
*MT 17 VIA

 UNIT 'MT17':
    DLP ID =Ø4
    BASE NUMBER = ØØØ
    RELATIVE UNIT (W.R.T. DLP) = 1
               PATH INFORMATION
               ----------------

    PATHID  PROC   IOPPORT  LEMPORT  DLPNUM  PATHSTATUS
      Ø2      3       1        Ø        6    RESERVED, ONLINE
```

```
 ENTER DESIRED PATHID
*Ø2
 AWAITING DIRECTIVE
*REPEAT 1Ø2 FOR 3
 STARTING SECTION Ø1Ø2
 --> ECHO OP - ALL BITS ON
 STARTING SECTION Ø1Ø2
 --> ECHO OP - ALL BITS ON
 STARTING SECTION Ø1Ø2
 --> ECHO OP - ALL BITS ON
 AWAITING DIRECTIVE
*RUN 2ØØ-END
 STARTING SECTION Ø2ØØ
 --> OP CODES TEST
 ### TEST SEL = Ø2ØØ  TEST RUN = Ø2Ø1 ################
 OPCODE+VAR = 2F1ØØØ
 OPCODE = 2FØØØØ = TEST
 + UNIT NUMBER = 1(Ø1)
 & IDLENGTH = ØØØ6 CHARACTERS
 CYCLE=ØØ1 I/O=ØØØ13 ERR:IOP=ØØØ L=ØØØ RD=ØØ1 DATA=ØØØ
 RESULT DESCRIPTOR IS 411ØØØØØ
 RD WD1 4ØØØ = DESCRIPTOR ERROR
 RD WD1 Ø1ØØ = TAPE UNIT NOT READY
 RD WD1 ØØ1Ø = WRITE LOCKOUT
 RESULT DESCRIPTOR EX ØØ8ØØØØØ
 RD WD1 ØØ8Ø = BOT (BEGINNING OF TAPE)
 ### RD ERROR #################### RD ERROR #########
 STOPPED ON ERROR
*GO
 ### TEST SEL = Ø2ØØ  TEST RUN = Ø2Ø1 ################
 OPCODE+VAR = 2F1ØØØ
 OPCODE = 2FØØØØ = TEST
 + UNIT NUMBER = 1(Ø1)
 & IDLENGTH = ØØØ6 CHARACTERS
 CYCLE=ØØ2 I/O=ØØØ14 ERR:IOP=ØØØ L=ØØØ RD=ØØ2 DATA=ØØØ
 RESULT DESCRIPTOR IS 411ØØØØØ
 RD WD1 4ØØØ = DESCRIPTOR ERROR
 RD WD1 Ø1ØØ = TAPE UNIT NOT READY
 RD WD1 ØØ1Ø = WRITE LOCKOUT
 RESULT DESCRIPTOR EX ØØ8ØØØØØ
 RD WD1 ØØ8Ø = BOT (BEGINNING OF TAPE)
 ### RD ERROR #################### RD ERROR #########
 STOPPED ON ERROR
*QUIT
 *** EOJ PTD ***
```

Section 10
REPORT_LOG_ENTRIES

REPORT_LOG_ENTRIES is an exported MCP procedure that forwards copies of selected types of system log entries to user programs whenever the system generates those log entries. REPORT_LOG_ENTRIES allows the user program to specify the types of log entries that should be forwarded. If a selected log entry has been suppressed from the system log file, REPORT_LOG_ENTRIES still forwards it to the user program. The following are some of the types of applications that can be implemented through the use of the REPORT_LOG_ENTRIES procedure:

- Real-time billing applications. By monitoring Major Type 1, Minor Type 2 (EOJ) or Minor Type 4 (EOT) log entries, the application can track the total system resource usage accumulated by each usercode or charge code.
- Security auditing applications. By monitoring log entries such as Major Type 6, Minor Type 4 (Security Violation), the application can detect attempts to breach system security as they occur.
- User notification applications. For example, by monitoring Major Type 1, Minor Type 12 (Print Request Complete) log entries, an application can determine that a user's print request has completed and can send a notification to that user.

A related system-monitoring procedure exported by the MCP is the STATUS_CHANGE_REQUEST procedure, which is discussed in Section 11.

User Program Requirements for REPORT_LOG_ENTRIES

Because one of the parameters to REPORT_LOG_ENTRIES is a DCALGOL queue, user programs that call this procedure must be written in DCALGOL.

The REPORT_LOG_ENTRIES procedure can be called only by system software and by privileged programs. You can use the MP (Mark Program) system command to mark programs as privileged programs. The MP command is discussed in the *A Series System Commands Operations Reference Manual.*

Note: *Being a privileged user is not equivalent to using a program that is marked with privileged status. Being a privileged user is insufficient to allow the use of this exported MCP procedure.*

Declaring the REPORT_LOG_ENTRIES Procedure

The user program must contain a declaration of the MCPSUPPORT library and a declaration of the REPORT_LOG_ENTRIES procedure. These declarations appear as follows:

```
LIBRARY MCP(LIBACCESS=BYFUNCTION,FUNCTIONNAME="MCPSUPPORT");

REAL PROCEDURE REPORT_LOG_ENTRIES (Q,VAL,INFO,EXTRA);
    VALUE VAL;
    REAL VAL;
    QUEUE Q;
    ARRAY INFO, EXTRA[Ø];
    LIBRARY MCP;
```

The following parameters to this procedure are explained under the following headings in this section:

Parameter	Heading
Q	"Allocating an Intercom Queue for REPORT_LOG_ENTRIES" "Waiting for Log Entries" "Interpreting the Log Entries"
INFO	"Selecting Log Entry Types" "Using the Security Mask Field"
VAL	"Selecting Log Entry Types" "Using the Security Mask Field"
Procedure result	"Detecting Error Conditions for REPORT_LOG_ENTRIES"

The EXTRA parameter is reserved for future use, and is not further explained in this section.

Selecting Log Entry Types

The user program uses the INFO parameter to specify a list of log entry types; it uses the VAL parameter to specify whether monitoring is enabled or disabled for the log entry types listed in INFO.

The INFO parameter identifies log entries by major and minor types. For definitions of all the major and minor types logged by the system, refer to Section 12, "SUMLOG." The format of this parameter is explained in Table 10–1.

Table 10–1. INFO Parameter Format

Word	Contents
0	The number of valid words in the array. If this word stores a value of 2, then all the minor types associated with the major type in word 1 are selected.
1	The major type of the log entries.
2–*n*	The minor types of the log entries.

To enable monitoring of the log entries listed in the INFO parameter, the user program can assign a value of 1 to the VAL parameter. To discontinue monitoring of the log entries listed in the INFO parameter, the user program can assign a value of 0 to the VAL parameter. By repeatedly invoking the REPORT_LOG_ENTRIES procedure with varying INFO and VAL values, the user program can selectively modify the list of log entry types that are monitored.

For example, a user program could use the following statements to enable monitoring of major type 1, minor type 5 (File Open) and minor type 6 (File Close) log entries:

```
A[Ø] := 4;
A[1] := 1;
A[2] := 5;
A[3] := 6;
REPORT_LOG_ENTRIES(Q,1,A,X);
```

The same user program could then use the following statements to disable monitoring of major type 1, minor type 6 (File Close) log entries. Note that the second parameter, VAL, is assigned a 0 to indicate that the specified log entry type is to be disabled rather than enabled.

```
A[Ø] := 3;
A[1] := 1;
A[2] := 6;
REPORT_LOG_ENTRIES(Q,Ø,A,X);
```

The result of the previous example is that only major type 1, minor type 5 (File Open) entries continue to be monitored. The same user program can then use the following statements to enable monitoring of all major type 2 log entries:

```
A[Ø] := 2;
A[1] := 2;
REPORT_LOG_ENTRIES(Q,1,A,X);
```

After these statements are executed, all major type 2 log entries and all major type 1, minor type 5 (File Open) log entries will be monitored.

Using the Security Mask Field

On systems where InfoGuard security enhancement software is installed, the user program can use the security mask field of the VAL parameter to restrict the log entries returned based on their security relevance. The mask value causes the system to return a subset of the log entry types enabled by the INFO parameter.

The security mask field is located at [07:04]. The following are the meanings of the bits in this field:

Table 10–2. VAL Parameter Security Field

Bit	Contents
7	If set, reports log entries for actions that failed because they were security violations.
6	If set, reports log entries for actions that succeeded, but were security relevant. This category includes the following actions: • Some SETSTATUS entries • Use of the ??SECAD (Security Administrator Authorization) primitive system command • Assignments of attributes to tape volumes and directories • Operations on guard files • Changes to the USERDATAFILE • Initiation of a process with privileged or security administrator status • CANDE or MARC log-on of a user with privileged or security administrator status
5	If set, reports log entries for actions that failed and were not security relevant.
4	If set, reports log entries for actions that succeeded and were not security relevant.

If the value of this field is zero (all bits are reset), then all the log entries permitted by the INFO parameter are reported.

On non-InfoGuard systems, the system ignores the security mask field and returns all log entries allowed by the INFO parameter.

Allocating an Intercom Queue for REPORT_LOG_ENTRIES

If the DCALGOL queue passed to the REPORT_LOG_ENTRIES procedure has not been initialized, the system automatically initializes the queue as a new *intercom queue*. An intercom queue is a queue that provides communication among various system software processes. The system can dynamically create up to 47 intercom queues for user programs that call REPORT_LOG_ENTRIES or the STATUS_CHANGE_REQUEST procedure, which is described in Section 11.

Because the number of intercom queues is limited, you might wish to minimize the number of intercom queues the system must allocate for a user program. Fortunately, the system permits a user program to use a single intercom queue for more than one purpose. If a user program passes REPORT_LOG_ENTRIES an intercom queue that has already been initialized, the system uses the existing intercom queue rather than creating a new one. Thus, if you pass the same queue to the REPORT_LOG_ENTRIES procedure and the STATUS_CHANGE_REQUEST procedure, the system allocates only one intercom queue for their use.

Additionally, message control systems (MCSs) can use the DCALGOL *SETUPINTERCOM* function to explicitly initialize an intercom queue before passing it to the REPORT_LOG_ENTRIES procedure. The SETUPINTERCOM function is discussed in the *A Series DCALGOL Programming Reference Manual*.

Detecting Error Conditions for REPORT_LOG_ENTRIES

The REPORT_LOG_ENTRIES procedure returns a real value as the procedure result. Nonzero values indicate that an error occurred. The following are the possible values:

Result	Meaning
–1	The system could not allocate an intercom queue slot for the procedure. This error occurs if the system has already allocated 47 intercom queues to user programs calling the REPORT_LOG_ENTRIES and STATUS_CHANGE_REQUEST procedures.
–2 through –4	These values each indicate internal errors in the operating system.
–5	One or more of the minor types specified by the INFO parameter are invalid.
–6	The size of the INFO array is incompatible with the request. The INFO array must be at least two words long, and the array must be large enough to hold the number of entries specified in word 0 of the array.

Waiting for Log Entries

The REPORT_LOG_ENTRIES procedure exits once the system has inspected the VAL and INFO parameters and assigned the Q parameter to an intercom queue. Since the user program declares the actual parameters globally to the REPORT_LOG_ENTRIES procedure, the actual parameters continue to exist after the procedure exits. The user program can monitor the arrival of event messages by waiting on the QINSERTEVENT attribute of the queue. The QINSERTEVENT attribute is discussed in the *A Series DCALGOL Programming Reference Manual*.

Note: *Some log entries can be too large to be inserted in a DCALGOL queue. A program using REPORT_LOG_ENTRIES receives no notification of such log entries. The current size limit is 1536 words.*

Interpreting the Log Entries

The log entries returned in the Q parameter have the format explained in Table 10–3.

Table 10–3. Q Parameter Log Entry Format

Word	Contents	
0	If you are using the same intercom queue for more than one purpose, you can inspect the contents of this word to determine if the message was inserted by REPORT_LOG_ENTRIES or some other mechanism.	
	Field	**Contents**
	[47:08]	Contains the value 21.
	[39:08]	Contains the value 15.
	[15:16]	Contains the value 20.
1	Time of day in multiples of 2.4 microseconds.	
2–4	These words are not used	
5–*n*	The log entry. Log entries follow the formats documented in Section 12, "SUMLOG," except that they begin in word 5 instead of word 0. To make the log entry match the formats shown in the SUMLOG section, copy words 5–*n* of the message to a second array, starting at word 0 of the second array.	

REPORT_LOG_ENTRIES User Program Example

The following program uses the REPORT_LOG_ENTRIES procedure to monitor log-on activity on the system. An explanation follows the program.

```
% *****************************************************************%
%                                                                  %
%  PROGRAM:      FRIEND                                            %
%  DESCRIPTION:  NOTIFIES CALLER WHEN A SPECIFIED USER LOGS ON     %
%                                                                  %
% *****************************************************************%

BEGIN

LIBRARY MCP(LIBACCESS=BYFUNCTION, FUNCTIONNAME="MCPSUPPORT");
REAL PROCEDURE REPORT_LOG_ENTRIES (Q, VAL, INFO, EXTRA);
  VALUE VAL;
  REAL VAL;
  QUEUE Q;
  ARRAY INFO, EXTRA [0];
  LIBRARY MCP;

DEFINE
```

```
  USERCODEINDEX = [19:20] #,
  NEWLOGON = 1 #,
  MSGHEADER = 5 #,
  SIGNONCLASS = MSGHEADER + 5 #,
  LINKTOUSERCODE = MSGHEADER + 7 #,
  LINESIZE = 80 #,
  MAXNAMESIZE = LINESIZE #,
  MAXMINORTYPES = 100 #,
  QENTRYSIZE = 100 #,
  SCREENSIZE = 1920 #;

INTEGER  NAMESIZE,
         QINDEX;
ARRAY    NAME [0:MAXNAMESIZE-1],
         QENTRY [0:QENTRYSIZE-1],
         TYPES [0:MAXMINORTYPES-1];
EBCDIC ARRAY EA [0:SCREENSIZE -1];
BOOLEAN FOUND;
QUEUE   Q;
POINTER QPTR,
        EAPTR;
FILE    STDIO (KIND = REMOTE,
               MYUSE = IO,
               MAXRECSIZE = SCREENSIZE,
               UNITS = CHARACTERS,
               BLOCKSTRUCTURE = EXTERNAL);

TYPES [0] := 3;
TYPES [1] := 4;
TYPES [2] := 1;

IF (REPORT_LOG_ENTRIES (Q, 1, TYPES, TYPES) < 0) THEN
   WRITE (STDIO, SCREENSIZE,
          "ERROR: INTERCOM QUEUE SLOT UNAVAILABLE")
ELSE
  BEGIN
  WRITE (STDIO, SCREENSIZE, "ENTER FRIEND'S NAME:");
  READ (STDIO, MAXNAMESIZE, NAME);
  NAMESIZE := STDIO.CURRENTRECORD;
  REPLACE EAPTR:EA BY "WILL NOTIFY WHEN ",
                      POINTER(NAME) FOR NAMESIZE,
                      " LOGS ON";
  WRITE (STDIO, OFFSET(EAPTR), EA);

  FOUND := FALSE;
  WHILE NOT FOUND DO
    BEGIN
    WAIT (Q.QINSERTEVENT);
    REMOVE (QENTRY, Q);
    IF (QENTRY [SIGNONCLASS] = NEWLOGON) THEN
      BEGIN
      % FIND THE BEGINNING INDEX FOR THE USERCODE LINK
```

```
450       % MUST ADD "MSGHEADER" TO ADJUST FOR THE MESSAGE FORMAT
455       QINDEX := QENTRY[LINKTOUSERCODE].USERCODEINDEX + MSGHEADER;
460
465       % POINT TO THE 8-BIT ID LENGTH (SEE STANDARD FORM FILE TITLE)
470       QPTR := POINTER (QENTRY[QINDEX]) + 3;
475
480       IF (REAL(QPTR,1) = NAMESIZE) THEN
485         IF (QPTR + 1) = NAME FOR NAMESIZE THEN
490           BEGIN
495           REPLACE EAPTR:EA BY POINTER(NAME) FOR NAMESIZE,
500                               " HAS LOGGED ON !";
505           WRITE (STDIO, OFFSET(EAPTR), EA);
510           FOUND := TRUE;
515           END;
520       END;
525     END WHILE;
530   END;
535 END.
```

You must mark the object code file as privileged with the MP (Mark Program) system command.

The statement at line 345 invokes REPORT_LOG_ENTRIES as a function and checks to see if the result value is less than zero; if so, the system was unable to allocate an intercom queue, and the program displays an error message.

The following are the effects of the parameters passed at line 345:

- The 1 passed to the VAL parameter specifies that this invocation requests one or more new log entry types to be monitored.
- The TYPES array passed to the INFO parameter causes log entries of Major Type 4, Minor Type 1 (Log-On Entry) to be requested. This effect results because of the word 1 of TYPES was previously assigned a 4, and word 2 of TYPES was previously assigned a 1.
- The TYPES array passed to the EXTRA parameter fulfills the requirement of passing a real array to EXTRA, even though this parameter currently has no meaning.

If the intercom queue was allocated successfully, the statements at lines 370 through 400 request the user to enter a name, and then they display a confirmation message. The name is stored in the NAME array.

Execution then enters the WHILE loop at lines 415 through 525. The statement at line 425 causes the program to wait for a message to appear in intercom queue Q. The statement at line 430 causes the program to remove the message from the queue and place it in array QENTRY.

The statements at lines 435 through 520 make use of the Major Type 4, Minor Type 1 (Log-On Entry) description in Section 12, "SUMLOG." Because the log entry starts at word 5 of QENTRY, the program adds 5 to the offset of the log entry words. Thus, the statement at line 435 inspects word 10 of QENTRY for the sign-on class, because the sign-on class word of the log entry is normally word 5.

The statement at line 455 assigns the offset of the usercode entry to QINDEX. To get this offset, the program looks in the usercode link word (word 7 in the log entry, thus word 12 in QENTRY). Table 12-1, "Link to Variable Items," shows that field [19:20] of a link word stores the offset. The program therefore extracts the value from field [19:20] and adds 5.

The usercode is stored in standard form, as described in the discussion of the DISPLAYTOSTANDARD function in the *A Series DCALGOL Programming Reference Manual.* Standard form starts with three bytes that are useful for describing file titles, but not necessary for describing usercodes. The statement at line 470 assigns QPTR to a position just past this irrelevant data, pointing at the start of the length field.

The statement at line 480 checks to see if the length of the usercode matches that of the usercode previously requested. If the length matches, the statement at line 485 compares the contents of the two usercodes. If a match is found, the statements at lines 490 through 515 notify the user that his or her friend has logged on.

Section 11
STATUS_CHANGE_REQUEST

STATUS_CHANGE_REQUEST is an exported MCP procedure that allows user programs to monitor selected types of system events. The types of events that can be monitored include changes in process status and changes in system initialization status. STATUS_CHANGE_REQUEST accepts a parameter that you can use to specify the types of events to be monitored. The STATUS_CHANGE_REQUEST feature can be a useful element of supervisor programs and programs that display system status information to operators.

Because STATUS_CHANGE_REQUEST is primarily a notification mechanism, it returns only basic information about the events it reports. For detailed information about most of these events, the program can use other interfaces such as REPORT_LOG_ENTRIES, GETSTATUS, and SYSTEMSTATUS. The REPORT_LOG_ENTRIES interface is described in Section 10. GETSTATUS is described in the *A Series GETSTATUS/SETSTATUS Programming Reference Manual.* SYSTEMSTATUS is described in the *A Series SYSTEMSTATUS Programming Reference Manual.*

User Program Requirements for STATUS_CHANGE_REQUEST

Because one of the parameters to STATUS_CHANGE_REQUEST is a DCALGOL queue, user programs that call this procedure must be written in DCALGOL.

The STATUS_CHANGE_REQUEST procedure can be called only by system software and by privileged programs. You can use the MP (Mark Program) system command to mark programs as privileged programs. The MP command is discussed in the *A Series System Commands Operations Reference Manual.*

Note: *Being a privileged user is not equivalent to using a program that is marked with privileged status. Being a privileged user is insufficient to allow the use of this exported MCP procedure.*

Declaring the STATUS_CHANGE_REQUEST Procedure

The user program must contain a declaration of the MCPSUPPORT library and a declaration of the STATUS_CHANGE_REQUEST procedure. These declarations appear as follows:

```
LIBRARY MCP(LIBACCESS=BYFUNCTION,FUNCTIONNAME="MCPSUPPORT");
REAL PROCEDURE STATUS_CHANGE_REQUEST(Q,SCEVENTS);
      ARRAY SCEVENTS[*];
      QUEUE Q;
      LIBRARY MCP;
```

The parameters to this procedure are explained under the following headings in this section:

Parameter	Heading
SCEVENTS	"Selecting Events for Notification"
Q	"Allocating an Intercom Queue for STATUS_CHANGE_REQUEST" "Interpreting the Event Messages"
Procedure result	"Detecting Queue Allocation Errors for STATUS_CHANGE_REQUEST"

Selecting Events for Notification

To request notification of a particular type of event, the user program sets the corresponding bit in Word 0 of the SCEVENTS parameter. Table 11–1 shows the correspondence between bit positions and event types.

Table 11–1. Event Types

Word	Bit Position	Event
0	[22:01]	A process terminated. This category includes all processes, whether or not they are WFL processes.
	[21:01]	An operator has entered a system command, or the system has displayed a response to a system command.
	[19:01]	A WFL job terminated.
	[18:01]	A WFL job was initiated.
	[17:01]	The MCP data comm tables have been deallocated, and all data comm has been terminated.
	[16:01]	System initialization has reached the point where the data comm tables have been built. At this point, the MCP has read the DATACOMINFO file and has built station, line, and MCS tables. At this point, it is possible for an MCS to initialize its primary queue.
	[15:01]	A WFL job was discontinued while it was in a job queue.
	[14:01]	A WFL job entered a job queue.
	[13:01]	A library process resumed execution.
	[12:01]	A process was initiated. This category includes all processes, whether or not they are WFL processes.
	[11:01]	A process froze and became a library process.
	[10:01]	System initialization has reached the point where WFL jobs can be initiated.

continued

Table 11–1. Event Types (cont.)

Word	Bit Position	Event
	[09:01]	There was a change to the number of user processes that have a particular database open.
	[07:01]	A process has just created and closed a new tape file, producing tape label information.
	[06:01]	A unit changed status (for example, became online or offline or was purged).
	[05:01]	A process issued an RSVP message.
	[04:01]	A process changed priority.
	[03:01]	A process other than a WFL process terminated.
	[02:01]	A process other than a WFL process was initiated.
	[01:01]	A process was scheduled.

Allocating an Intercom Queue for STATUS_CHANGE_REQUEST

If the DCALGOL queue passed to the Q parameter has not been initialized, the system automatically initializes the queue as a new *intercom queue.* An intercom queue is a queue that provides communication among various system software processes. The system can dynamically create up to 47 intercom queues for user programs that call STATUS_CHANGE_REQUEST or the REPORT_LOG_ENTRIES procedure, which is described in Section 10.

Because the number of intercom queues is limited, you might wish to minimize the number of intercom queues the system must allocate for a user program. Fortunately, the system permits a user program to use a single intercom queue for more than one purpose. If a user program passes STATUS_CHANGE_REQUEST an intercom queue that has already been initialized, the system uses the existing intercom queue rather than creating a new one. Thus, if you pass the same queue to the STATUS_CHANGE_REQUEST procedure and the REPORT_LOG_ENTRIES procedure, the system allocates only one intercom queue for their use.

Additionally, message control systems (MCSs) can use the DCALGOL *SETUPINTERCOM* function to explicitly initialize an intercom queue before passing it to the STATUS_CHANGE_REQUEST procedure. The SETUPINTERCOM function is discussed in the *A Series DCALGOL Programming Reference Manual.*

Detecting Queue Allocation Errors for STATUS_CHANGE_REQUEST

STATUS_CHANGE_REQUEST is a real procedure that returns a value of –1 if the system cannot allocate an intercom queue slot for the procedure. This error occurs if the system has already allocated 47 intercom queues to user programs calling the STATUS_CHANGE_REQUEST and REPORT_LOG_ENTRIES procedures.

Waiting for Event Messages

The STATUS_CHANGE_REQUEST procedure exits once the system has inspected the SCEVENTS parameter and assigned the Q parameter to an intercom queue. Since the program declares the actual parameters globally to the STATUS_CHANGE_REQUEST procedure, the actual parameters continue to exist after the procedure exits. The program can monitor the arrival of event messages by waiting on the QINSERTEVENT attribute of the DCALGOL queue that was passed to the Q parameter. The QINSERTEVENT attribute is discussed in the *A Series DCALGOL Programming Reference Manual.*

Interpreting the Event Messages

The event messages returned in the Q parameter are in an encoded format, with separate fields that indicate the type of event, the time of day, and sufficient information to uniquely identify the event.

Note: *The format of the event messages is heavily dependent on internal operating system structures. While care is taken to minimize format changes, Unisys reserves the right to change these message formats without a 3-release warning.*

All messages have the basic structure explained in Table 11–2.

Table 11–2. Event Message Basic Format

Word	Contents	
0	If you are using the same intercom queue for more than one purpose, you can inspect the contents of this word to determine if the message was inserted by STATUS_CHANGE_REQUEST or some other mechanism.	
	Field	**Contents**
	[47:08]	Contains the value 21.
	[39:08]	Contains the value 15.

continued

Table 11–2. Event Message Basic Format (cont.)

Word	Contents	
	[15:16]	Contains a value identifying the type of event. The possible event types, and their meanings, are listed in Table 11–3, "Event Message Variable Formats." The value 20 is never used by STATUS_CHANGE_REQUEST; a value of 20 in this field indicates that the message was inserted by REPORT_LOG_ENTRIES.
1	Time of day in multiples of 2.4 microseconds.	
2	The size of the message starting in word 5.	
3	The date and time, in the same format as that returned by the TIME(7) function in ALGOL. The value is divided into the following fields:	
	Field	**Contents**
	[47:12]	Year (1900 through 1999)
	[35:06]	Month (1 through 12)
	[29:06]	Day (1 through 31)
	[23:06]	Hour (0 through 23)
	[17:06]	Minute (0 through 59)
	[11:06]	Second (0 through 59)
	[05:06]	Day of the week (0 = Sunday, 1 = Monday, and so on)
4	This word is not used.	
5–*n*	Contents vary, depending on the event type reported in field [15:16] of word 0. Refer to Table 11–3 for information about these words. Word 6 duplicates the value from field [15:16] of word 0, unless otherwise specified in Table 11–3.	

Table 11–3 describes the portions of the event message format that vary according to the type of event that is being reported. The event types shown correspond to the value of field [15:16] of word 0.

Table 11–3. Event Message Variable Formats

Event Type	Word	Contents	
1 (Process Scheduled)	7	Contains the SERIAL word for the scheduled process.†	
2 (Non-WFL Process Initiated)	7	Contains the SERIAL word for the task.†	
3 (Non-WFL Process Terminated)	7	Contains the SERIAL word for the task.†	
	8	Contains the value of the HISTORY task attribute of the task. For the format of this value, refer to the *A Series Task Attributes Programming Reference Manual.*	
4 (Priority Changed)	7	Contains the SERIAL word for the process.†	
	8	The new value of the PRIORITY task attribute.	
5 (RSVP Issued)	7	Contains the SERIAL word for the process.†	
	8	Bit 47 of this word is always set, indicating that the process is suspended. The following bits indicate the allowable responses:	
		16	NOTOK (Do Not Reactivate) system command
		15	FA (File Attribute) system command
		14	QT (Quit) system command
		13	No response is necessary; the process is executing a DCKEYIN function and will resume automatically when the function is completed.
		12	AX (Accept) system command
		11	OK (Reactivate) system command (must be entered from an ODT)
		10	OU (Output Unit) system command
		9	UL (Unlabeled) system command
		8	FM (Form Message) system command
		7	IL (Ignore Label) system command
		6	CL (Clear) system command
		5	SV (Save) system command
		4	FR (Final Reel) system command
		3	OF (Optional File) system command

† For information about the SERIAL word format, refer to Table 11–4.

continued

Table 11–3. Event Message Variable Formats (cont.)

Event Type	Word	Contents	
		2	OK (Reactivate) system command (can be entered from an ODT or a remote terminal)
		1	RM (Remove) system command
		0	DS (Discontinue) system command
6 (Unit Status Changed)	7	Field [11:12] contains the unit type of the unit that changed status. The values associated with each unit type are the same as the values documented for the KIND file attribute in the *A Series File Attributes Programming Reference Manual.*	
	8	Field [15:16] contains the unit number of the affected unit.	
7 (Tape Label Created)	5	Contains the value 9.	
	6	Contains the value 7.	
	7	The SERIAL word for the process that created the tape label.†	
	8	Indicates which of the remaining words of the message store valid file attribute values. Each word stores a valid value if the corresponding bit of this word is set. To determine the bit number corresponding to a particular word, subtract 5 from the word number; thus, bit 4 corresponds to word 9.	
	9	The SAVEFACTOR file attribute value.	
	10	The DENSITY file attribute value.	
	11	**Field**	**Contents**
		[37:14]	The CYCLE file attribute value.
		[23:08]	The VERSION file attribute value.
		[14:15]	The FILESECTION file attribute value.
	12	The index, relative to word 5, of the word where the tape file title begins. (Add 5 for the correct index.) The tape file title is stored in standard form.‡	
	13	The SERIALNO file attribute value.	
	14	The UNITNO file attribute value.	

† For information about the SERIAL word format, refer to Table 11–4.

‡ For information about standard form, refer to the description of the DISPLAYTOSTANDARD function in the *A Series DCALGOL Programming Reference Manual.*

continued

Table 11–3. Event Message Variable Formats (cont.)

Event Type	Word	Contents
	15	The KIND file attribute value.
	16	The index, relative to word 5, of the word where the NAME task attribute of the job is stored. (Add 5 for the correct index.) The NAME value is stored in standard form.‡
	17	The index, relative to word 5, of the word where the NAME task attribute of the process opening the file is stored. (Add 5 for the correct index.) The NAME value is stored in standard form.‡
	18–*n*	Information pointed to by words 12, 16, and 17.
9 (Database User Count Changed)	7	Contains the SERIAL word for the database stack.†
	8	The number of user processes with that database open.
10 (WFL Available)	None affected	
11 (Library Froze)	7	The SERIAL word for the library process.†
12 (Process Initiation)	5	Information for internal use by the operating system.
	6	The SERIAL word for the process.†
	7	The PRIORITY task attribute value of the process.
	8	Not used.
	9–11	The USERCODE task attribute of the process. Field [47:08] of word 9 stores the length of the usercode. The usercode itself starts at bit 39 of word 9, and can extend through words 10 and 11.
	12–14	The compiler name, if this is a compilation. Field [47:08] of word 9 stores the length of the compiler name. The compiler name itself starts at bit 39 of word 9, and can extend through words 10 and 11.
	15–*n*	The NAME task attribute of the process, expressed in standard form.‡
13 (Library Thawed)	7	The SERIAL word for the library process.†
14 (WFL Job Queued)	7	The MIXNUMBER task attribute of the WFL job.
	8	The CLASS task attribute of the WFL job.
15 (WFL Job Discontinued from Queue)	7	The MIXNUMBER task attribute of the WFL job.

† For information about the SERIAL word format, refer to Table 11–4.

continued

‡ For information about standard form, refer to the description of the DISPLAYTOSTANDARD function in the *A Series DCALGOL Programming Reference Manual.*

Table 11–3. Event Message Variable Formats (cont.)

Event Type	Word	Contents
16 (Data Comm Tables Built)	None Affected	
17 (Data Comm Tables Deallocated)	None Affected	
18 (WFL Job Initiated)	7	The SERIAL word for the WFL job.†
19 (WFL Job Terminated)	7	The SERIAL word for the WFL job.†
21 (System Command or Response)	5	The time of day, expressed in units of 2.4 microseconds.
	6–*n*	The system command or response text. The text is expressed as EBCDIC characters, but can include undisplayed characters that are used to control the operator display terminal (ODT).
22 (Process Terminated)	6	The SERIAL word for the process.†
	7	The HISTORY task attribute of the process.
	8	The STOPPOINT task attribute value of the process. For information about STOPPOINT, refer to the *A Series Task Attributes Programming Reference Manual.*
	9–10	These words are reserved for future use.
	11–13	The USERCODE task attribute of the process. Field [47:08] of word 11 stores the length of the usercode. The usercode itself starts at bit 39 of word 11, and can extend through words 12 and 13.
	14	Index to the location of the compiler name, if this process is a compilation. You must add 5 to the value stored here to determine the correct word index. If the process is not a compilation, this word stores a 0 (zero).
	15–19	Information for internal use by the operating system.
	20	The beginning of the NAME task attribute of the process, expressed in standard form.‡

† For information about the SERIAL word format, refer to Table 11–4.

‡ For information about standard form, refer to the description of the DISPLAYTOSTANDARD function in the *A Series DCALGOL Programming Reference Manual.*

Table 11–3, “Event Message Variable Formats,” includes several references to the SERIAL word. Table 11–4 describes the format of this word.

Table 11–4. SERIAL Word Format

Field	Contents
[15:16]	The mix number of the process.
[31:16]	The mix number of the job for the process.
[47:12]	The stack number of the process.

STATUS_CHANGE_REQUEST User Program Example

The following program uses the STATUS_CHANGE_REQUEST procedure to monitor the initiation of processes. An explanation follows the program.

```
$INSTALLATION 1
% *****************************************************************%
%                                                                  %
%  PROGRAM:      BOJ AND BOT NOTICE                                %
%  DESCRIPTION:  WHENEVER A PROCESS BEGINS, THIS PROGRAM SENDS     %
%                INFORMATION CONCERNING THE PROCESS TO THE SCREEN  %
%                                                                  %
% ******************************************************************

BEGIN

LIBRARY MCP(LIBACCESS=BYFUNCTION, FUNCTIONNAME="MCPSUPPORT");
REAL PROCEDURE STATUS_CHANGE_REQUEST (Q, SCEVENTS);
  ARRAY SCEVENTS[*];
  QUEUE Q;
  LIBRARY MCP;

DEFINE
  PROGNAMEMSGSIZE = [47:8] #,
  TASKNO = [15:16] #,
  JOBNO = [31:16] #,
  BOJNOTICE = [12:1] #,
  MIXNUMSIZE = 6 #,
  MAXEVENTS = 100 #,
  MAXQENTRIES = 100 #,
  MAXMSGSIZE = 100 #,
  TIMESIZE = 8 #,
  LINESIZE = 80 #,
  SCREENSIZE = 24*80 #,
  SERIALI = 6 #,
  PROGNAMEMSGI = 15 #,
  TIMEI = 1 #;

INTEGER  JOBNUM,
         TASKNUM;
POINTER  PMPTR,
         PPTR,
         TPTR,
         OPTR;
ARRAY    SCEVENTS[0:MAXEVENTS-1],
         PROGNAMEMSG[0:MAXMSGSIZE-1],
         PROGNAME[0:MAXMSGSIZE-1],
         THETIME[0:TIMESIZE-1],
         OUTPUT[0:LINESIZE-1],
         QENTRY[0:MAXQENTRIES-1];
QUEUE    Q;
FILE     STDIO (KIND = REMOTE,
```

```
                MYUSE = IO,
                MAXRECSIZE = SCREENSIZE,
                UNITS = CHARACTERS,
                BLOCKSTRUCTURE = EXTERNAL);
PROCEDURE PUT_TIME(PTR, TIM);
  VALUE TIM;
  REAL TIM;
  POINTER PTR;
 BEGIN
  TIM := TIM * 2.4@-6;
  REPLACE PTR BY TIM DIV 3600 FOR 2 DIGITS, ":",
    TIM MOD 3600 DIV 60 FOR 2 DIGITS, ":",
    TIM MOD 60 FOR 2 DIGITS;
 END PUT_TIME;
SCEVENTS[0].BOJNOTICE := 1;
IF (STATUS_CHANGE_REQUEST(Q, SCEVENTS) < 0) THEN
  WRITE(STDIO, SCREENSIZE, "ERROR: INTERCOM QUEUE SLOT UNAVAILABLE")
ELSE
  BEGIN
  WRITE(STDIO, SCREENSIZE, "PROGRAM INITIALIZED");
  WHILE TRUE DO
    BEGIN
    WRITE (STDIO, SCREENSIZE, "...WAITING FOR BOJ");
    WAIT (Q.QINSERTEVENT);
    REMOVE (QENTRY, Q);
    REPLACE POINTER(PROGNAMEMSG) BY POINTER(QENTRY[PROGNAMEMSGI])
      FOR MAXMSGSIZE;
    PMPTR := POINTER(PROGNAMEMSG);
    PPTR := POINTER(PROGNAME);
    STANDARDTODISPLAY(PMPTR, PPTR);
    JOBNUM := QENTRY[SERIALI].JOBNO;
    TASKNUM := QENTRY[SERIALI].TASKNO;
    TPTR := POINTER(THETIME);
    PUT_TIME(TPTR, QENTRY[TIMEI]);
    OPTR := OUTPUT;
    REPLACE OPTR:OUTPUT BY "PROGRAM NAME:  ";
    REPLACE OPTR:OPTR BY POINTER(PROGNAME)
      FOR (LINESIZE-OFFSET(OPTR)) WHILE NEQ ".";
    WRITE(STDIO, OFFSET(OPTR), OUTPUT);
    OPTR := OUTPUT;
    REPLACE OPTR:OUTPUT BY "        JOB#:  ";
    REPLACE OPTR:OPTR BY TASKNUM FOR MIXNUMSIZE DIGITS;
    WRITE(STDIO, OFFSET(OPTR), OUTPUT);
```

```
795     OPTR := OUTPUT;
800     REPLACE OPTR:OUTPUT BY "       TASK#:  ";
805     REPLACE OPTR:OPTR BY TASKNUM FOR MIXNUMSIZE DIGITS;
810     WRITE(STDIO, OFFSET(OPTR), OUTPUT);
815
820     OPTR := OUTPUT;
825     REPLACE OPTR:OUTPUT BY "TIME STARTED:  ";
830     REPLACE OPTR:OPTR BY TPTR FOR TIMESIZE;
835     WRITE(STDIO, OFFSET(OPTR), OUTPUT);
840     END;
845   END;
850 END.
```

You must mark the object code file as privileged with the MP (Mark Program) system command.

The *$INSTALLATION 1* statement at line 100 gives the program access to an installation intrinsic that supports the STANDARDTODISPLAY statement used later in the program.

The statement at line 620 in this program turns on bit 12 in the SCEVENTS parameter, thus requesting notification of all process initiations that take place on the system. The meanings of the bits in this parameter are documented in Table 11–1, "Event Types," in this section.

The statement at line 630 invokes STATUS_CHANGE_REQUEST as a function and checks to see if the result value is less than zero; if so, the system was unable to allocate an intercom queue, and the program displays an error message.

If the intercom queue was allocated successfully, the program enters the WHILE loop at lines 655 through 840. The statement at line 670 causes the program to wait for a message to appear in intercom queue Q. The statement at line 675 causes the program to remove the message from the queue and place it in array QENTRY.

The statements at lines 680 through 715 extract information from the message stored in QENTRY. These statements make use of the format information for event type 12 (Process Initiation) that is documented in Table 11-3, "Event Message Variable Formats."

Thus, lines 680 through 685 extract the process name beginning at word 15 of QENTRY. Because the process name is stored in standard form, the statement at line 700 invokes STANDARDTODISPLAY to convert the name to display form. Line 710 extracts the job number stored in field [31:16] of word 6. Line 715 extracts the task number stored in field [15:16] of word 6.

Line 730 extracts the time of day from word 1 of QENTRY. This portion of the message format is documented in Table 11–2, "Event Message Basic Format." The PUT_TIME procedure converts the time from units of 2.4 microseconds to an HH:MM:SS format for display.

The remainder of the program displays the process name, job number, task number, and time at a remote terminal.

Section 12
SUMLOG

This section serves as a reference to users of the system log (SUMLOG) information on Unisys A Series systems. The system log is a disk file stored on the family designated by the DL SUMLOG form of the DL (Disk Location) system command. This log contains entries concerning jobs previously run, operating system activity, and other associated information regarding the past status of the machine environment.

Understanding Log Releases

Periodically, the system has to create a new SYSTEM/SUMLOG file and save the old SYSTEM/SUMLOG file under a different title. This action is referred to as a *log release*. The system performs a log release whenever any of the following situations occur:

- When the existing SUMLOG becomes 95 percent filled. Because the SUMLOG file holds up to 100,000 records, an automatic log release takes place at approximately 95,000 records.
- When an operator enters a TL (Transfer Log) system command.
- When an operator enters the DL LOG <family> form of the DL (Disk Location) system command.

When the system performs a log release for any of these reasons, the system changes the title of the existing SYSTEM/SUMLOG file to a title of the following form:

```
SUMLOG/<system serial number>/<date>/<log number>
```

In this title format, the <log number> is a 6-digit integer that is one greater than that used at the previous log release. The operating system maintains the <log number> in a table on the halt/load family. The <log numbers> increase until the next cold-start or until a new halt/load family is used.

Controlling Log Contents

The information stored in the system log is grouped into units called *log entries*. Log entries are classified into various *major types*, based on the general type of information they store. Within each major type, log entries are classified into *minor types* based on the specific types of information they store.

Each major type and minor type has both a name and a number. For example, log entries that record task initiations are Major Type 1 (Job or Task Entry), Minor Type 3 (BOT). The system places a log entry of this type in this system log each time a task is initiated on the system.

The system administrator can control the contents of the system log in the following ways:

- By specifying which of the possible log entry types should be logged by the system
- By specifying whether anonymous file accounting or anonymous task accounting should be used
- By creating programs that insert installation-defined entries in the system log

The following subsections describe these methods for controlling system log contents.

Selecting the Entry Types to be Logged

The LOGGING (Logging Options) system command can be used to control which log entries are logged in the system SUMLOG file and/or job file, including installation-defined entries that are written to the SUMLOG file. This way, the site manager can bypass unneeded information entries and minimize the volume of log entries.

The default list of entry types to to be logged is given in the *Default Log Option Setting* column in Table 12–5, "Log Entry Classes," later in this section. Deviations from the default settings affect the system performance. Any event that is added causes a performance degradation, depending upon the frequency of the event occurrence. Directory actions (Major Type 16) are an example of a frequently occurring event. Conversely, any events that are deleted decrease the operating system time spent on logging, resulting in a performance improvement. It is recommended that at least the following events always be logged:

Major Type	Minor Type	Event
0	1	Establish identity (EI) (Not selectable)
1	1	Beginning of job (BOJ)
1	2	End of job (EOJ)
1	3	Beginning of task (BOT)
1	4	End of task (EOT)
2		All maintenance entries
4	1	MCS session log-on (LOGON)
4	2	MCS session log-off (LOGOFF)
6	1	Halt/load entry
6	3	SETSTATUS entry

The *LOGGING* * form of the LOGGING command can be used to restore the default logging settings at any time.

The LG (Log for Mix Number) system command and the user structure element LOGSELECT (accessible through the MAKEUSER utility) can generate additional logging on specified tasks and usercodes. (LG is only available on an InfoGuard system.) This part of the selective logging mechanism can include additional audit information at a fraction of the cost that it takes to log the event at all times for all tasks.

Requesting Copies of Configuration Files

The system creates copies of certain configuration files whenever log entry types associated with those configuration files are issued. These configuration files include COMS load files, the DATACOMINFOFILE, guard files, and the USERDATAFILE. You can use the LOGGING (Logging Options) system command to enable logging of the log entry types that cause these copies to be created.

The copies of configuration files are created on the family specified by the DL LOGS option of the DL (Disk Location) system command. Each file has a title of the form "SUMLOG/<type>/<date>/<time>", where <date> and <time> are the date and time the copy was initiated. The <date> is in the format HHMMSS, and the <time> is in the format MMDDYY. The <type> value is determined from the major and minor type of the log entry as follows:

Major/Minor Type	Description	<type>
6, 11	New USERDATA install	USERDATAFILE
16, 1	File creation and copy (The copy is made only if the file is a guard file.)	GUARDFILE
17, 2	Data comm installation and copy	DATACOMINFO
18, 12	Load file copy	COMSLOAD

For the format of these log entries, refer to "Log Entry Types" later in this section.

Using Anonymous Task and File Accounting

Anonymous task accounting and anonymous file accounting are two methods of improving system performance by reducing the number of log entries the system generates.

A process is eligible for anonymous task accounting if all the following conditions are true:

- The process is a task—that is, a dependent process.
- The process has the same usercode as its parent.
- The process is not initiated directly from a CANDE or MARC session or from a WFL job.

If anonymous task accounting is in effect for a particular task, the system does not create Major Type 1, Minor Type 3 (BOT) or 4 (EOT) log entries for that task. Further, the system does not issue BOT or EOT system messages for the task, and the task does not appear in the C (Completed Mix Entries) system command display.

All user processes are eligible for anonymous file accounting. If anonymous file accounting is in effect for a particular process, the system does not create Major Type 1, Minor Type 5 (File Open) or 6 (File Close) log entries for that process. Instead, the system issues a Major Type 1, Minor Type 25 (File Statistics) entry when the process terminates.

If anonymous task accounting and anonymous file accounting are both in effect for a particular process, then the system does not issue the File Statistics entry when the process terminates. Instead, the system adds the statistics to the File Statistics entry that is issued when the parent terminates.

A programmer can use the DEPTASKACCOUNTING and FILEACCOUNTING task attributes to request anonymous task accounting or anonymous file accounting for a process. The system administrator can use the ACCOUNTING (Resource Accounting) system command to specify a systemwide default value for these task attributes. The system administrator can also specify default DEPTASKACCOUNTING and FILEACCOUNTING values in individual usercode definitions. For further information about these task attributes, refer to the *A Series Task Attributes Programming Reference Manual.*

Defining Installation Log Entries

Most log entries are generated and handled internally by the MCP. Entries generated internally call the MCP procedure WRITELOG. The MCP procedure WRITELOG is used to mark log entries with a timestamp and send them to the system log file and to the job file, if available.

Selected entries are generated outside the MCP. Software outside the MCP cannot write to the SUMLOG directly, because the SUMLOG is marked by the MCP as a read-only file. However, software outside the MCP can generate log entries by calling MCP library object MCP_LOGGER, which is also available as the intrinsic MCSLOGGER.

Log entries generated by MCP_LOGGER or MCSLOGGER cannot exceed 1023 words in length.

MCP_LOGGER can be used to log the following events, shown in Table 12–1 according to the capabilities and identity of the calling program:

Table 12–1. Events Logged by MCP_LOGGER

Major Type	Minor Type	Entry Class	Caller Status
4	ALL	MCS	MCS, SSL, IR
6	7	MISCELLANEOUS Controller Command	IR
7	ALL	INSTALLATION	MCS, SSL, IR, PU
11	ALL	BNA	SSL
13	ALL	BNA Version 2	SSL
18	ALL	COMS CONFIGURATION	MCS, SSL, IR
25	ALL	HLCN	PU
27	ALL	TCP/IP	PU
28	ALL	NMS	PU
30	ALL	SNA	PU

Legend
MCS Message control system (CANDE, COMS, or MARC)
SSL System support library–BNA
IR Independent runner (CONTROLLER)
PU Privileged program

The following example program logs an installation entry using MCP_LOGGER. It identifies the syntax and illustrates the use of the external logging interface from an ALGOL program. It must be a privileged program.

```
BEGIN
LIBRARY MCPSUPPORT(LIBACCESS=BYFUNCTION);
BOOLEAN PROCEDURE MCP_LOGGER(LEN,REC);
    VALUE   LEN;
    INTEGER LEN;
    ARRAY   REC[*];
    LIBRARY MCPSUPPORT;
DEFINE
    LOGTYPEX    = 3 #,
        ERR_ANYFAULT     = ALL1 #,
        ERR_TOO_LONG     = 1 #,
        ERR_MCS          = 2 #,
        ERR_NOT_LOGGED   = 3 #,
        ERR_MAJOR        = 4 #,
        ERR_MIXNUMBER    = 5 #,
        ERR_JOBFILE      = 6 #,
        ERR_TOO_SHORT    = 7 #,
```

```
        ERR_BADLINK      = 8 #,
        LOGFIXEDLENF= [47: 6] #,
        LOGLENGTHF  = [41:1Ø] #,
        LOGRESULTF  = [31: 2] #,
        LOGVISIBLEF = [29: 2] #,
        LOGMAJORF   = [27:12] #,
        LOGMAJINST  = 7 #,
        LOGMINORF   = [15:16] #;
ARRAY LOG_ENTRY[Ø:29];
BOOLEAN BOOL;
REAL RSLT;


%%%% SET UP CONTENTS OF LOG_ENTRY[4] .. END
%%%% ASSUME TOTAL LENGTH = 3Ø; LENGTH OF FIXED PART = 6;

LOG_ENTRY[LOGTYPEX] := Ø &          6 LOGFIXEDLENF
                         & LOGMAJINST LOGMAJORF
                         &         12 LOGMINORF;
IF BOOL := MCP_LOGGER(3Ø,LOG_ENTRY) THEN
  BEGIN
    RSLT := REAL (BOOL).[11:8];
    DISPLAY("MCP_LOGGER CALL ERROR  "  !! STRING(RSLT,*));
  END;
END.
```

Using Log Analysis Utilities

Several utilities are available for analyzing and reporting on system log and job log contents. You can modify the output of some of these utilities by creating a support library that customizes log analysis.

Using Standard Utilities

The following are log analysis utilities and features provided by Unisys:

Utility	Explanation
LOGANALYZER	This utility reports the contents of selected types of log entries. LOGANALYZER is well suited for historical reporting and security auditing. For further information, refer to Section 7, "LOGANALYZER."
LOGGER	This utility reports on totals and averages of information contained in selected types of log entries. LOGGER is well suited to billing and workload analysis. For further information, refer to Section 8, "LOGGER."

continued

continued

Utility	Explanation
System Management Facility II (SMFII)	The LOGCONSOLIDATOR component of SMFII extracts workload analysis and hardware reliability information from the system log. The QUERY component of SMFII can analyze and format this information into a wide variety of reports. For further information, refer to the *A Series System Management Facility II (SMFII) Resource Management Operations Reference Manual* and the *A Series System Management Facility II (SMFII) Query Operations Guide.*
JOBFORMATTER	This system library generates job summaries when jobs terminate. For information about how to control the issuing of job summaries, refer to the process history discussion in the *A Series Task Management Programming Guide.*

Customizing Log Analysis

In addition to producing job summaries, JOBFORMATTER exports procedures that LOGANALYZER uses to perform log analysis. You can customize the analysis performed by JOBFORMATTER, and hence the contents of job summaries and LOGANALYZER reports.

To customize JOBFORMATTER log analysis, you must write a support library called SITESUPPORT that exports a procedure called PINSTALLATION. JOBFORMATTER attempts to call the PINSTALLATION procedure; if the first call is successful, JOBFORMATTER calls PINSTALLATION for each job log or system log entry. If the PINSTALLATION procedure returns a result indicating that it did not analyze the entry, JOBFORMATTER performs its standard analysis; otherwise it skips the entry.

The following actions must be taken in order for site-customized log analysis to be implemented:

First, the site analysis code must be contained in a library program as a real typed procedure. Its interface is declared (using ALGOL) as follows:

```
REAL PROCEDURE PINSTALLATION(LBUF,BUF,GL_RTLINE,INFO);
    VALUE INFO;
    ARRAY LBUF,BUF[*];
    PROCEDURE GL_RTLINE(N);
        VALUE N;
        REAL  N;
        FORMAL;
    REAL INFO;
```

The parameters are described as follows:

Parameter	Description
LBUF	The output buffer used by GL_RTLINE. LBUF is 43 words long. When GL_RTLINE is called, it writes the first 132 characters of the buffer. If there are nonblank characters beyond byte 132 in LBUF, then GL_RTLINE writes them on succeeding print lines as needed.
BUF	Contains a complete log entry for analysis.
GL_RTLINE	The output routine. IF N = 0 THEN LBUF is written; if N > 0 then N blank lines are written. This procedure can be called as many times as needed for complete analysis of the log entry.
INFO.[47:47]	Not used.
INFO.[0: 1]	Analysis requestor. If this bit is off, then the analysis is for a job file summary. If this bit is on, it is a call by LOGANALYZER for a system summary log.
<return value>	If the <return value> is 0, then the log entry was not analyzed; if it is 1, then the log entry was analyzed.

This procedure should analyze and report the log entry in BUF by calling the output procedure GL_RTLINE one or more times, passing the analyzed text in LBUF, and using the information in INFO. Note that since the user of PINSTALLATION is the SHAREDBYALL library JOBFORMATTER, the effective sharing of the SITESUPPORT library is therefore SHAREDBYALL, regardless of the declared sharing in the library itself.

The library must be installed as the support function SITESUPPORT, using the SL (Support Library) system command.

Writing Log Analysis Programs

If you need log analysis capabilities significantly different from those provided by A Series utilities, you can write your own log analysis application. Before writing an application, you need to understand the log structure information presented in the following subsection.

Depending on the security configuration of your system, your log analysis application might be able to read the system log directly, or the application might have to use the SDASUPPORT library interface to read the system log. Details of this interface are given under "Using the SDASUPPORT Library" later in this section.

Understanding Log Structure

The system log, SYSTEM/SUMLOG, is designed by default as a protected disk file of up to 99 rows of 1000 segments per row, consisting of a varying number of 30-word physical records. The following subsections explain

- The relationship between logical log entries and physical records
- The contents of the first four words of each entry

- The order in which entries are placed in the log
- The format of LINK words used in log entries
- How to write an application that runs successfully even if the format of log entries changes

Understanding Log Entries and Physical Records

The number of words of information in each entry differs not only for the different log entry types but also between specific entries of the same type (because they include program names, file names, and other variable-length information). The exact number of records in an entry is entered in the first word (word 0, field [39:8]) of the 30-word record of the entry. The length of the entry in words is contained in field [41:10] of Word 3 for entries containing up to 1022 words. For entries that are 1023 words or longer, the field has all ten bits on. In this case, use the format of the log entry, described in "Log Entry Types" later in this section, to determine the length of the log entry.

The log is organized so that bad information in one record or a disk error does not hamper retrieval of other records. According to this organization, all logical log entries are split into contiguous, fixed-size physical records of 30 words each. The first word of every physical record contains a record group description that describes its position in the logical log entry. The MCP procedure WRITELOG breaks up the logical log entry into physical log records.

When analyzing the log, a program must reassemble the logical entry from the physical records found in the log before it uses the word numbers listed for the log entries under "Log Entry Types" in this section. Those word numbers refer to the words as contained in the logical entry, not as they occur after the entry has been broken up into physical records.

The relationship between the words of the logical entry and their location in the physical log record is shown in the following table:

Physical Record Number	Physical Record Word 0	Logical Entry Data Words
1	Record 1 of 3	0–29
2	Record 2 of 3	30–58
3	Record 3 of 3	59–87

Understanding the First Four Words

The first four words (0 through 3) of every logical log entry contain the information shown in Table 12–2, "First Four Log Entry Words." Note that the record group description is found in Word 0 of every physical log record as well as in Word 0 of the logical log entry.

Table 12–2. First Four Log Entry Words

Word	Description
0	Record group description
	[47: 8] The cardinality of this record within the logical log entry ("this is record m of n") beginning with 1
	[39: 8] The total number of physical records in this logical log entry
	[31:16] The JOBNUMBER value of the process that caused the log entry
	[15:16] The MIXNUMBER value of the process that caused the log entry
1	Julian date in binary
2	Time of day, in units of 2.4 microseconds
3	Log entry type code
	[47: 6] Length of fixed part of entry in words
	[41:10] Length of the entire log entry in words, unless all the bits are on (value equal to 1023), in which case the entry is longer than 1022 words. For these long entries, use the format described in "Log Entry Types" later in this section to determine the length of the log entry.
	[31: 2] Result indicator (InfoGuard only)
	0 = Successful actions that are not security relevant
	1 = Failed actions that are not security violations
	2 = Successful actions that are security relevant, including SETSTATUS entries requiring security administration; ??SECAD primitive; security attribute assignments (volume and directory); attachments of guard files to files; USERDATA changes; BOJ/BOT/LOGON of privileged user, security administrator, or privileged program
	3 = Failed actions that are security violations, including general security violations and MCS security violations
	[29: 2] Record visibility (InfoGuard only)
	0 = Unspecified
	1 = Public–maintenance and halt/load records
	2 = Private–records that are not public or secured
	3 = Secured–MCS security violations
	[27:12] Major type. This value gives a general indication of the type of log entry.

continued

Table 12–2. **First Four Log Entry Words** (cont.)

Word	Description
	[15:16] Minor type. This value indicates the specific log entry type. Note that minor type values are unique only within a given major type. For a list of log types, refer to Table 12–5, "Log Entry Classes."

Understanding Variations from Chronological Order

The physical records of each log entry are always contiguous in the SUMLOG. However, these entries are not always logged in strict chronological order. Aside from the effects of the TR (Time Reset) and DR (Date Reset) system commands, unusual timestamps might occur when a halt/load is done after part of the system has been powered off and on. Furthermore, when contention occurs over which task is to write to the log next, the entries are made in the order of the task's priorities, not in the time sequence of their requests. The physical records of a single log entry are never split across two different logs, even if the TL (Transfer Log) system command was entered while the entry was being written. Entries are always made in the current SUMLOG file. Therefore, entries for a single job or task can occur in all the SUMLOG files that were active while the job or task was running.

Understanding the LINK Words

Each log entry is divided into a fixed part and a variable part. All items of information are given a reserved location in the fixed part. However, this location is not adequate to contain items such as task names and file titles, whose sizes vary. In these cases, the location in the fixed part is used to contain a LINK to the actual information, which is placed in the variable part of the log entry. Because all the fixed information precedes the variable information, items or their LINKs are always found at the same index in the fixed part, and for any given level of the log, all records of the same type have the same fixed-part size and structure.

The consistency in indexing of the fixed part of a given type of entry helps preserve upward compatibility between Mark releases because new items of log information are always added at the end of the fixed part and are thus invisible to log programs that do not expect to find them. Downward compatibility is somewhat more difficult to achieve because analysis programs must first check the length of the fixed part in order to determine if the information they seek is present in the version of the log under analysis.

Several formats exist for the LINK word used to refer to variable-length information. These include a standard LINK word format and specialized LINK formats for Hardware and Software Configuration records, I/O Exception records, BNA, and Volume Directory records. These formats are discussed in the following subsections.

Format of Standard LINK Words

This standard format for the LINK provides a word index to the start of the variable-length information and is the format referred to whenever the word LINK occurs in most log-entry descriptions. The layout of the LINK word is shown in Table 12–3, "Link to Variable Items."

Table 12–3. Link to Variable Items

Field	Meaning
[39:20]	Length of this item in words.
[19:20]	Index of word in log entry where this item starts. It is found in the variable part of the log entry.

Format of Hardware and Software Configuration LINK Words

Major Type 2, Minor Type 17 (Hardware Configuration) and 18 (Software Configuration) entries require a LINK containing more information than the standard LINK contains. This additional information includes the type of variable-length data referred to and the amount of data available. To accommodate this information, a special format for a LINK word and a structure for a log record using these LINKs has been defined. The layout of this LINK word is as follows:

Field	Meaning
[47: 8]	LINK_TYPE
[39: 4]	LINK_VERSION
[35:12]	ITEM_COUNT
[23: 8]	ITEM_WIDTH
[15:16]	START_INDEX

This type of LINK describes a contiguous group of array entries known as a *region*. LINKs can appear within a region that is described by a LINK; this recursive use creates a tree structure. This LINK structure is described in the following paragraphs.

The LINK_TYPE field of the LINK specifies the kind of information encoded in this region.

The LINK_VERSION field identifies the version of the format of the information. Unisys changes the LINK_VERSION when the offset or format of an item changes between system Mark releases.

The START_INDEX field of the LINK specifies the array index, in words, where the region described by the LINK starts. Specific items residing in the region have specific offsets from the START_INDEX of that region.

The ITEM_COUNT field contains the number of items found in the region described by the LINK.

Regions can be either homogeneous or heterogeneous. If a region is homogeneous, all items that it contains must have the same size and format. The size is stored in the ITEM_WIDTH field of the LINK. The ITEM_WIDTH value might be in units of words or bytes, depending on the LINK_TYPE. To determine the units used for ITEM_WIDTH for a given LINK_TYPE, refer to the description of that particular LINK_TYPE.

If the region is heterogeneous, the ITEM_WIDTH field is 0 and the ITEM_COUNT is given in words.

The direct part of a structure contains entries with a known fixed offset. The indirect part of a structure must be reached by following LINKs from a direct part. An indirect part can become a direct part when the LINK describing the indirect part is followed. This direct part has all the properties of any other direct part.

Every user program should always verify that the LINK_TYPE and LINK_VERSION fields contain the expected values. If they do not, the program should not attempt to decode the information. The ITEM_COUNT field should be checked to verify that the expected number of items is present. The ITEM_WIDTH field should also be checked to verify that the item size conforms to the initial specifications. As long as the LINK_TYPE and LINK_VERSION match the values that the decoding program expects, the data are in a format that the program understands. The ITEM_COUNT or ITEM_WIDTH, or both, can be larger or smaller than expected. The program should be coded to ignore any extra data in an item or unexpected items in a region.

Format of I/O Exception LINK Words

Major Type 2, Minor Type 21 (I/O Exception) entries make use of a LINK word with the following format:

Table 12–4. I/O Exception LINK Word Format

Field	Meaning
[47:16]	Version.
[31:16]	Length of the item.
[15:16]	Offset of the word where the item starts.

Format of BNA and HLCN LINK Words

Major Type 11 (BNA) and Major Type 25 (HLCN) entries use a LINK word format that is almost the same as the standard LINK format. The only difference is that, in BNA and HLCN LINK words, the length field is usually given in units of bytes rather than words. These LINK words are specified to be either of type *byte* or type *word* in the record descriptions for these major types.

Format of Volume Directory LINK Words

For Major Type 15, Minor Types 6 through 11 (Volume Directory) entries, the link format is as follows:

Field	Meaning
[23:12]	Length of the item in words
[11:12]	Index to the start of the item, minus 3.

Preparing for Log Entry Format Changes

In general, the log entry formats described in this section are taken from the current system software release. However, some log entry types that are no longer used on the current release continue to be documented in this section. This section also documents some individual words and fields that are no longer used. This information is retained to assist people in the analysis of system logs generated under the previous system software release. This section makes no attempt to document log entry types, words, or fields that have been unused for two or more Mark releases.

Note: *Although every effort is made to keep the fields as described in this section, new system software releases can introduce new log entry types or add new words to an existing log entry. Because of this fact, you should write log analysis programs to handle unrecognized major and minor types.*

The addition of new words to existing log entries is typically transparent to existing log analysis programs, because words are normally added at the end of the fixed part of the record. For example, in the BOJ record, additions would be made after the nineteenth word. A log analysis program can determine the current length of a log entry by checking the length field of Word 3.

The information in this section applies equally to all A Series systems except where otherwise noted.

Understanding Log Access Security

The ability of a process to read from the SUMLOG depends to some extent on whether the process has *direct read access* to the SUMLOG file. A process receives direct read access if at least one of the following conditions is true:

- The process is privileged.
- The SUMLOG file has a SECURITYTYPE file attribute value of PUBLIC.
- The SUMLOG file has a SECURITYTYPE file attribute value of GUARDED or CONTROLLED, and has an associated guard file that allows the process read access to the log.

The SECURITYTYPE value of the system log file is inherited from the old log file when a new log file is created. The default SECURITYTYPE value is specified by the NONUSERFILES system security option, which can have a value of PUBLIC or

PRIVATE. You can use the SECOPT (Security Options) system command to assign system security options such as NONUSERFILES.

A process with direct read access to the system log file can read the log by simply opening the log file and executing read statements. A process with direct read access can also read the log file indirectly by invoking procedures in the SDASUPPORT library.

If a process lacks direct read access to the system log file, and InfoGuard security enhancement software is *not* installed on the system, the process cannot read the system log at all. If InfoGuard security enhancement software *is* installed, the process can read the system log by invoking procedures in the SDASUPPORT library. However, the SDASUPPORT library performs filtering that limits the log entries visible to the process.

For more information about the SDASUPPORT library, and the filtering it performs, refer to the following subsection.

Using the SDASUPPORT Library

The SDASUPPORT system library provides a generalized interface to any SUMLOG file. The SDASUPPORT library is used by LOGANALYZER, LOGGER, and SMFII to access log files. The SDASUPPORT library can also be used by any application program that reads a SUMLOG file.

To install the SDASUPPORT library on non-InfoGuard systems, use the SL (Support Library) system command to associate *SYSTEM/SDASUPPORT with the function SDASUPPORT. On InfoGuard systems, use the SL command to associate *SYSTEM/IGSDASUPPORT with the function SDASUPPORT.

Understanding SDASUPPORT Filtering

On non-InfoGuard systems, only processes with direct read access to the log can use SDASUPPORT. The user process has access to all the export objects described under "Objects Exported by SDASUPPORT," later in this section. However, the following values cannot be used for the STYPE parameter to the LOG_SELECT function: 3 (USERCODE SET), 4 (ACCESS CODE SET), 5 (FILTERING SET), 6 (RESULT SET), and 7 (CHARGECODE SET). These values return an error. SDASUPPORT does not perform any filtering on non-InfoGuard systems.

On InfoGuard systems, all processes can use SDASUPPORT. The user process has access to all the export objects described under "Objects Exported by SDASUPPORT," later in this section. The user process can use the LOG_SELECT procedure to request filtering based on usercode, access code, or charge code. Filtering causes only log entries generated by processes with the specified usercode, access code, or charge code to be visible.

Note: By default, SDASUPPORT performs filtering based on the usercode of the user process. This is true regardless of whether the user process has direct read access to the system log. Processes that have direct read access can disable usercode filtering by invoking LOG_SELECT with an STYPE value of 3 (USERCODE SET) and an SBUFFER value of 48"00". Processes that lack direct read access are not allowed to disable usercode filtering.

Objects Exported by SDASUPPORT

The SDASUPPORT library exports the following procedures:

Procedure Name	Function
LOG_ATTRIBUTE	Reads selected log file attributes
LOG_CLOSE	Closes the log file
LOG_GET	Reads the next visible log file entry
LOG_OPEN	Sets the title and opens the log file
LOG_READ	Reads the next visible log file record
LOG_SEEK	Seeks the log file record
LOG_SELECT	Selects access modes for the log file
LOG_SKIP	Skips log file records
SDASUPPORT_VERSION	Returns the version of the SDASUPPORT library

The following subsections discuss the procedures exported by SDASUPPORT.

LOG_ATTRIBUTE

The LOG_ATTRIBUTE procedure reads selected file attributes of the SUMLOG file. The procedure is set up as follows:

```
INTEGER PROCEDURE LOG_ATTRIBUTE(ATT,ATTPTR);
   VALUE   ATT,ATTPTR;
   INTEGER ATT;
   POINTER ATTPTR;
```

In this procedure, the ATT parameter specifies the file attribute number of the file attribute that is being interrogated. In languages such as ALGOL and COBOL74, the VALUE function can be used to return the number of a file attribute; for example, the ALGOL expression VALUE(FILEKIND) returns the attribute number 58. For a complete list of file attribute numbers, refer to the *A Series File Attributes Programming Reference Manual.*

For integer and mnemonic file attributes, the procedure result contains the value of the attribute. For character string file attributes, the procedure result contains the length of the attribute value, and the value itself is returned at the location pointed to by ATTPTR.

The file attributes that can be interrogated are

- AREASIZE
- BLOCKSIZE
- LASTRECORD
- MINRECSIZE
- MAXRECSIZE
- NEXTRECORD
- RECORD
- SECURITYGUARD
- SECURITYTYPE
- SECURITYUSE
- TITLE

If LOG_ATTRIBUTE is used to interrogate any other file attributes, a value of –1 is returned as the procedure result.

If the TITLE file attribute is interrogated, and the selected log file is the current system SUMLOG file, then bit [47:1] of the procedure result is set. The procedure result also contains the length of the TITLE value.

LOG_CLOSE

The LOG_CLOSE procedure closes a log file and is set up a follows:

```
INTEGER PROCEDURE LOG_CLOSE;
```

The results are as follows:

Value	Meaning
0	File closed

LOG_GET

The LOG_GET procedure returns the next visible entry in the log file. The procedure is set up as follows:

```
BOOLEAN PROCEDURE LOG_GET(BUFFER);
    ARRAY   BUFFER[Ø];
```

The parameter for this procedure is a buffer for the return of the complete log entry. The procedure returns the read logical result descriptor if the log was opened, or an end-of-file (EOF) condition if the log was not opened.

It is better to use the LOG_GET procedure rather than the LOG_READ procedure because the LOG_GET procedure returns a complete log entry. The BUFFER array is resized if necessary for the entry to fit. The return value contains the length of the entry, in words, in field [47:20].

LOG_OPEN

The LOG_OPEN procedure sets the title and opens a log file with the given title. The procedure is set up as follows:

```
INTEGER PROCEDURE LOG_OPEN(TITLE_PTR);
    VALUE   TITLE_PTR;
    POINTER TITLE_PTR;
```

TITLE_PTR points to a period (.) or pointer to a file title. An empty title opens the current SUMLOG file. The results are as follows:

Value	Meaning
0	File already opened
1	File successfully opened
2	Attribute error on title
3	File not a valid log file
4	Usercode not allowed
5	File not available

LOG_READ

The LOG_READ procedure returns the first record of the next visible entry in the log file. This procedure is set up as follows:

```
BOOLEAN PROCEDURE LOG_READ(BUFFER);
    ARRAY   BUFFER[Ø];
```

The parameter for this procedure is a buffer for the return of the log file record data. The procedure returns the read logical result descriptor if the log was opened, or an end-of-file (EOF) condition if the log was not opened.

The LOG_READ procedure is similar to the LOG_GET procedure, except that LOG_READ reads only the first record in an entry, whereas LOG_GET reads the entire entry.

LOG_SEEK

The LOG_SEEK procedure repositions the log file record pointer at the record with the specified relative record number. The procedure is set up as follows:

```
PROCEDURE LOG_SEEK (RECORD);
    VALUE RECORD;
    INTEGER RECORD;
```

Note that, if LOG_SEEK leaves the record pointer positioned in the middle of a multirecord log entry, the next read operation automatically skips to the start of the next logical log entry. If LOG_READ is used for the next read operation, the first record of the next visible log entry is returned. If LOG_GET is used for the next read operation, all of the next visible log entry is returned.

LOG_SELECT

The LOG_SELECT procedure selects access modes for the log file. The procedure is set up as follows:

```
INTEGER PROCEDURE LOG_SELECT(STYPE,SBUFFER);
    VALUE   STYPE;
    REAL    STYPE;
    ARRAY   SBUFFER[Ø];
```

In this procedure, STYPE contains the selection case, and SBUFFER contains the selection data, which is case dependent.

The results are case dependent; negative is failure.

The following functions can be selected:

STYPE Value	Name	Function
0	CLEAR	Clears the specification of a timeframe that might have been set up by a call to LOG_SELECT with STYPE = 1.
1	TIMEFRAME SET	Causes only log entries to be returned that fall in the time range specified in the SBUFFER parameter. SBUFFER[0] and SBUFFER[1] contain the start date and start time, and SBUFFER[2] and SBUFFER[3] contain the end date and end time. If SBUFFER[0] contains a value of 7, the dates and times are in SBUFFER[1] and SBUFFER[3] in TIME(7) format. Otherwise, the dates are in SBUFFER[0] and SBUFFER[2] in Julian date format, and the times are in SBUFFER[1] and SBUFFER[3] in minutes as an integer value.

continued

continued

STYPE Value	Name	Function
2	LOGRANGE GET	Returns the date and time of the first record in the log file in SBUFFER[0] and SBUFFER[1], and the date and time of the last record in the log file in SBUFFER[2] and SBUFFER[3]. If SBUFFER[0] contains a value of 7, the dates and times are returned in SBUFFER[1] and SBUFFER[3] in TIME(7) format. Otherwise, the dates are returned in SBUFFER[0] and SBUFFER[2] in Julian date format, and the times are returned in SBUFFER[1] and SBUFFER[3] in minutes as an integer value.
3	USERCODE SET	This function is available only with InfoGuard SDASUPPORT. If the function is not available, a –1 error code is returned. The SBUFFER parameter contains a USERCODE as a standard form name. If the SDASUPPORT library functions in filtered mode (see FILTERING SET), filtering is performed using this USERCODE. If the caller does not have direct read access to the SUMLOG file to be opened, and the USERCODE is not equal to the caller's, an open error (4) is returned by LOG_OPEN. If SBUFFER contains an invalid name, a –2 error code is returned. If a log file is currently open, a –3 error code is returned. Three special names are valid: if SBUFFER[0].[47: 8] contains 48"00", no filtering based on USERCODE is performed. If SBUFFER[0].[47:24] contains 48"030000", the search target is for entries without an associated USERCODE. If SBUFFER[0].[47:24] contains 48"030100", the search target is the USERCODE of the calling process.
4	ACCESS CODE SET	This function is available only with InfoGuard SDASUPPORT. If the function is not available, a –1 error code is returned. The SBUFFER parameter contains an accesscode as a standard name. If the SDASUPPORT library functions in filtered mode (see FILTERING SET), filtering is performed using this accesscode. If SBUFFER contains an invalid name, a –2 error code is returned. If a log file is currently open, a –3 error code is returned.

continued

continued

STYPE Value	Name	Function
		Three special names are valid: if SBUFFER[0].[47:08] contains 48"00", no filtering based on ACCESSCODE is performed. If SBUFFER[0].[47:24] contains 48"030000", the search target is for entries without an associated ACCESSCODE. If SBUFFER[0].[47:24] contains 48"030100", the search target is the ACCESSCODE of the calling process.
5	FILTERING SET	This function is available only with InfoGuard SDASUPPORT. SBUFFER[0] contains one of the following values: • 0 = No change; the current setting is returned. • 1 = Enter a filtering mode in which LOG_GET and LOG_READ return only those entries that match the USERCODE, CHARGECODE, and ACCESSCODE of the user process. • 2 = Enter a filtering mode similar to that specified by a value of 1, except that LOG_GET and LOG_READ are also allowed access to public entries (such as maintenance and halt/load entries). • 3 = Exit filtering mode if the caller has direct read access to the SUMLOG file; otherwise, the filtering value of 2 is used. This function can be selected only when no log file is open. If a log file is open, a –3 error code is returned.
6	RESULT SET	This function is available only with InfoGuard SDASUPPORT. If the function is not available, a –1 error code is returned. SBUFFER[0].[3:4] contains the following RESULT value bit mask: • Bit [3: 1], if set, causes security violation entries to be visible. • Bit [2: 1], if set, causes security relevant entries to be visible. • Bit [1: 1], if set, causes failure entries to be visible. • Bit [0: 1], if set, causes successful entries to be visible. The default is all RESULT types are passed.

continued

continued

STYPE Value	Name	Function
7	CHARGECODE SET	This function is available only with InfoGuard SDASUPPORT. If the function is not available, a –1 error code is returned. The SBUFFER parameter contains a chargecode as a standard name. If the SDASUPPORT library functions in filtered mode (see FILTERING SET), filtering is performed using this chargecode. If SBUFFER contains an invalid name, a –2 error code is returned. If a log file is currently open, a –3 error code is returned. Three special names are valid: if SBUFFER contains 48"00", no filtering based on CHARGECODE is performed; SBUFFER contains 48"030000", the search target is for entries without an associated CHARGECODE; SBUFFER contains 48"030100", the search target is the CHARGECODE of the calling process.

The following STYPE values are valid only if the SDASUPPORT library was compiled with the compiler option DIAGNOSTICS set:

STYPE Value	Option	Description
1021	NO WARNING	Suppresses unwanted or inappropriate warnings. Currently the only warning that can be suppressed is the warning that the Major Type 2, Minor Type 16 (IOP I/O Error) entry will be discontinued.
1022	EXPERIMENTAL OUTPUT TO PRINTER	Forces diagnostic information to be written to a printer file.
1023	EXPERIMENTAL SET	Causes diagnostic information to be generated. If the SDASUPPORT library is called by a remote caller, the information is written to a remote file, otherwise it is written to a printer file.

LOG_SKIP

The LOG_SKIP procedure skips a number of records in the log file. The procedure is set up as follows:

```
BOOLEAN PROCEDURE LOG_SKIP(MODE);
    VALUE   MODE;
    INTEGER MODE;
```

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	61			BNA Version 1 debug entry
	62			Add X.25 terminal entry
	63			Modify X.25 terminal entry
	64			Delete X.25 terminal entry
	65			Add X.25 application entry
	66			Modify X.25 application entry
	67			Delete X.25 application entry
	68			X.25 MCS phase change entry
	69			X.25 call established entry
	70			X.25 call terminated entry
	71			X.25 call statistics entry
	72			X.25 endpoint moved entry
	73			X.25 report entry
13		S	S	BNA VERSION 2 ENTRY
	0			General BNA Version 2 log entry
	1			Local task support activity log entry
14		S	S	MLS MESSAGE ENTRY
	1			RSVP message entry
	4			INFO message entry
	9			Unit RSVP message entry
	10			Special RSVP message entry
15				VOLUME STATUS ENTRY
	1			Volume online entry
	2			Volume offline entry
	3			Tape volume purged entry
	4			Tape volume expired entry

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	5			Tape volume newfile entry
	6			Directory add entry
	7			Directory change entry
	8			Directory delete entry
	9			Directory destroy entry
	10			Directory overwrite entry
	11			Directory purge entry
16				FILE STATUS ENTRY
	1			File creation and copy entry
	2			File creation entry
	3			File removal entry
	4			File title change entry
	5			File security attribute entry
	6			File copy entry
17				DATA COMM CONFIGURATION ENTRY
	1			Data comm IDC change entry
	2			New data comm install and copy entry
	3			New data comm install entry
18				COMS CONFIGURATION ENTRY
	1			Agenda change entry
	2			Program change entry
	3			Processing-item change entry
	4			Processing-item list change entry
	5			Library change entry
	6			Database change entry
	7			Station change entry

Legend
S SUMLOG all
J Job log all

continued

The converted SUMLOG files can be analyzed by programs that were designed to analyze Mark 3.6 SUMLOG files.

```
BEGIN
FILE    LOG(DEPENDENTSPECS=TRUE,UPDATEFILE=TRUE); %TO BE FILE EQUATED
ARRAY   BUFFER[Ø:29];
INTEGER W;
DEFINE  LOGLINKX                    = Ø #,
            LOGBLOCKF               = [47: 8] #,
        LOGTYPEX                    = 3 #,
            LOGRESULT_VISIBILITYF   = [31: 4] #,
            LOGMAJORF               = [27:12] #;

READ(LOG,3Ø,BUFFER);
IF BUFFER[LOGLINKX].LOGBLOCKF NEQ  1 OR
   BUFFER[LOGTYPEX].LOGMAJORF NEQ 1Ø THEN
    BEGIN
    REPLACE POINTER(BUFFER) BY "FILE IS NOT A SUMLOG FILE",Ø;
    DISPLAY(BUFFER);
    MYSELF.STATUS := -1;
    END;

WHILE NOT READ(LOG,3Ø,BUFFER) DO
    BEGIN
    IF BUFFER[LOGLINKX].LOGBLOCKF EQL 1 THEN
        IF BUFFER[LOGTYPEX].LOGRESULT_VISIBILITYF NEQ Ø THEN
            BEGIN
            BUFFER[LOGTYPEX].LOGRESULT_VISIBILITYF := Ø;
            WRITE(LOG,3Ø,BUFFER);
            W :=* +1;
            END;
    END;
REPLACE POINTER(BUFFER[Ø]) BY W FOR * DIGITS," UPDATES PERFORMED",Ø;
DISPLAY(BUFFER);
END.
```

Log Entry Types

A number of major classes of log entries (identified by major type numbers) have been established; each major class is divided into subclasses that are identified by minor type numbers. The log entry classes are listed in Table 12–5, "Log Entry Classes," and a detailed description of the format of each type is presented later in this section. The log entry formats in this section are presented in order according to their major and minor type numbers.

The *Default Log Option Setting* column in Table 12–5 indicates which log entry types are logged by default. The LOGGING (Logging Options) system command can be used to select the log entry types to be logged. For further information about this feature, refer to "Selecting the Entry Types to be Logged" in this section.

The *MINIMAL Log Option Setting* column specifies the log setting when the LOGGING MINIMAL system command is issued.

Major Types 8 and 9 were used in previous releases but are no longer used. These major types are listed in Table 12–5 to show that major types with these numbers formerly existed. However, the formats for log entries with these major types are no longer listed in the section.

Major and Minor Type = 0 are conditions reserved to flag null records.

Table 12–5. Log Entry Classes

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
0				IDENTIFICATION ENTRY
	1	S+J	S+J	Establish identity entry
1				JOB ENTRY
	1	S+J	S+J	Beginning-of-job (BOJ) entry
	2	S+J	S+J	End-of-job (EOJ) entry
	3	S+J	S+J	Beginning-of-task (BOT) entry
	4	S+J	S+J	End-of-task (EOT) entry
	5	S		File-open entry
	6	S		File-close entry
	7	S+J	S+J	Controller-rejected job entry
	8	J	J	Aborted-history entry
	9			Usercode validation entry

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	10	S		File interval entry
	11	S		Print request entry
	12	S		Print request complete entry
	13			Start printing file entry
	14			Finish printing file entry
	15			Library link entry
	16			Library delink entry
	17			Library freeze entry
	18			Library resume entry
	19			Database open entry
	20			Database close entry
	21			Database freeze entry
	22			Database resume entry
	25	S		File statistics record entry
2		S	S	MAINTENANCE ENTRY
	11			Library maintenance error entry
	12			Mainframe error entry
	14			Disk file optimizer error entry
	16			I/O error entry (EMS/HDU/RMM)
	17			Hardware configuration entry
	18			Software configuration entry
	19			Environment configuration entry (Extended memory systems only)
	20			Internal processor errors entry (A 1, A 2, A 3, A 4, A 5, A 9, A 10 only)
	21			I/O exception entry (MLIOEXCEPTION)

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	22			Unit I/O counts entry
	23			Mainframe hardware report entry
3		J	J	STRING ENTRY
	1			RSVP message entry
	2			FATAL error message entry
	3			NONFATAL error message entry
	4			SYSTEM message entry
	5			UNIT error message entry
	6			DIRECTORY error message entry
	7			DISPLAY message entry
	8			STATUS message entry
	9			UNIT RSVP message entry
4				MCS ENTRY
	1	S+J	S+J	Session log-on entry
	2	S+J	S+J	Session log-off entry
	3	S+J		RJE control card entry
	4	S+J	J	MCS message entry
	5	S+J	J	MCS time accrued entry
	6	S	S+J	MCS security violation entry
	7	S		MCS station application entry
	8			Remote window open/close entry
	9			MCS window open/close entry
	10	S		Direct window open/close entry
	11			MCS command entry
5		S	S	NSP MAINTENANCE ENTRY
	1			NSP initialization entry
	2			MCS initialization entry

Legend

S SUMLOG all

J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	4			MCS result message entry
	6			UIO-DC unsuccessful I/O entry
	7			UIO-DC abnormal termination entry
6				MISCELLANEOUS ENTRY
	1	S+J	S	Halt/load entry
	2	S+J	S+J	Log release entry
	3	S+J	S+J	SETSTATUS entry
	4	S	S	Security violation entry
	5	S	J	Deimplementation warning entry
	6	S		Power off entry
	7	S		Controller command entry
	8	S		Print subsystem command entry
	9	S		USERDATA change entry
	10			New USERDATA install and copy entry
	11	S		New USERDATA install entry
	12	S		Primitive command entry
	13			Idle system timer entry
	14			Halt load reason entry
	15	S		Disk resource control errors entry
	16	S		Security-relevant warning entry
7		S+J	S+J	INSTALLATION ENTRY
8				(Not currently used)
9				(Not currently used)
10		S	S	DATE/TIME RESET
	1			Log created by A 2 entry
	2			Log created by A 6 entry

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	5			Log created by A 3 entry
	6			Log created by A 9 entry
	7			Log created by A 15 entry
	9			Log created by A 10 entry
	10			Log created by A 5 entry
	15			Log created by A 12 entry
	16			Log created by A 1 entry
	17			Log created by A 17 entry
	19			Log created by A 4 entry
	20			Log created by Micro A entry
11		S	S	BNA Version 1 ENTRY
	1			Set network services manager (NSM) attribute entry
	2			Set port level manager (PLM) attribute entry
	3			Set router attribute entry
	4			Set station level manager (SLM) attribute entry
	5			Network services manager (NSM) phase change entry
	6			Add host entry
	7			Add node entry
	8			Host status entry
	9			Delete host entry
	10			Delete node entry
	11			Port level manager (PLM) error entry
	12			Port level manager (PLM) log entry

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	13			Resistance factor change entry
	14			Start trace entry
	15			Routing refresh entry
	16			Router control frame sent entry
	17			Router control frame received entry
	18			Router frame error entry
	19			Destination node status change entry
	20			Router monitor frame copy entry
	21			Router monitor traffic summary entry
	22			Add profile entry
	23			Delete profile entry
	24			Router node existence entry
	25			Modify station entry
	26			Delete station entry
	27			Add ensemble entry
	28			Modify ensemble entry
	29			Delete ensemble entry
	30			Add connection entry
	31			Modify connection entry
	32			Delete connection entry
	33			Clear call entry
	34			Establish call entry
	35			Await call entry
	36			Send test entry
	37			Test received entry

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	38			Test response received entry
	39			Link reset entry (Not currently used)
	40			Open connection port entry
	41			Close connection port entry
	42			Station attach entry
	43			Station detach entry
	44			Save station entry
	45			Ready station entry
	46			Open station dialog entry
	47			Close station dialog entry
	48			Station attach entry
	49			Station detach entry
	50			Neighbor restart entry (Not currently used)
	51			Neighbor remote busy entry (Not currently used)
	52			Station log report entry
	53			Station validation failure entry
	54			Station monitor entry
	55			Add station entry
	56			Operator message entry
	57			Operator initiated assign entry (Not currently used)
	58			SCM return entry (Not currently used)
	59			Target initiated assign entry (Not currently used)
	60			SCM frame received entry (Not currently used)

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	61			BNA Version 1 debug entry
	62			Add X.25 terminal entry
	63			Modify X.25 terminal entry
	64			Delete X.25 terminal entry
	65			Add X.25 application entry
	66			Modify X.25 application entry
	67			Delete X.25 application entry
	68			X.25 MCS phase change entry
	69			X.25 call established entry
	70			X.25 call terminated entry
	71			X.25 call statistics entry
	72			X.25 endpoint moved entry
	73			X.25 report entry
12		S		VSID ENTRY
	1			Subsystem initiation entry
	2			Subsystem termination entry
	3			Unit online entry
	4			Unit offline entry
	5			Unit ready entry
	6			Unit not-ready entry
	7			Unit in-use entry
	8			Unit not-in-use entry
	9			Begin printing a copy entry
	10			End of printing a copy entry
	11			Special log information entry
	12			Peripheral interface error entry
13		S	S	BNA VERSION 2 ENTRY

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	0			General BNA Version 2 log entry
	1			Local task support activity log entry
14		S	S	MLS MESSAGE ENTRY
	1			RSVP message entry
	4			INFO message entry
	9			Unit RSVP message entry
	10			Special RSVP message entry
15				VOLUME STATUS ENTRY
	1			Volume online entry
	2			Volume offline entry
	3			Tape volume purged entry
	4			Tape volume expired entry
	5			Tape volume newfile entry
	6			Directory add entry
	7			Directory change entry
	8			Directory delete entry
	9			Directory destroy entry
	10			Directory overwrite entry
	11			Directory purge entry
16				FILE STATUS ENTRY
	1			File creation and copy entry
	2			File creation entry
	3			File removal entry
	4			File title change entry
	5			File security attribute entry
	6			File copy entry

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. **Log Entry Classes** (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	8			Station list change entry
	9			Window change entry
	10			Window list change entry
	11			Usercode change entry
	12			Load file and copy entry
	13			Load file entry
19				DIAGNOSTICS ENTRY
	1			Distributed systems services (DSSs) entry
	4			Message Handling System (MHS) entry
	5			Open Systems Interconnection (OSI) entry
	6			TCP/IP (TCPIP) entry
	7			Network-independent software (NIS) entry
	11			Host LAN Connection (HLCN) entry
	12			OSI Directory (DIR) entry
	13			Network Management System (NMS) entry
	16			Systems Network Architecture (SNA) entry
25		S	S	HOST LAN CONNECTION ENTRY
	1			Change network processor entry
	2			Network processor report entry
	3			Add NETACCESSPOINT entry
	4			Change NETACCESSPOINT entry
	5			Delete NETACCESSPOINT entry

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
	6			Reset router entry
	7			Network manager report entry
27				TCP/IP ENTRY
	1			Transmission Control Protocol (TCP) layer entry
	2			Internet Protocol (IP) entry
	3			Address Resolution Protocol (ARP) function entry
	4			Route Information Protocol (RIP) function entry
	5			User Datagram Protocol (UDP) layer entry
	6			Internet Control Message Protocol (ICMP) function entry
	7			TCP Manager (TCPMGR) function entry
	8			Connect open and close reports (PIM) entry
	9			Reserved
	10			Entries generated during the process of an SNMP command.
	11			Miscellaneous (OTHER) entry
28				NETWORK MANAGEMENT ENTRY
	1			Local Control Facility (LCF) entry
	2			Network Control Facility (NCF) entry
	3			SNMP agent (SNMP) entry
	4			Miscellaneous (OTHER) entry

Legend
S SUMLOG all
J Job log all

continued

Table 12–5. Log Entry Classes (cont.)

Major Type	Minor Type	Default Log Option Setting	MINIMAL Log Option Setting	Entry Class
30				SYSTEMS NETWORK ARCHITECTURE (SNA) ENTRY
	1			Noteworthy network event (REPORT) entry
	2			Network frame reported as a result of the TRACE + command
	3			Unexpected fault condition (ERROR) entry
	4			Solicited status request and response (INQUIRY) entry
	5			Solicited action request and response (COMMAND) entry
	6			Miscellaneous (OTHER) entry

Legend
S SUMLOG all
J Job log all

Major Type = 0 Establish Identity Entry (LOGMAJ_IDENTYTIV)

Major Type 0, Minor Type 1 (Establish Identity) entries describe the identity associated with a mix number, when it is not otherwise contained in the SUMLOG or the identity is changed. An identity is made from a usercode, access code, and origination. Typically, these appear in the SUMLOG right after a Transfer Log (TL) has been performed.

Table 12–6. Major Type 0–Minor Type 1

Minor Type = 1 Establish Identity (LOGMIN ESTAB IDV)	
Words	**Description**
0–3	As described in Table 12-2, “First Four Log Entry Words”
4	Privilege status [9: 1] Security administrator [0: 1] Privileged user
5	Type [15: 1] MCS Session [14:15] Process type (if not MCS session) 0 = Dependent task (PROCESS) 1 = Coroutine (CALL) 2 = Independent task (RUN) 3 = Job Stack
6	LINK to external or station name
7	LINK to usercode
8	LINK to charge code
9	LINK to session attribute PPB
10	Source information [23: 8] MCS number (if MCS Session) [15: 1] MCS Session [14:15] LSN number (if MCS Session)

continued

Table 12–6. Major Type 0–Minor Type 1 (cont.)

Minor Type = 1 Establish Identity (LOGMIN ESTAB IDV)	
Words	**Description**
11	LINK to access code
12	Time started (timestamp)
13	LINK to MCS name
14	Stack number
	[47:32] = This field is not used
	[15:16] = Stack number
15–end	Variable-length data pointed to by the LINKs in this entry

Table 12–7. Major Type 0–Minor Type 2

Minor Type = 1 Change Identify (LOGMIN_CHANGE_IDV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Mix number information
	[19:20] New mix number
	[39:20] Old mix number

Major Type = 1 Job or Task Entry (LOGMAJJOB)

Table 12–8. Major Type 1–Minor Types 1 and 3

Minor Type = 1 BOJ Entry (LOGBOJTYP)
Minor Type = 3 BOT Entry (LOGBOTYP)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Task priority information [47:24] Job queue [23:24] Priority
5	Task type [47: 1] Job restarted, either manually or automatically after a halt/load [46: 1] Task fired off by DMS access routines; typically, by a DMS ABORT routine. [45: 1] MCS fired off by Datacom Controller stack [44: 1] Task contains unsafe code [43: 1] Task is subject of TADS session [41: 1] Privileged program, if the result indicator field of Word 3 has a value of 2 (security relevant) [39: 1] Program marked with security administrator privilege [38: 1] Program marked with tasking status by the MP (Mark Program) system command [37: 1] Job manually restarted (bit [47: 1] is also set) [15:16] Process type 0 = Dependent task (PROCESS) 1 = Coroutine (CALL) 2 = Independent task (RUN) 3 = Job stack
6	LINK to external name

continued

Table 12–8. Major Type 1–Minor Types 1 and 3 (cont.)

Minor Type = 1 BOJ Entry (LOGBOJTYP)
Minor Type = 3 BOT Entry (LOGBOTYP)

Words	Description
7	LINK to usercode
8	LINK to charge code
9	LINK to compiled name. Contains name of object code file if task was COMPILE or BIND; otherwise, value is zero
10	Task source and destination information [45: 6] Destination MCS [39: 1] Destination is REMOTE [38:15] Destination UNIT [21: 6] Originating MCS [15: 1] Origin is REMOTE [14:15] Originating UNIT If [15: 1] (above) is SET, the [14:15] field contains a logical station number (LSN). If [15: 1] is RESET, word 19 of this entry contains the device number of the originating unit.
11	LINK to access code
12	Job entry time [47:16] Contains the date [31:32] Holds the entry time The job entry time is the timestamp of the code file. Thus, for a job, it is the time the WFL compiler finished with it and it entered the system. For a task, it is the timestamp of the object code file being run. It is zero if the code stack was being shared with an already running task. The date format is [47:16] (Julian date – 70000). The entry time format is [31:32] (time of day in ticks DIV 16).
13	Compiler information [47: 1] Interprocess communication (IPC) capable [46: 1] Sort capable

continued

Table 12–8. Major Type 1–Minor Types 1 and 3 (cont.)

Minor Type = 1 BOJ Entry (LOGBOJTYP)
Minor Type = 3 BOT Entry (LOGBOTYP)

Words	Description
	[45: 1] Control program
	[44: 1] DMS capable
	[43: 1] No uplevel pointers
	[42: 1] Privileged program
	[41: 1] Library capable
	[40: 1] WFL: No global file equation
	[39: 1] Privileged transparent program
	[38: 1] Suppressed
	[37: 1] Resident program
	[31: 8] Language type
	0 = ALGOL
	1 = COBOL
	2 = FORTRAN
	4 = PL/I
	5 = JOVIAL
	6 = NEWP
	7 = ESPOL
	8 = DCALGOL
	9 = BASIC
	10 = WFL
	14 = PASCAL
	15 = RPG
	16 = FORTRAN77
	17 = COBOL74
	18 = SORT
	254 = INTRINSICS ($INTRINSICS set)

continued

Table 12–8. Major Type 1–Minor Types 1 and 3 (cont.)

Minor Type = 1 BOJ Entry (LOGBOJTYP)
Minor Type = 3 BOT Entry (LOGBOTYP)

Words	Description
	255 = MCP ($MCP set)
	[23: 8] Compiler mark number and level number
	[15: 1] Program has tasking privilege.
	[14: 1] Program has security administrator privilege.
	[13: 1] Program has security administrator transparent privilege.
	[12: 1] Program has tasking transparent privilege.
	[9:10] Compiler cycle number
14	Prior to the Mark 3.9 system software release, this word contained the following SWAPPER information:
	[39:10] Number of swap space core slots in use
	[15: 8] SUBSPACE attribute requested
	0 = Do not run in swap space
	1 = Data (D2) stack only in swap space
	2 = Data in swap space. Code (D1) stack in swap space if code file is in my directory.
	3 = Data and code in swap space
	[7: 8] SUBSPACE actually granted (Same as [15: 8], but 2 is not valid.)
15	Prior to the Mark 3.9 system software release, this word contained the numbers of ASNs used by the process.
	[39:20] ASN of code (D1) stack
	[19:20] ASN of data (D2) stack
16	Prior to the Mark 3.9 system software release, this word contained a LINK to the value of the SUBSYSTEM task attribute.
17	LINK to the value of the ITINERARY chain. The LENGTH field is in characters rather than words.

continued

Table 12–8. Major Type 1–Minor Types 1 and 3 (cont.)

Minor Type = 1 BOJ Entry (LOGBOJTYP) **Minor Type = 3 BOT Entry (LOGBOTYP)**	
Words	**Description**
18	Amount of time spent in the schedule
19	Contains the external device number of the originating unit (if not a remote station; see Word 10, field [14:15] of this entry).
20-21	(Reserved for future use)
22	LINK to the value of the SOURCENAME task attribute
23	LINK to the value of the MCSNAME task attribute
24	Stack number
	[47:32] = This field is not used
	[15:16] = Stack number
25–end	Variable-length data pointed to by the LINKs in this entry.

Table 12–9. Major Type 1–Minor Types 2 and 4

Minor Type = 2 EOJ Entry (LOGEOJ) **Minor Type = 4 EOT Entry (LOGEOTYP)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Task priority information
	[47:24] Job queue
	[23:24] Priority
5	Task type
	[47: 1] Job restarted
	[46: 1] Task fired off by DMS access routines; typically, by a DMS ABORT routine

continued

Table 12–9. Major Type 1–Minor Types 2 and 4 (cont.)

Minor Type = 2 EOJ Entry (LOGEOJ)
Minor Type = 4 EOT Entry (LOGEOTYP)

Words	Description
	[45: 1] MCS fired off by Datacom Controller stack
	[15:16] Process type
	0 = Dependent task (PROCESS)
	1 = Coroutine (CALL)
	2 = Independent task (RUN)
	3 = Job stack
6	LINK to external name
7	LINK to usercode
8	LINK to charge code
9	LINK to compiled name. Contains name of object code file if task was COMPILE or BIND, otherwise value is zero.
10	Task source and destination information
	[45: 6] Destination MCS
	[39: 1] Destination is REMOTE
	[38:15] Destination UNIT
	[21: 6] Originating MCS
	[15: 1] Origin is REMOTE
	[14:15] Originating UNIT
	If [15: 1] (above) is SET, the [14:15] field of this word contains a logical station number (LSN). If [15: 1] is RESET, Word 35 of this entry contains the device number of the originating unit.
11	Processor time†
12	I/O time†
13	Number of lines printed†
14	Number of cards read†

† Includes resource usage of any offspring using anonymous task accounting

continued

Table 12-9. Major Type 1-Minor Types 2 and 4 (cont.)

Minor Type = 2 EOJ Entry (LOGEOJ)
Minor Type = 4 EOT Entry (LOGEOTYP)

Words	Description
15	Number of cards punched†
16	Memory integral code, D1 stack
17	Memory integral data, D2 stack
18	Termination condition. Contains the value of the HISTORY task attribute at the time of termination. For a description of the format of this word, refer to the *A Series Task Attributes Programming Reference Manual.*
19	Prior to the Mark 3.9 system software release, this word contained the following SWAPPER information [39:10] Number of swap space core slots in use [15: 8] SUBSPACE attribute requested 0 = Do not run in swap space. 1 = Data (D2) stack only in swap space. 2 = Data in swap space. Code (D1) stack in swap space if code file is in my directory. 3 = Data and code in swap space. [7: 8] SUBSPACE actually granted (Same as [15: 8], but 2 is not valid.)
21	Average code space
22	Average data space
23	Elapsed time
24	LINK to access code
25	Compiler information [47: 1] IPC capable [46: 1] Sort capable [45: 1] Control program [44: 1] DMS capable

† Includes resource usage of any offspring using anonymous task accounting

continued

Table 12-9. Major Type 1–Minor Types 2 and 4 (cont.)

Minor Type = 2 EOJ Entry (LOGEOJ) **Minor Type = 4 EOT Entry (LOGEOTYP)**	
Words	**Description**
	[43: 1] No uplevel pointers
	[42: 1] Privileged program
	[41: 1] Library capable
	[40: 1] WFL: No global file equation
	[39: 1] Privileged transparent
	[38: 1] Suppressed
	[37: 1] Resident program
	[31: 8] Language type
	0 = ALGOL
	1 = COBOL
	2 = FORTRAN
	4 = PL/I
	5 = JOVIAL
	6 = NEWP
	7 = ESPOL
	8 = DCALGOL
	9 = BASIC
	10 = WFL
	14 = PASCAL
	15 = RPG
	16 = FORTRAN77
	17 = COBOL74
	18 = SORT
	254 = INTRINSICS ($INTRINSICS set)
	255 = MCP ($MCP set)
	[23: 8] Compiler mark number and level number

† Includes resource usage of any offspring using anonymous task accounting

continued

Table 12–9. Major Type 1–Minor Types 2 and 4 (cont.)

Minor Type = 2 EOJ Entry (LOGEOJ)
Minor Type = 4 EOT Entry (LOGEOTYP)

Words	Description
	[9:10] Compiler cycle number
26	Prior to the Mark 3.9 system software release, this word contained the core allocated and occupied. ASN I is valid if bit I in the appropriate field is set. [31: 8] Data core allowed (= Word 36) [23: 8] Code core allowed (= Word 37) [15: 8] Data core occupied (= Word 38) [7: 8] Code core occupied (= Word 39) ***Note:*** *For extended memory systems containing more than seven local ASNs, the values of Words 36 through 39 should be used.*
27	LINK to the value of the SUBSYSTEM attribute
28	LINK to the value of the ITINERARY chain The LENGTH field is in characters rather than words.
29	Amount of time spent in the ready queue
30	Amount of processor time spent handling initial P-bits for this task†
31	Number of initial P-bits for this task†
32	Amount of processor time spent handling all other P-bits for this task†
33	Number of other P-bits for this task†
34	Time at which the task was initiated [47:16] Date (Julian date – 70000) [31:32] Entry time (time of day in ticks DIV 16)
35	Contains the external device number of the originating unit (if not a remote station; see word 10, field [14:15] of this entry).
36	Data core allocated
37	Code core allocated

† Includes resource usage of any offspring using anonymous task accounting

continued

Table 12-9. Major Type 1–Minor Types 2 and 4 (cont.)

Minor Type = 2 EOJ Entry (LOGEOJ)
Minor Type = 4 EOT Entry (LOGEOTYP)

Words	Description
38	Core occupied
39	Code core occupied
40	Maximum number of ASDs used at any one time by the job/task. In logs generated prior to the Mark 3.9 release on ASN systems, this word is zero.
41	Integer count of tape volume assignments logged for the process.† This word contains the number of tape-file open operations (except those that follow a close operation with the RETAIN option), reel switches, and references to tape volumes in library maintenance COPY and ADD statements. All printer backup tape file opens are included in this count.
42	Integer count of requests for offline disk volumes that the operator readied or otherwise handled so that the job could proceed.†
43	Real number that measures the amount of temporary disk space the task or job used.† The number logged is the product of the average number of disk segments used for temporary disk storage times the number of seconds the task or job ran (elapsed time).
44	Real number that measures the amount of permanent disk space the task or job used.† The number logged is the product of the average number of disk segments occupied by permanent disk files that the task had open times the number of seconds the job or task ran (elapsed time).
45	Maximum save memory used at any one time by the job/task. In logs generated prior to the Mark 3.9 release on ASN systems, this word is zero.
46	Stack number
	[47:32] = This field is not used
	[15:16] = Stack number
47	Variable-length data pointed to by the LINKs in this entry.

† Includes resource usage of any offspring using anonymous task accounting

If anonymous task accounting is in effect for a task, no EOT log entry is logged for that task. Certain resource usage statistics of the task are added to the resource usage statistics of the parent. These statistics are reflected in words 11 through 15, 30 through 33, and 41 through 44 of the parent EOJ or EOT log entry. For further information about anonymous task accounting, refer to "Using Anonymous Task and File Accounting" earlier in this section.

Table 12–10. Major Type 1–Minor Type 5

Minor Type = 5 File Open Entry (LOGOPENTYPE)	
Words	**Description**
0–3	As described in Table 12–2, “First Four Log Entry Words”
4	File size in units of words (valid for disk and pack only)
5	LINK to internal name
6	LINK to external name
7	Open description word [47: 2] File Access Rule used 1 = Actor rights 2 = Declarer rights 3 = Union of Actor and Declarer rights [45: 1] Value of DUMMYFILE attribute [44: 1] An unused field [43: 4] The high-order four bits of the open error message number. The low-order bits are stored in field [15: 8]. The open error message number identifies the system message that explains the open error. When an open error occurs, the system issues a Major Type 14, Minor Type 4 (INFO Message) log entry that contains the text of this system message. When such a log entry is printed by LOGANALYZER or appears in a job summary, the message text is preceded by the string *MSRFIB* and the open error message number. For example, open error message number 74 refers to system message MSRFIB74, “NOT CLOSED”. For processes discontinued by open errors, the open error message number also corresponds to the HISTORYREASON task attribute value. If the HISTORYTYPE value is 4 (DSEDV) and the HISTORYCAUSE value is 8 (SOFTIOERRCAUSEV), then the open error message number equals the HISTORYREASON value. If the HISTORYTYPE value is 4 (DSEDV) and the HISTORYCAUSE value is 14 (SOFTIOERR2CAUSEV), then the open error message is equal to the HISTORYREASON value plus 256. For descriptions of the HISTORYTYPE, HISTORYCAUSE, and HISTORYREASON task attributes, refer to the *A Series Task Attributes Programming Reference Manual*. [31: 8] Logical type (See Table 12–22, “Peripheral Logical Types.”) [23: 8] Value of the FILEKIND file attribute

continued

Table 12–10. Major Type 1–Minor Type 5 (cont.)

Minor Type = 5 File Open Entry (LOGOPENTYPE)	
Words	**Description**
	[15: 8] The low-order eight bits of the open error message number. For further information, refer to the description of Field [43: 4]. [7: 7] Value of the MYUSE file attribute: 1 = INPUT 2 = OUTPUT 3 = I/O [0: 1] If this bit is 1, then [15: 8] contains an error code.
8	LINK to access code
9	OPEN parameters [46: 1] 1 if file is a valid subfile [41:10] Subfile index, if file is a valid subfile [15: 4] Motion 1 = Positioning requested 2 = No positioning requested [11: 4] Position 1 = AT FRONT 2 = AT END [7: 8] Open type For nonport files: 1 = OPEN WAIT 2 = AVAILABLE 3 = OFFER 4 = RESIDENT 5 = PRESENT 6 = PBT REEL SWITCH OPEN 7 = STACK ASSIGNED OPEN REVERSE For port files:

continued

Table 12–10. Major Type 1–Minor Type 5 (cont.)

Minor Type = 5 File Open Entry (LOGOPENTYPE)	
Words	**Description**
	1 = WAIT 2 = AVAILABLE 3 = RETURN (OFFER) 4 = DONTWAIT
10	Device number
11	Stack using the file (actor) [47:12] Stack number [31:16] Job number [15:16] Task number
12	Stack owning the file (declarer; same word format as Word 11)
13	LINK to port file information. This LINK is 0 if this entry is not for a port file. If the LINK is nonzero, it points to an area used to contain a block of additional information for port files. The information contained in that block is listed in the following subtable. The location of each item is described by an offset from the start of this block of information. The LINKs within this block of information point to strings of characters. The length of each string is given in characters (not words) in the length field [39:20] of the LINK. The index field [19:20] of the LINK provides the word index in the log entry where the information starts. **Offset** — **Contents** 0 — LINK to YOURHOSTNAME value 1 — LINK to YOURNAME value 2 — Value of MAXSEGSIZE attribute 3 — LINK to YOURUSERCODE value 4 — Value of MYPORTADDRESS attribute 5 — Value of MYSUBPORTADDRESS attribute 6 — LINK to SERVICE value 7 — LINK to PROVIDERGROUP value 8 — LINK to the name of the provider selected, if the OPEN succeeded; otherwise, 0 (zero)

continued

Table 12–10. Major Type 1–Minor Type 5 (cont.)

Minor Type = 5 File Open Entry (LOGOPENTYPE)		
Words	**Description**	
	9	CONNECTTIMELIMIT value, if one was specified; otherwise, 0 (zero)
	10	Initiation primitive invoked
		0 = OPEN
		1 = AWAITOPEN
	11	RESPOND invocation
		1 = RESPOND(ACCEPTCLOSE)
		2 = RESPOND(REJECTCLOSE)
		31 = RESPOND was not invoked
	18	ASSOCIATED DATA transmission
		0 = ASSOCIATED DATA was sent
		1 = ASSOCIATED DATA was not sent
	19	ASSOCIATED DATA receipt
		0 = ASSOCIATED DATA was received
		1 = ASSOCIATED DATA was not received
14	LINK to family name in substandard form.	
15–end	Variable-length data pointed to by the LINKs in this entry.	

See Tables 12–22 through 12–25 for information on peripheral types.

Table 12–11. Major Type 1–Minor Type 6

Minor Type = 6 File Close Entry (LOGCLOSETYPE)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	File size in units of words (valid for disk and pack only)
5	LINK to internal name (standard form)
6	LINK to external name (standard form)
7	Close description word
	[47: 2] File Access Rule used
	1 = Actor rights
	2 = Declarer rights
	3 = Union of actor and declarer rights
	[45: 1] Value of DUMMYFILE file attribute
	[43: 4] The high-order four bits of the close error message number. The low-order bits are stored in field [15: 8].
	The close error message number identifies the system message that explains the close error. When a close error occurs, the system issues a Major Type 14, Minor Type 4 (INFO Message) log entry that contains the text of this system message. When such a log entry is printed by LOGANALYZER or appears in a job summary, the message text is preceded by the string *MSRFIB* and the close error message number. For example, close error message number 19 refers to system message MSRFIB19, "FAILED ENTER".
	For processes discontinued by close errors, the close error message number also corresponds to the HISTORYREASON task attribute value. If the HISTORYTYPE value is 4 (DSEDV) and the HISTORYCAUSE value is 8 (SOFTIOERRCAUSEV), then the close error message number equals the HISTORYREASON value. If the HISTORYTYPE value is 4 (DSEDV) and the HISTORYCAUSE value is 14 (SOFTIOERR2CAUSEV), then the close error message is equal to the HISTORYREASON value plus 256. For descriptions of the HISTORYTYPE, HISTORYCAUSE, and HISTORYREASON task attributes, refer to the *A Series Task Attributes Programming Reference Manual.*
	[31: 8] Logical type (See Table 12–22, "Peripheral Logical Types.")
	[23: 8] Value of FILEKIND file attribute

continued

Table 12–11. Major Type 1–Minor Type 6 (cont.)

Minor Type = 6 File Close Entry (LOGCLOSETYPE)	
Words	**Description**
	[15: 8] The low-order eight bits of the close error message number. For further information, refer to the description of Field [43: 4].
	[7: 7] Value of MYUSE file attribute 1 = INPUT 2 = OUTPUT 3 = I/O
	[0: 1] If this bit is a 1, then field [15: 8] contains an error code.
8	Name qualification information [42:15] USASI generation number [27:11] SAVEFACTOR [16:17] Creation date in binary Julian format
9	File blocking information [47:16] BLOCKSIZE [27:11] MINRECSIZE [15:16] MAXRECSIZE
10	Transaction count (number of logical I/O operations)
11	Elapsed I/O time
12	Miscellaneous information For a disk file [47: 1] Installation-allocated disk (IAD) file [46: 1] Permanent file [44: 1] Updated file [43: 1] CLOSE with CRUNCH [42: 1] Protected file [40: 1] Interchange disk pack file

continued

Table 12–11. Major Type 1–Minor Type 6 (cont.)

Minor Type = 6 File Close Entry (LOGCLOSETYPE)	
Words	**Description**
	[32: 1] Duplicated file
	[31: 4] Number of duplicate copies
	[23:24] Value of AREASIZE attribute
	For a tape file
	[15: 1] Contains 1 if the file was a multireel file
	[14:15] Reel number
	For units other than tape or disk, this word contains zeros
13	Serial number (six EBCDIC characters) for tape file. This word is 0 for other units.
14	Unit information
	[47:16] INTMODE
	[25:14] CYCLE
	[11: 8] VERSION
	[0: 1] UNITS
15	Close parameters
	[46: 1] Valid subfile
	[39:16] Subfile index, if file is a valid subfile
	[23: 8] Disposition
	1 = REWIND
	2 = NOREWIND
	3 = SAVE
	4 = LOCK
	5 = PURGE
	6 = CRUNCH
	7 = HERE
	8 = BLOCKEXIT
	[15: 8] Association

continued

Table 12–11. Major Type 1–Minor Type 6 (cont.)

Minor Type = 6 File Close Entry (LOGCLOSETYPE)	
Words	**Description**
	1 = RELEASE 2 = RETAIN 3 = RESERVE 4 = DISABLE [7: 8] Close type For nonport files: 1 = REGULAR CLOSE 2 = REEL CLOSE 3 = DONT WAIT For port files (note that disposition and association fields are not used): 0 = BLOCKEXIT 1 = NORMAL 2 = DONTWAIT
16	Physical I/O count [47:24] Read count [23:24] Write count
17	LINK to FORMMESSAGE (standard form)
18	LINK to port file information. This LINK is 0 if this entry is not for a port file. If the LINK is nonzero, it points to an area used to contain a block of additional information for port files. The information contained in that block is listed in the following subtable. The location of each item is described by an offset from the start of this block of information. The LINKs within this block of information point to strings of characters. The length of each string is given in characters (not words) in the length field [39:20] of the LINK. The index field [19:20] of the LINK provides the word index in the log entry where the information starts. **Offset** — **Contents** 0 — LINK to YOURHOSTNAME value 1 — LINK to YOURNAME value

continued

Table 12–11. Major Type 1–Minor Type 6 (cont.)

Minor Type = 6 File Close Entry (LOGCLOSETYPE)	
Words	**Description**
2	Value of MAXSEGSIZE attribute
3	LINK to YOURUSERCODE value
4	Number of messages sent
5	Number of messages received
6	Number of segments sent
7	Number of segments received
8	Number of retransmissions
9	Number of control frames sent
10	Value of MYPORTADDRESS attribute
11	Value of MYSUBPORTADDRESS attribute
12	LINK to SERVICE value
13	LINK to PROVIDERGROUP value
14	LINK to provider selected, if the CLOSE succeeded; otherwise, 0 (zero)
15	CLOSE type 0 = ABORT 1 = REQUEST
16	INITIATOR 0 = Locally initiated 1 = Correspondent initiated 2 = Provider initiated
17	RESPOND invocation 1 = RESPOND(ACCEPTCLOSE) 2 = RESPOND(REJECTCLOSE) 31 = RESPOND was not invoked
18	ASSOCIATED DATA transmission 0 = ASSOCIATED DATA was sent 1 = ASSOCIATED DATA was not sent

continued

Table 12–11. Major Type 1–Minor Type 6 (cont.)

Minor Type = 6 File Close Entry (LOGCLOSETYPE)	
Words	**Description**
	19 ASSOCIATED DATA receipt 0 = ASSOCIATED DATA was received 1 = ASSOCIATED DATA was not received
19	External device number
20	Stack using the file (actor)
	[47:12] Stack number
	[31:16] Job number
	[15:16] Task number
21	Stack owning the file (declarer; same field format as Word 20)
22	LINK to family name in substandard form
23	Number of physical read operations initiated for the file
24	Number of physical write operations initiated for the file
25	FILESTRUCTURE for disk files. This word is -1 for other kinds of files.
	0 = ALIGNED180
	1 = STREAM
	5 = BLOCKED
26	Buffer size in words
27	Buffer size source
	1 = The buffer size for the file was based on the BLOCKSIZE file attribute because the BUFFERSIZE attribute is not applicable to this file.
	2 = The buffer size for the file was specified through an assignment to the BUFFERSIZE file attribute.
	3 = The buffer size for the file was based on the BUFFERGOAL memory management parameter of the SF (Set Factors) system command.

continued

Table 12–11. Major Type 1–Minor Type 6 (cont.)

Minor Type = 6 File Close Entry (LOGCLOSETYPE)	
Words	**Description**
28	Density for tape files. This word is –1 for other units.
	0 = BPI800
	1 = BPI556
	2 = BPI200
	3 = BPI1600
	4 = BPI6250
	5 = BPI38000
	6 = BPI1250
29	Physical Data Representation
	[47:16] EXTMODE
	[31:16] CCSVERSION
	[15:01] CCSVERSION validity bit (1 if CCSVERSION contains a valid value)
30–end	Variable-length data pointed to by the LINKs in this entry

See Tables 12–22 through 12–25 for information on peripheral types.

Table 12–12. Major Type 1–Minor Type 7

Minor Type = 7 Job Rejected Entry (LOGABORTTYPE)	
Words	**Description**
0–3	As described in Table 12–2, “First Four Log Entry Words”
4	The reason the job was rejected.
	[31: 8] Reason for controller reject:
	1 = Discontinued out of a job queue

continued

Table 12–12. Major Type 1–Minor Type 7 (cont.)

Minor Type = 7 Job Rejected Entry (LOGABORTTYPE)	
Words	**Description**
	2 = Discontinued by attribute error on job start [23:16] Error number If [31: 8] = 1, then 0 = Insufficient space to store the job file header in the job description (JOBDESC) file 1 = Invalid priority specification 2 = Invalid processor limit specification 3 = Invalid I/O limit specification 4 = Invalid card limit specification 5 = Invalid lines limit specification 6 = Invalid wait limit specification 7 = Invalid elapsed limit specification 8 = Invalid disk limit specification 11 = Invalid tape limit specification 12 = SAVECORE job attribute value higher than job queue limit 99 = Job had no CLASS attribute, and could not be assigned to any job queue due to conflicts between job attributes and job queue attributes 100 = Invalid family specification 101 = Invalid queue specification 103 = Discontinued while in the queue 104 = WFL syntax error 105 = Insufficient disk space on the JOB family to allocate stack-rollout and job summary areas for the job If [31: 8] = 2, then the attribute error number

Table 12–13. Major Type 1–Minor Type 8

Minor Type = 8 Job/Task Abort History (LOGABORTHISTORY)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Length of STACKHISTORY task attribute value in bytes
5–end	STACKHISTORY task attribute value. For information about the format of this value, refer to the *A Series Task Attribute Programming Reference Manual.*

The system logs Usercode Validation log entries when a usercode is validated through USERDATA function 3.

Table 12–14. Major Type 1–Minor Type 9

Minor Type = 9 Usercode Validation (LOGVALIDATE)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	If nonzero, the mix number of the process for which the validated usercode was assigned. If zero, then no process was supplied or the process was not running.
5	LINK to old usercode.
6	LINK to validated usercode.
7–end	Variable-length data pointed to by the LINKS in this entry.

The format and contents of the File Interval log entry are almost identical to those of the Major Type 1, Minor Type 6 (File Close) entry. The following table indicates the exceptions.

Table 12–15. Major Type 1–Minor Type 10

Minor Type = 10 File Interval Entry (LOGINTERVALTYPE)	
Words	**Description**
3	Field [15:16] Minor Type = 10
7	Field [15: 8] is undefined.
7	Field [0: 1] is undefined.
15	Field [23: 8] is undefined.
15	Field [15: 8] is undefined.
15	Field [7: 8] is undefined.
18	Offset 15 (CLOSE type) is undefined.

For more information, see Table 12–11, "Major Type 1–Minor Type 6."

Table 12–16. Major Type 1–Minor Types 11 through 14

Minor Type = 11 Print Request (LOGPR_REQ_BEGIN) Minor Type = 12 Print Request Complete (LOGPR_REQ_END) Minor Type = 13 Start Printing File (LOGPR_FILE_START) Minor Type = 14 Finish Printing File (LOGPR_FILE_END)	
Words	**Description**
0–3	As described in Table 12-2, "First Four Log Entry Words"
4	Request information
	[47: 1] Restart from a checkpoint (start printing file only)
	[47: 8] Result of print (finish printing file only)
	0 = No error
	1 = Print discontinued by operator (REQUEUE,QT,DS)

continued

Table 12–16. Major Type 1–Minor Types 11 through 14 (cont.)

Minor Type = 11 Print Request (LOGPR_REQ_BEGIN)
Minor Type = 12 Print Request Complete (LOGPR_REQ_END)
Minor Type = 13 Start Printing File (LOGPR_FILE_START)
Minor Type = 14 Finish Printing File (LOGPR_FILE_END)

Words	Description
	2 = Print repositioned by operator (SKIP)
	3 = Print discontinued by fault (transform error)
	4 = Print discontinued by device (disk/printer error)
	[39: 1] Removed after print (finish printing file only)
	0 = File was not removed
	1 = File was removed
	[27:28] Print request number
5	LINK to usercode of request originator
6	LINK to origin device name (substandard form)
7	LINK to PRINTCHARGE attribute value
8	Originating job type
	1 = Session
	2 = WFL job
	3 = Jobsummary only
	4 = MCP function
	5 = Independent program
9	LINK to job name
10	LINK to FORMID attribute value
11	AFTER attribute value
12	Original or actual request size in lines
13	[15:16] TRAINID attribute value
14	LINK to PRINTERCONTROL attribute value
15	LINK to NOTE attribute value

continued

Table 12–16. Major Type 1–Minor Types 11 through 14 (cont.)

Minor Type = 11 Print Request (LOGPR_REQ_BEGIN)
Minor Type = 12 Print Request Complete (LOGPR_REQ_END)
Minor Type = 13 Start Printing File (LOGPR_FILE_START)
Minor Type = 14 Finish Printing File (LOGPR_FILE_END)

Words	Description
16	Origination information [31:16] Job/Session number of originator [15:16] Task number of creator
17	LINK to hostname of originating request, if not originated on the local host
18	LINK to DESTINATION attribute value
19	LINK to backup file name. This word is used only for Minor Types 13 (Start Printing File) and 14 (Finish Printing File).
20	Elapsed printing time in units of 2.4 microseconds. This word is used only for Minor Types 12 (Print Request Complete) and 14 (Finish Printing File).
21	File size in pages, if available. This word is used only for Minor Types 12 (Print Request Complete), 13 (Start Printing File), and 14 (Finish Printing File).
22	LINK to PRINTPARTIAL file attribute value
23	LINK to REQUESTNAME print modifier value
24–end	Variable-length data, pointed to by the LINK words in this entry

Table 12–17. Major Type 1–Minor Types 15 and 16

Minor Type = 15 Library Link (LOGMIN_LIBLINKV)
Minor Type = 16 Library Delink (LOGMIN_LIBDELINKV)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words." Note that the mix number in Word 0 is the mix number of the process linking to or delinking from the library.
4	Linkage/delinkage result. Negative values indicate failure. For a list of the possible values and their meanings, refer to the description of the LINKLIBRARY function in the *A Series ALGOL Programming Reference Manual, Volume 1: Basic Implementation.*
5	[47:12] Stack number of process linking to or delinking from the library
6	LINK to code file name of linking program in standard form
7	LINK to library attributes of linking program
8	Number of libraries linked to or delinked from
9	LINK to library information stream, containing the number of items specified in Word 8 The items are of variable length. Each item contains: Word 0: [47:12] Stack number of library process [15:16] Mix number of library process Word 1: Linkage security class of library. For a description of the possible values, refer to the description of word 4, field [19: 4] in Table 12–18, "Major Type 1–Minor Types 17 and 18." Word 2–end: Code file name of library in standard form
10–end	Variable-length data pointed to by the LINKs in this entry

Table 12–18. Major Type 1–Minor Types 17 and 18

Minor Type = 17 Library Freeze (LOGMIN_LIBFREEZEV) **Minor Type = 18 Library Resume (LOGMIN_LIBTHAWV)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words." Note that the mix number in Word 0 is the mix number of the library process.
4	Library status [47:12] Library stack number [35: 4] Library state 0 = Initiated by user 1 = Initiated by linker 2 = Not allowed to freeze 3 = Usable library 4 = Nonresumable library 5 = Resuming library 6 = Resumed library 7 = Library never froze [30: 3] Sharing specification 0 = Private 1 = Shared by run unit 3 = Shared by address space 4 = Shared by all [27: 1] Permanency 0 = Temporary 1 = Permanent [26: 1] SET if trusted library [25: 2] LIBACCESS attribute value 0 = By title 1 = By function 2 = By initiator

continued

Table 12–18. Major Type 1–Minor Types 17 and 18 (cont.)

Minor Type = 17 Library Freeze (LOGMIN_LIBFREEZEV) **Minor Type = 18 Library Resume (LOGMIN_LIBTHAWV)**	
Words	**Description**
	[19: 4] Linkage security class 0 = Default class. Any user process can link to this library. 1 = MCP class. Only user processes with the MCP linkage class can link to this library. 2 = Message control system (MCS) class. 3 = Environment software class. This class includes processes such as Data Management System II (DMSII), the Test and Debug System (TADS), and KEYEDIO. 4 = Privileged program class. 5 = Compiler class. Only user processes with compiler or MCP linkage class can link to the library. 6–7 = Reserved for system software. 8–15 = Site-dependent linkage classes.
5	Library HISTORY attribute value [23: 8] Reason [15: 8] Cause [7: 8] Type
6	LINK to code file name of library in standard form
7–end	Variable-length data pointed to by the LINKs in this entry

Table 12–19. Major Type 1–Minor Types 19 and 20

Minor Type = 19 Database Open (LOGMIN_DBOPENV)
Minor Type = 20 Database Close (LOGMIN_DBCLOSEV)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Result (negative values indicate failure)
5	LINK to code file name of the database user
6	LINK to usercode of the database user
7	LINK to the access code of the database user
8	LINK to the external name of the database
9	LINK to the internal name of the database
10	Database mix identification
	[47:12] Stack number
	[31:16] Job number
	[15:16] Mixnumber
11	SIB level information at database open
	[47: 8] SIB format level
	[35:12] Access routines implementation level
	[23:12] Update level of description file
	[11:12] Logical database number
12	Database open information
	[47:12] Stack number of the local buffer manager
	[15: 1] SET if database is running under SWAPPER
	[14: 1] SET if database has label equation
	[13: 6] ASN of local buffer manager
	[7: 8] open type
13–end	Variable-length data pointed to by the LINKs in this entry

Table 12–20. Major Type 1–Minor Types 21 and 22

Minor Type = 21 Database Freeze (LOGMIN_DBFREEZEV)
Minor Type = 22 Database Resume (LOGMIN_DBRESUMEV)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to the database name
5	LINK to code file name of access routines
6	Stack number information [47:12] Database stack number [31:12] Access routine stack number
7–end	Variable-length data pointed to by the LINKs in this entry

The Major Type 1, Minor Type 25 (File Statistics) entry is logged only for processes using anonymous file accounting. For further information, refer to "Using Anonymous Task and File Accounting" earlier in this section.

Table 12-21. Major Type 1–Minor Type 25

Minor Type = 25 File Statistics Entry (LOGMIN_FILESTATSV)	
Words	**Description**
0–3	As described in Table 12-2, "First Four Log Entry Words"
4	Number of disk files used
5	Elapsed I/O time for disk files
6	Transaction count (number of logical I/O operations) for disk files
7	Physical read count for disk files
8	Physical write count for disk files
9	Number of nondisk files used
10	Elapsed I/O time for nondisk files
11	Transaction count (number of logical I/O operations) for nondisk files
12	Physical read count for nondisk files
13	Physical write count for nondisk files

Major Type = 2 Maintenance Entry (LOGMAJMAINT)

Tables 12–22 through 12–25 present peripheral type information that is referred to by many of the log entry descriptions for Major Type 2.

Table 12–22. Peripheral Logical Types

Logical Type	Description
0	NO UNIT
1	Disk
2	ODT
3	Remote
4	CDROM
6	Buffered line printer
7	Train printer
8	HYPERchannel
9	Card reader
10	Pseudo Card Reader
11	Card punch
12	Not used
13	7-track magnetic tape
14	9-track NRZ tape II
15	9-track phase-encoded (PE) or group-coded recording (GCR) tape III
16	Data comm NSP and LSP
17	Pack
18	SCSI unit
20	Host control
21	Floating point array processor (FPSAP)
22	Printer
23	Image printer (in line printer mode)
24	Special
27	ICP
38	EBCDIC buffered printer

continued

Table 12–22. Peripheral Logical Types (cont.)

Logical Type	Description
39	Unbuffered printer
40	Voice channel
45	Tape

Table 12–23. Peripheral Subtypes

Device	Peripheral Subtype	Description
For PACK (DLP types HT1, HTS1, HTS2, SMD1)	0	215 pack
	1	215 pack (new)
	2	225 pack
	3	235 pack
	4	206 pack sequential
	5	206 pack interlaced
	6	207 pack sequential
	7	207 pack interlaced
	8	659 pack sequential
	9	659 pack interlaced
	10	677 pack sequential
	11	677 pack interlaced
	12	3652 pack sequential (1440-byte sectors)
	13	3652 pack interlaced (1440-byte sectors)
	14	3675 pack sequential (1440-byte sectors)
	15	3675 pack interlaced (1440-byte sectors)
	16	SMD (214) pack
	17	SMD (226) pack

continued

Table 12–23. Peripheral Subtypes (cont.)

Device	Peripheral Subtype	Description
	18	3652 pack sequential (180-byte sectors)
	19	3675 pack sequential (180-byte sectors)
	20	3864 12-MB solid state disk
	21	3864 24-MB solid state disk
	22	3864 36-MB solid state disk
	23	3864 48-MB solid state disk
	24	3864 60-MB solid state disk
	25	3864 72-MB solid state disk
	26	B9494-12 (3680) pack sequential
	27	SMD (236) pack
	28	SMD (256) pack
	29	SCSI/131 pack
	30	SCSI/130 pack
	31	SCSI/564 pack
	32	3682 pack
	33	3680 pack sequential (1440-byte sectors)
	34	3680 pack interlaced (1440-byte sectors)
	35	3680 (180-byte sectors)
	36	Twin turbo pack
	37	SCSI/674 pack
	38	SCSI/280 pack
	39	SCSI/564 pack
	40	SCSI/287 pack
	41	SCSI/552 pack
	42–43	Reserved for internal use
	44	SCSI/286 pack
	45	SCSI/546 pack
	46–48	Reserved for internal use

continued

Table 12–23. Peripheral Subtypes (cont.)

Device	Peripheral Subtype	Description
	49	IPI/M2671P pack
	50	SCSI/1367 pack
	63	New SCSI/IPI pack
For TAPE (DLP types MT1, MT2, MT3, MT5, MT6, MTFIPS1)	0	Invalid
	2	Magtape II (9-track NRZ)
	3	Magtape III (9-track PE)
	4	Magtape group-coded recording (GCR)
	6	9-track NRZ tape (no extended status).
	7	9-track PE tape (no extended status).
	9	Half-inch cartridge tape.
For UIO Data comm (DLP types FR1, FR2, FR3, SC1, SC2, SC3, SC4, SC5)	1	Network support processor (NSP)–models 1 and 2
	2	NSP–multiple host model
	3	Line support processor (LSP)—subbroadband, programmable read-only memory (PROM) type
	4	LSP—subbroadband, random access memory (RAM) type
	5	LSP—broadband (bit) type
	6	NSP—blocked
	7	NSP—multiple host, blocked
	8	NSP—data communications data link processor (DCDLP), blocked
For Printers (DLP types TP1, TP2, TP3, TP5)	0	450/750-lines-per-minute train printer.
	1	1100/1500-lines-per-minute train printer.
	3	Other printers using a translation table.
For Host Controls (DLP type HC2)	1	Host control-1 and host control-2

Table 12–24. Peripheral Densities

Device	Peripheral Density	Description
TAPE	0	800-bits-per-inch (bpi) NRZ
	3	1600-bpi PE
	4	6250-bpi GCR
	5	38000-bpi half-inch cartridge tape
	6	1250-bpi quarter-inch cartridge tape

Table 12–25. Peripheral DLP Types

Logical Type	Peripheral Subtype	Peripheral Density	DLP Abbreviation	DLP Name
1	2	2	D5N1	5N DISK
2			ODT1	OPERATOR DISPLAY
2			ODT2	OPERATOR DISPLAY-2
7	0, 1		TP1	450/750/1100/1500 LPM TRAIN PRINTER
7	2		TP2	B9246-X BUFFERED PRINTERS
7			TP3	B924-B/B924-C 1200/2000 LPM BUFFERED DRUM PRINTERS
8			HY	HYPERchannel
9			CR1	CARD READER
11			CP1	CARD PUNCH
14	2	0	MT3	9-TRACK NRZ MAG TAPE
14	2	0	MTFIPS1	FIPS TAPE
15	3	3	MT1	PE MAG TAPE
15	4	3, 4	MT2	GCR MAG TAPE

continued

Table 12–25. Peripheral DLP Types (cont.)

Logical Type	Peripheral Subtype	Peripheral Density	DLP Abbreviation	DLP Name
15		3, 4	MT5	GCR FORMATTER-TYPE MAG TAPE
15	3	3	MT6	STREAMER TAPE
15	3, 4	3, 4	MTFIPS1	FIPS TAPE
15	4	5	MTFIPS3	MTFIPS3 TAPE
15	4	6	SCSITAPE	SCSI TAPE
16	1		SC1	DATA COMM NSP: STANDARD (MODEL 1 & 2)
16	2		SC2	DATA COMM NSP: MULTIPLE HOST
16	3		FR1	DATA COMM LSP: SUB-BROADBAND (PROM)
16	4		FR2	DATA COMM LSP: SUB-BROADBAND (RAM)
16	5		FR3	DATA COMM LSP: 56KB BROADBAND BIT
16	6		SC3	DATA COMM NSP: BLOCKED MESSAGES
16	7		SC4	DATA COMM NSP: EXTENDED MEMORY
16	8		SC5	DATA COMM NSP: DCDLP
1	3, 5, 7, 9, 11		HT1	STANDARD HOST TRANSFER (DISK PACK)
17	4–11		HTS1	SEQUENTIAL HOST TRANSFER (DISK PACK)
17	26		HTS2	SEQUENTIAL HOST TRANSFER 2 (DISK PACK)

continued

Table 12–25. Peripheral DLP Types (cont.)

Logical Type	Peripheral Subtype	Peripheral Density	DLP Abbreviation	DLP Name
17	17, 27, 28		SMD1	STORAGE MODULE DEVICE (DISK PACK)
17	29, 30, 31, 37		SCSIDISK	SCSI Disk
17	29, 30, 31, 37		PK1SCSI(NATIVE)	Native SCSI Disk
17	32		IPIPK1	IPI 9399-H PACK
17	49		IPIPK2	IPI M9730 PACK
20	1		HC2	HOST CONTROL-2
26			IP	IMAGE PRINTER
27			ICP1	INBUILT COMMUNICATIONS PROCESSOR
40			VIM3	VOICE INTERFACE MODULE-3
59			SCSI1	SCSI UNIT (CAN BE DISK, TAPE, OR OTHER TYPE OF SCSI UNIT)

Table 12–26. Major Type 2–Minor Type 11

Minor Type = 11 Library Maintenance Compare (MLLIBERR)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Source information word, consisting of [42: 1] This bit is ON if the source is another host system (network file transfer) [41: 1] Disk flag (1 = disk) [40: 1] Tape flag (1 = tape) [31:12] Index of first word in 30 sector source library maintenance record on which compare failed

continued

Table 12–26. Major Type 2–Minor Type 11 (cont.)

Minor Type = 11 Library Maintenance Compare (MLLIBERR)	
Words	**Description**
	[19:20] Disk address
5	First source word on which the compare failed
6	Destination information word
	[42: 1] This bit is ON if the destination is another host system (network file transfer)
	[41: 1] Disk flag (1 = disk)
	[40: 1] Tape flag (1 = tape)
	[31:12] Index of first word in 30 sector destination library maintenance record on which compare failed
	[19:20] Disk address
7	First destination word on which compare failed
8	External device number of the source unit
9	External device number of the destination unit

Table 12–27. Major Type 2–Minor Type 12

Minor Type = 12 Mainframe Error Entry (MLMAINFRAMERR)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	MCP ID
	[47:12] = System serial number
	[35:12] = Mark digit
	[23:12] = Level number

continued

Table 12–27. Major Type 2–Minor Type 12 (cont.)

Minor Type = 12 Mainframe Error Entry (MLMAINFRAMERR)	
Words	**Description**
	[11:12] = Cycle number Note that the system serial number field ([47:12]) in this word does not have the capacity to store numbers greater than 4095. It is suggested that the user use the 16-bit system serial number field ([47:16]) of Word 9 of the Major Type 6, Minor Type 1 (Halt/Load) entry to avoid getting a misrepresented system serial number.
5	Time mainframe event occurred
6	Mainframe configuration information [47: 8] = System type The rest of this word has one of three different formats, according to the value found in [47:8]. The possible values, and the formats they imply for the rest of the word, are as follows: 1 = A 2 [19:20] = Data processor mask 5 = A 3 6 = A 9 7 = A 15 9 = A 10 10 = A 5 15 = A 12 16 = A 1 19 = A 4 20 = Micro A [31: 8] = Data processor mask [23: 8] = I/O processor mask [15: 8] = Memory mask 17 = A 17 [39: 3] = Console state 0 = Clock set at console

continued

Table 12–27. Major Type 2–Minor Type 12 (cont.)

Minor Type = 12 Mainframe Error Entry (MLMAINFRAMERR)	
Words	**Description**
	1 = Clock set by partition 1
	2 = Clock set by partition 2
	[36: 1] = Console clock validity
	0 = Console state and console ID not used
	1 = Console state and console ID are valid
	[35: 4] = Console ID field
	0, 1, or 2 depending on which console formed the report.
	[23: 8] = RMM mask
	[7: 1] = Distinguished I/O processor (IOP)
	0 = IOP 0 is distinguished
	1 = IOP 1 is distinguished
	[6: 1] = Distinguished task control processor (TCP)
	0 = TCP 0 is distinguished
	1 = TCP 1 is distinguished
	21 = A 16
	[39: 3] = Console state
	0 = Clock set at console
	1 = Clock set by partition 1
	2 = Clock set by partition 2
	[36: 1] = Console clock validity
	0 = Console state and console ID not used
	1 = Console state and console ID are valid
	[35: 4] = Console ID field
	0, 1, or 2 depending on which console formed the report.
	[23: 8] = RMM mask
	[7: 4] = Distinguished I/O processor (IOP)

continued

Table 12–27. Major Type 2–Minor Type 12 (cont.)

Minor Type = 12 Mainframe Error Entry (MLMAINFRAMERR)	
Words	**Description**
	0 = IOP 0 is distinguished 1 = IOP 1 is distinguished [3: 4] = Distinguished task control processor (TCP) 0 = TCP 0 is distinguished 1 = TCP 1 is distinguished
7–14	Not used
15–18	These words are described under "Fields Determined by Error Entry Type" following this table.
19	Error entry information [47: 1] = Report system error action [39: 4] = Box subtype [35: 4] = Box number of second box [31: 8] = Cause of failure CPM causes (all processor types) 1 = Retryable instruction 2 = Alarm interrupt in user code into CM 1 3 = Alarm interrupt in MCP code into CM 1 4 = Interrupt into CM 2 5 = First instruction retry failed 6 = All instruction retries failed 7 = Only CPM entered CM 3 IOP/HDU error causes 1 = Fail IOCB was detected 2 = Fail RD was found 3 = Hung I/O finally terminated

continued

Table 12–27. Major Type 2–Minor Type 12 (cont.)

Minor Type = 12 Mainframe Error Entry (MLMAINFRAMERR)	
Words	**Description**
	4 = No fail RD or fail IOCB
	5 = Memory related IOP error in RD
	6 = Cannot cause a completion event
	7 = Port death event caused
	Memory error causes (MSMS)
	1 = Fail 1 interrupt
	2 = Memory parity interrupt
	3 = Memory parity and fail 1 interrupt
	4 = Fail 1 into CM 2
	5 = Fail register found by polling
	6 = I/O exception memory error
	Environment fault causes
	1 = Fan fault
	2 = Fan fault, box has been removed from system
	3 = Fan fault, system has been halted
	4 = Temperature fault
	5 = Power regulator fault
	System error action causes
	1 = Box removed
	2 = Box removal aborted
	3 = Box reconfigured
	4 = Low memory removed
	5 = MSM THRESHOLD exceeded
	6 = Threshold exceeded (removal of only CPM aborted, CPM can be freed)
	Reconfiguration causes
	1 = Box freed

continued

Table 12–27. Major Type 2–Minor Type 12 (cont.)

Minor Type = 12 Mainframe Error Entry (MLMAINFRAMERR)	
Words	**Description**
	2 = Box acquired
	3 = Box swapped
	Mainframe event causes
	1 = System was halt/loaded
	2 = System dump was requested
	3 = Maintenance enabled/disabled for requestor
	4 = CPM Init Light response
	5 = MSM Init Light response
	6 = Unrecognized fail report from MAINTLINE
	7 = POWERNET event detected
	8 = Nonusable time from TCP communicate
	9 = CPM DEADSTOP report
	TCP error causes
	1 = Unexpected TCP service interrupt
	RMM error causes
	1 = Fail register detected
	Console action causes
	1 = HALT attempt
	5 = CONTINUE attempt
	6 = LOAD attempt
	7 = FORCE DUMP attempt
	8 = BOOT LOAD attempt
	Maintlink event causes
	1 = Unrecognized MS-MCP message
	2 = Unexpected MS-MCP response
	3 = ID in response did not match pending MCP-MS message
	4 = Maintlink message was timed out by the MCP

continued

Table 12–27. Major Type 2–Minor Type 12 (cont.)

Minor Type = 12 Mainframe Error Entry (MLMAINFRAMERR)	
Words	**Description**
	5 = MCP software error with maintlink
	6 = Maintlink came up on new unit
	[23: 8] = Box action
	Type of Entry = Reconfiguration
	1 = Operator requested
	Other types of entries
	1 = Threshold exceeded
	2 = Requestor to memory count excessive
	3 = Memory to requestor count excessive
	4 = Memory down
	5 = CPM lost while taking dump
	6 = Not used
	7 = Would not answer when dialed
	8 = CPM jumped to CM 2
	10 = Memory reconfigured in CM 3
	12 = Box reported a FAN FAULT
	[15: 4] = Type of box
	1 = Central processor module (A 12, A 15)
	2 = I/O processor module
	3 = Memory control module
	4 = Memory module
	5 = Maintenance diagnostic processor
	6 = Card tester
	7 = Memory storage module
	8 = Undefined
	9 = Auxiliary processor module

continued

Table 12–27. Major Type 2–Minor Type 12 (cont.)

Minor Type = 12 Mainframe Error Entry (MLMAINFRAMERR)	
Words	**Description**
	10 = Host data unit
	11 = A Series IOP processor (A 1, A 2, A 3, A 4, A 5, A 9, A 10)
	12 = RMM
	[11: 4] = Box number
	[7: 8] = Type of entry
	1 = CPM <X> interrupt
	2 = IOP <X> interrupt
	3 = Memory <X> interrupt
	4 = System error action
	5 = Mainframe event
	6 = System reconfiguration action
	7 = Box environment fault
	8 = Failed system module
	9 = TCP error
	10 = RMM error
	11 = Console action
	12 = Maintlink Event
20–29	These words are described in "Fields Determined by Error Entry Type" following this table.

Fields Determined by the Error Entry Type

This subsection describes Words 15 to 18 and Words 20 to 29. The contents of these words vary according to the value contained in the Type of Entry field [7:8] of Word 19, Error Entry Information. The descriptions that follow are grouped according to the Type of Entry value they are associated with.

Type of Entry = CPM <X> Interrupt

Word	Description
15	Task serial number
16	Fail register (A 12 and A 15 will be 0)
17	Not used
18	RCW where interrupt occurred
19	Error entry information (See the description of Word 19 in Table 12–27, “Major Type 2–Minor Type 12.”)
20	P1
21	P2
22	P2 TAG
23-29	Not used

Type of Entry = IOP <X> Interrupt

Word	Description
15	RMM = Result descriptor (if FAUCAUSE = 5); Number of port that failed (if FAUCAUSE = 6); Number of DTU that failed (if FAUCAUSE = 8); All others = not used
16	RMM = not used; All others = Fail register
17–18	Not used
19	RMM = not used; All others = Error entry information (See the description of Word 19 in Table 12–27, “Major Type 2–Minor Type 12.”)
20–29	RMM = Not used; All others = Failure data from fail IOCB (when fail IOCB was detected)

Type of Entry = Memory <X> Interrupt

Word	Description
15	Not used
16	Fail register
17	Failed control word if box type = memory storage module
	Bit [45:01] is true for single bit memory errors logged while the LAO diagnostic option is set (A 12, A 15, and A 17)
	24-bit fail register extension, valid only for board types MCSIM and SCHED= memory storage module (A 12 and A 15)
18	Memory subsystem module (MSM) box model information
	[7: 4] Memory Type
	1 = 2 million words per SU

continued

continued

Word	Description
	2 = 4 million words per SU
19	Error entry information (See the description of Word 19 in Table 12–27, “Major Type 2–Minor Type 12.”)
20–29	Not used

Type of Entry = System Error Action

Word	Description
15	Not used
16	Task information (only if bit [47: 1] of Word 19 is 0)
17–18	Not used
19	Error entry information (See the description of Word 19 in Table 12–27, “Major Type 2–Minor Type 12.”)
20–29	Not used

Type of Entry = Mainframe Event

The format of mainframe event entries depends on the failure cause recorded in field [31: 8] of Word 19.

Cause of Failure	Word	Description
1		System was halt/loaded (Large Systems)
	15	Not used
	16	Mainframe configuration
	17	Not used
	18	System halt information
	19	Error entry information
		[43:03] The system halt information
		[43:01] The halt information is valid
		[42:02] Cause of system halt
		0 = The console is incapable of supplying halt information. Valid for console halts.
		1 = Halt was due to manual intervention
		2 = Halt was due to internal problems
		3 = Cause of halt could not be detected

continued

continued

Cause of Failure	Word	Description
	20	Halt/load reason word 1
	21	Halt/load reason word 2
	22–29	Not used
1		System was halt/loaded (EMS systems)
	15	Not used
	16	Mainframe configuration
	17–18	Not used
	19	Error entry information (See the description of Word 19 in Table 12–27, "Major Type 2–Minor Type 12.")
	20	Halt/load reason word 1
	21	Halt/load reason word 2
	22–29	Not used
2		System dump was requested
	15	Task serial number
	16	Mainframe configuration at time of dump (same format as [39:40] of Word 5)
	17	Not used
	18	Dump was fatal
	19	Error entry information (See the description of Word 19 in Table 12–27, "Major Type 2–Minor Type 12.")
	20	Dump reason word 1
	21	Dump reason word 2
	22	Not used
	23–end	As many linked mainframe information words as necessary for A 1, A 2, A 3, A 4, A 5, A 9, A 10 systems
4		CPM Init Light response. This is a normal function and does not represent an error.
	15–29	Not used
5		MSM Init Light response. This is a normal function and does not represent an error.
	15	Not used

continued

continued

Cause of Failure	Word	Description
	16	Encoded halt/load reason
	18–29	Not used
6		Unrecognized fail report from MAINTLINE
	7–16	Data from the HMC buffer
7		POWERNET event detected
	15	Not used
	16	Fail register 1
		[47: 3] 4 = Type of entry
		[44: 5] 1 = Message format
		[39: 8] 2 = Fail entry length
		[23: 8] Logical unit type (hex)
		C0 = Peripheral unit
		C1 = IIO base
		C2 = BX387 disk exchange
		C3 = B9387 disk controller
		C4 = B9389 disk controller
		E1 = CPM
		E2 = HDU
		F1 = MSM
		FF = A 12 mainframe
		[15:16] Logical unit number
	17	Fail register 2
		[47:42] Powernet events
		[47:1] One or more power supplies margined
		[42:1] +5V overvoltage fail
		[41:1] +5V undervoltage fail
		[40:1] –5.2V overvoltage fail
		[39:1] –5.2V undervoltage fail
		[38:1] –2.0V overvoltage fail
		[37:1] –2.0V undervoltage fail
		[36:1] –3.3V overvoltage fail
		[35:1] –3.3V undervoltage fail

continued

continued

Cause of Failure	Word	Description
		[29:1] +5V fail 1
		[28:1] +5V fail 2
		[27:1] –5.2V fail 1
		[26:1] –5.2V fail 2
		[25:1] –2.0V fail 1
		[24:1] –2.0V fail 2
		[23:1] –3.3V fail 1
		[22:1] –3.3V fail 2
		[21:1] +12V fail 2
		[20:1] –12V fail 2
		[19:1] –4.5V fail 2
		[18:1] No response to power net monitor
		[17:1] Input module failure
		[16:1] +5V keep alive fail 1
		[11:1] Fan failure
		[10:1] Air quality failure
		[09:1] Temperature warning
		[08:1] Over temperature failure
		[07:1] 0 = System mode
		1 = Local mode
		[06:1] 0 = Powered on
		1 = Powered off
		[05:06] Most recent event pointer, set to the bit number of the event
	19	Entry info
		[07:8] (FATYPE) 5 = FAMAINEVNTV
		[31:8] (FACAUSE) 7 = SYSPOWERNET
8		Nonusable time from TCP communicate
	15	Not used
	16	Nonusable time
	17–18	Not used
	19	Error entry information
	20–29	Not used

continued

continued

Cause of Failure	Word	Description
9		CPM DEADSTOP report
	15	DEADSTOP word (for example, 4"DEAD00000251")
	16	DEADSTOP parameter (for example, 4"C6C1E4D3E340")
	17	DEADSTOP line number (for example, 4"000023B568")
	18	Not used
	20–29	Not used

Type of Entry = System Reconfiguration Action

Word	Description
15	Not used
16	Task info
17–18	Not used
19	Error entry information (see above)
20–29	Not used

Type of Entry = Failed System Module

Word	Description
17	CPM flip-flops:
	[44: 5] PCU flip-flops:
	[44: 1] = PPRGVC
	[43: 1] = PPRGRC
	[42: 1] = PACE
	[41: 1] = PACO
	[40: 1] = PLHE
	[37: 2] RU flip-flops:
	[37: 1] = RWCODE
	[36: 1] = RWCPER
	[23:24] MAU flip-flops:
	[23: 1] = MAQERR

continued

continued

Word	Description
	[22: 1] = MCATPE
	[21: 1] = MEMTPE
	[20: 1] = MEDFOP
	[19: 1] = MIQPE
	[18: 1] = MCACJE
	[17: 1] = MIDCPE
	[16: 1] = MAQUNF
	[15: 1] = MCACPF
	[14: 1] = MCATCF
	[13: 1] = MCACEF
	[12: 1] = MCAPE
	[11: 1] = MCCAPE
	[10: 1] = MCCOPE
	[9: 1] = MCEPE
	[8: 1] = FATAL
	[7: 1] = MCAERR
	[6: 1] = MCPE
	[5: 1] = MCVLE
	[4: 1] = MCFTE
	[3: 1] = MCCRPE
	[2: 1] = MRQFUL
	[1: 1] = MJRPE
	[0: 1] = MOUERR

Type of Entry = TCP Error

Word	Description
15–17	Not used
18	RCW where interrupt was taken
20	Not used
21	P2
22–29	Not used

Type of Entry = RMM Error

Word	Description
Cause of failure	Fail register detected
7–14	PCD information
15–17	Fail analysis information
18	Control information
20–29	Fail register report

Type of Entry = Console Action

Word	Description
7–18	Not used
20	[47: 5] load reason
	6 = Console load
	7 = Force dump
	8 = Boot load
	16 = Console load (autoload after deadstop 166)
	17 = Console load (autoload after deadstop 366)
	[23: 6] Mask of requestors failed disconnect
	[17: 6] Mask of requestors in partition
	[5: 1] TCP(1) disorderly halt
	[4: 1] TCP(0) disorderly halt
	[3: 1] IOP(1) disorderly halt
	[2: 1] IOP(0) disorderly halt
	[1: 1] MIC(1) disorderly halt
	[0: 1] MIC(0) disorderly halt
21	First through sixth EBCDIC characters of PCD name
22	[47:16] = Seventh and eighth characters of PCD name
	[31:16] = PCD hash ID
23–29	Not used

Type of Entry = Maintlink Event

Word	Description
7–18	Not used
20	MAINTLINEINFO
	[47:12] MAINTLINEIR stack number
	[35:12] Stack number of head of queue of waiters for response
	[23: 5] Message type for which the independent runner is awaiting a response
	[18:15] Maintline ODT unit number
	[3: 1] Maintline is up
	[2: 1] Maintline has error
	[1: 1] Maintline idle (not awaiting response)
	[0: 1] Maintline has outstanding memory storage unit (MSU) update
21	MAINTLINEINFO2
	[47: 8] Expected type of response
	[39: 8] Expected subtype of response
	[31:32] Message ID of message requires response
22	MAINTLINE INFO3
	[47: 1] IO timed out on an IO to the MAINTLINE unit
	[46: 1] The MAINTLINE has been up
	[45: 1] A NAK limit error has occurred
	[31:32] Message ID of most recent ??CONSL FAIL MS-MCP message
23–29	Not used

Table 12–28. Major Type 2–Minor Type 14

Minor Type = 14 DFO Error Entry (MLDFOERROR)	
Words	**Description**
This log entry type is not currently used.	

Table 12–29, "Major Type 2–Minor Type 16," describes the original minor type used for logging I/O errors. In a future release, all I/O errors will be logged in the Major Type 2, Minor Type 21 (MLIOEXCEPTION) entry, and Minor Type 16 will be deimplemented.

Table 12–29. Major Type 2–Minor Type 16

Minor Type = 16 IOP I/O Error Entry (MLIOERRORHDP)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	EMS/HDU/RMM log type
	[47: 8] EMS/HDU/RMM log type
	1 = This is an error-recovery log entry.
	2 = This is an error-IOCB log entry.
The rest of the entry is in either of two different formats, depending on the value contained in EMS/HDU/RMM log type, above.	
If this is an error-recovery log entry, as indicated by a value of 1 in field [47: 8], then the remainder of the log entry has the following format:	
4	Retry information.
	[39: 8] Log variant
	1 = Log entry is for normal error recovery and all information for a retry is present.
	2 = Log entry is for limited error recovery and some of the information for each retry is not present. The DLP command and the IOP CONTROL are not present for retry I/O operations. This entry type is only available on EMS systems.
	[27:12] Number of retries that were logged
	[15: 1] Not recovered
	0 = The I/O was recovered
	1 = The I/O could not be recovered
	[13: 2] Extended status control
	0 = No extended status information is available
	1 = The extended status information was read correctly
	2 = An error occurred while trying to read the extended status
	3 = No error occurred. The statistical data could not be logged.
	[11: 1] Extended status area

continued

Table 12–29. Major Type 2–Minor Type 16 (cont.)

Minor Type = 16 IOP I/O Error Entry (MLIOERRORHDP)	
Words	**Description**
	0 = The extended status information was put in the extended result descriptor area (words 24-33)
	1 = The extended status information was put in the variable length information area
	[10: 1] HDU system
	0 = This is not an HDU system
	1 = This is an HDU system
	[9:10] Extended status byte length
5	UNIT information.
	[47:12] Logical unit number
	[35:12] Internal device index
	[23: 6] Logical unit type (See Table 12–22, "Peripheral Logical Types.")
	[17: 6] Unit subtype
	[11:12] DLP ID
6	File and job information
	[47: 4] Number of file and job information items logged.
	File information includes the file title, the internal name, and the serial number. Job information is the job name.
	[43: 4] EMS/HDU/RMM log version number.
	0 = Before Mark 3.4.0 release
	1 = Command word 2 was added
	2 = RMM machines were included (retry area was enlarged)
	3 = I/O statistics words 3 and 4 were added
	4 = I/O statistics words 3 and 4 for RMM machines were added
	[13: 1] RMM machine
	0 = This is not an RMM machine; refer to Word 4, field [10: 1]
	1 = This is an RMM machine
	[12: 1] Result descriptors are missing from list.

continued

Table 12–29. Major Type 2–Minor Type 16 (cont.)

Minor Type = 16 IOP I/O Error Entry (MLIOERRORHDP)	
Words	**Description**
	[11:12] File and job information location.
	The word offset in this entry where the file and job information starts.
The remainder of the fixed portion of the log entry contains information about the original I/O.	
7	IOP control word
	[19:20] IOP CONTROL (EMS systems and HDU systems)
	RMM control word (RMM systems)
	[31:32] IOP control field
	[19: 1] Word mode
	0 = Character mode operation
	1 = Word mode operation
8	DLP address for EMS systems and HDU systems.
	Path word for RMM machines
	[47:16] Port number
	[31:16] Control number
	[15:16] Unit number
9	Device number
10	For HDU systems, the device number of the DLP used during the I/O. For EMS systems, this word is 0.
11	RMM bus information
12	IOP state and result
13	I/O START time
14	STACK numbers
	[47:12] STACK number of IOCB owner
	[35:12] STACK number of I/O initiator
15	MIX numbers of process charged for the I/O action
	[47:16] JOB number
	[31:16] TASK number

continued

Table 12–29. Major Type 2–Minor Type 16 (cont.)

Minor Type = 16 IOP I/O Error Entry (MLIOERRORHDP)	
Words	**Description**
16	I/O control word
17	AREA length descriptor
	[27:20] Length of area
18	AREA address descriptor
	[31:32] Address of area
19	Logical result descriptor
20	Exception ID
21	DLP command (word 1)
22	DLP result (word 1)
23	DLP result (word 2)
24–33	Extended result descriptor area. Used when extended result descriptor information is 10 words or less. Otherwise, the variable length information area is used (see below).
34	I/O statistics (word 1)
	[47:24] Total number of read I/O operations to unit since the last halt/load.
	[23:24] Total number of write I/O operations to unit since the last halt/load.
35	I/O statistics (word 2)
	[47:24] Total number of read errors to unit since the last halt/load.
	[23:24] Total number of write errors to unit since the last halt/load.
36	DLP command (word 2)
37	I/O statistics (word 3)
	[47:24] Total number of read errors successfully recovered by the controller.
	[23:24] Total number of write errors successfully recovered by the controller.
38	I/O statistics (word 4)
	Total bytes transferred to or from unit since halt/load or the last time the counter was initialized.

continued

Table 12–29. Major Type 2–Minor Type 16 (cont.)

Minor Type = 16 IOP I/O Error Entry (MLIOERRORHDP)	
Words	**Description**
This section contains retry entries of up to eight words each. The number of retries (Word 4) that are logged is variable, so this section is of variable length. The format of each entry is as follows:	
39–N	RETRY entries
Entry Word 0	Retry DLP command (word 1) (not present if the field [39:8] of Word 4 stores a log variant of 2)
Entry Word 1	Retry DLP result (word 1)
Entry Word 2	Retry DLP result (word 2)
Entry Word 3	[47: 1] Actual retry
	0 = Entry is not actual attempt to retry I/O (for example, repositioning a tape)
	1 = Entry is an actual attempt to retry I/O
	[19:20] RETRY IOPCONTROL (not present if the field [39:8] of Word 4 stores a log variant of 2)
	[31:32] Control field
Entry Word 4	Retry IOP STATE and RESULT
Entry Word 5	Retry DLP command (word 2)
Entry Word 6	Path word (RMM machines)
	[47:16] Port number
	[31:16] Control number
	[15:16] Unit number
Entry Word 7	IOP miscellaneous information word (RMM machines)
	[47: 1] Entry is an actual retry
	[46:17] LRD
	[29:10] Exception ID
This section contains file, job, and extended status information. Because the length of the previous section varies, the location of the starting word of this section is also variable. The index to the starting word of this section is in field [11:12] of Word 6. The number of file and job information items (field [47:4] of Word 6) is variable, as is the length of each item. The first word of each item specifies the type and length of the item.	

continued

Table 12–29. Major Type 2–Minor Type 16 (cont.)

Minor Type = 16 IOP I/O Error Entry (MLIOERRORHDP)	
Words	**Description**
N+1–end	Variable-length information
0	Item descriptor
	[39:20] length of item (including this word)
	[19:20] TYPE of item
	1 = Item is SERIAL NUMBER information
	2 = Item is FILE TITLE information
	3 = Item is JOB NAME information
	7 = Item is INTERNAL NAME information
	9 = Item is EXTENDED STATUS information
If TYPE = 1	(SERIAL NUMBER)
	Note: *Words 1 through 3 of the SERIAL NUMBER information are not valid for disks or packs.*
1	LEBCONTROL
2	GENEALOGY1
	[47: 9] Label level
	[38: 1] 1 if CYCLE was specified
	[37:14] CYCLE
	[23: 8] VERSION
	[15: 1] 1 if FILESECTION was specified
	[14:15] FILESECTION number
3	GENEALOGY2
	[47:16] System serial number of system that created file
	[31: 4] GENERATION attribute
	[27:11] SAVEFACTOR attribute
	[16:17] CREATIONDATE (Julian)
4	SERIALNO in EBCDIC (Present only if the unit is a pack or mag tape)

continued

Table 12–29. Major Type 2–Minor Type 16 (cont.)

Minor Type = 16 IOP I/O Error Entry (MLIOERRORHDP)	
Words	**Description**
If TYPE = 2	(FILE TITLE)
1	Area number of the file on pack or disk. This word is all ones if no valid row index is present or if the unit is not a pack or disk.
2–n	If the length of the FILE TITLE item is greater than two, the rest of the item is the file title in standard form.
If TYPE = 3	(JOB NAME)
1–n	The job name in standard form
If TYPE = 7	(INTERNAL NAME)
1–n	The internal name in standard form
If TYPE = 9	(EXTENDED STATUS)
1–n	The EXTENDED STATUS information (used when the extended status data length is greater than 10 words)
If this is an error-IOCB log entry, as indicated by a value of 2 in field [47: 8] of Word 4, then the remainder of the log entry has the following format:	
4	Number of DLP result area words [7: 8] Number of words of information in the DLP result area of the Error IOCB
5–n	Error IOCB information This section contains the DLP result area information of the error IOCB. Each DLP result area word is stored as 52 bits of information starting at Word 5 in the log entry. Figure 12–1, "DLP Result Format for IOP I/O Errors," shows the format of this area. Refer to individual system reference manuals for a more detailed description of the information returned in these words. Note that the whole error IOCB is not returned.

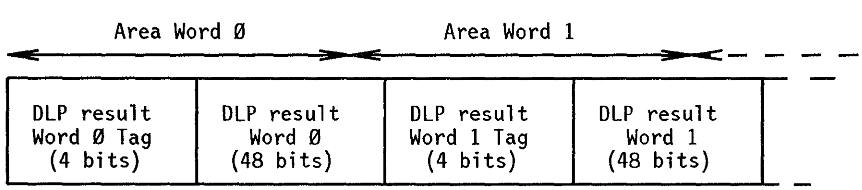

Figure 12–1. DLP Result Format for IOP I/O Errors

For more information about peripheral types, see Tables 12–22 through 12–25.

The Hardware Configuration log entry uses LINKs to store information in a tree structure. One LINK is at the top of the tree; this LINK describes a region that contains several additional LINKs. Each of these LINKs describes a region that contains information about data processors, I/O processors, memory, or peripherals. For further information about the LINKs used in this entry, refer to the "Format of Hardware and Software Configuration LINK Words" subsection earlier in this section.

The fixed part of the hardware configuration entry consists of the information in Table 12–30, "Major Type 2–Minor Type 17."

Table 12–30. Major Type 2–Minor Type 17

Minor Type = 17 Hardware Configuration (HARDWARE_CONFIG)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	General information [47: 3] Partition number (A 12, A 15, A 16, and A 17 systems) [44: 5] Number of data processors in system [39: 8] Number of I/O processors in system [31: 8] System type. For a list of the possible values, refer to the description of Word 3, field [15:16] in Table 12–66, "Major Type 10–All Minor Types." [23: 8] Total number of memory subsystem modules (MSMs) (A 12, A 15, A 16, and A 17) [15: 4] EMODE level
5	LINK to the hardware information
6–end	Variable-length hardware information pointed to by the LINK in Word 5

All of the actual hardware configuration information resides in the variable part of the configuration entry. Accessing any of this information requires that the LINK at Word 5 be examined.

The LINK at Word 5 contains the following values:

LINK_TYPE	1
LINK_VERSION	0
ITEM_COUNT	10 (LINKs)
ITEM_WIDTH	1 (word)

The format of the region described by the LINK at Word 5 varies, depending on the system type. For A 1, A 2, A 3, A 4, A 5, A 6, A 9, A 10, and Micro A systems, this region has the following format:

Offset	Contents	LINK_TYPE
0	LINK to data processor information	2
1	LINK to I/O processor information	3
2	LINK to memory information	4
3	LINK to peripheral information	5
4	LINK to extended pack information	6
6	LINK to physical memory configuration (A 9 or A 10 systems)	20

The value given in each LINK in the preceding table and the format of the information corresponding to each LINK are given under "Link Descriptions for EMS Systems" later in this section.

On A 12 and A 15 systems, the format is as follows:

Offset	Contents	LINK_TYPE
0	LINK to data processor information	2
1	LINK to I/O processor information	11
2	LINK to memory information	10
3	LINK to peripheral information	17
4	LINK to extended pack information	19
5	LINK to I/O base information	15

The value contained in each LINK in the preceding table and the format of the information corresponding to each LINK are given under "Link Descriptions for HDU Systems" later in this section.

On A 17 systems, the format is as follows:

Offset	Contents	LINK_TYPE
0	LINK to data processor information	2
1	LINK to I/O processor information	21
2	LINK to memory information	10
3	LINK to peripheral information	17
4	LINK to extended pack information	19
5	LINK to path group information	28
7	LINK to task control processor	29
9	LINK to continuation information	207

The value contained in each LINK in the preceding table and the format of the information corresponding to each LINK are given under "Link Descriptions for A 17 Systems" later in this section.

Note: *If the LINK to continuation information in an A 17 hardware configuration entry is not 0, then the hardware configuration information is distributed over two or more log entries. Refer to the description of "LINK [5] + 9 : Continuation Information (A 17 Systems)" later in this section.*

On A 16 systems, the format is as follows:

Offset	Contents	LINK_TYPE
0	LINK to data processor information	2
1	LINK to I/O processor information (RMM and IOM machines)	21
2	LINK to memory information	10
3	LINK to peripheral information	17
4	LINK to extended pack information	19
5	LINK to path group information (RMM and IOM machines)	25
6	0 (not used)	
7	LINK to task control processor	29
8	LINK to DOMAIN status	30

The value contained in each LINK in the preceding table and the format of the information corresponding to each LINK are given under "Link Descriptions for A 16 Systems" later in this section.

The array indices given in the *Offset* column in the preceding tables are not absolute indices; they are relative to the START_INDEX of the LINK at Word 5. For example, on A 12, and A 15 systems, the LINK to I/O base information is found at array entry [5] + 5.

Link Descriptions for EMS Systems

LINK [5] + 0 : Data Processor Information (EMS Systems)

This LINK has the following values:

LINK_TYPE	2
LINK_VERSION	0
ITEM_COUNT	number of data processors
ITEM_WIDTH	7 (words)

Each data processor entry has the following format:

Entry Word	Description
0	PROCESSOR_NUMBER
1	PROCESSOR_TYPE 1 = A 2 5 = A 3 6 = A 9 9 = A 10 10 = A 5 16 = A 1 19 = A 4 20 = Micro A
2–3	0
4	PROCESSOR_IOP_MASK I/O processors that this data processor can see; bit "i" is ON if this data processor can see I/O processor number "i".
5	PROCESSOR_CACHE_STATUS 0 = Not present 1 = Disabled 2 = Enabled
6	0

LINK [5] + 1 : I/O Processor Information (EMS Systems)

This LINK has the following values:

LINK_TYPE	3

continued

continued

LINK_VERSION	0
ITEM_COUNT	Number of I/O processors
ITEM_WIDTH	4 (words)

Each I/O processor entry has the following format:

Entry Word	Description
0	IOP_NUMBER
1	IOP_TYPE Currently, this word always stores a 4.
2	IOP_FIRMWARE_LEVEL Currently, this word is always 0.
3	IOP_PROCESSOR_MASK Data processors that this I/O processor can see; bit "i" is ON if this I/O processor can see data processor number "i".

LINK [5] + 2 : Memory Information (EMS Systems)

This LINK has the following values:

LINK_VERSION	0
LINK_TYPE	4
ITEM_COUNT	Number of memory chunks
ITEM_WIDTH	5 (words)

Each memory entry has the following format:

Entry Word	Description
0	This word always stores a 3. Prior to the Mark 3.9 system software release, this word contained the box or ASN number and the following fields: [1: 1] GLOBAL_MEMORY [0: 1] WRITE_OK
1	MEMORY_START_ADDRESS
2	MEMORY_END_ADDRESS
3	MEMORY_PROCESSOR_MASK Data processors that can see this memory; bit *i* is ON if I/O processor number *i* can see this memory.

continued

continued

Entry Word	Description
4	MEMORY_IOP_MASK The I/O processors that can see this memory; bit i is ON if data processor number i can see this memory.

Prior to the Mark 3.9 system software release, it was possible for two entries to be made for the same chunk of GLOBAL Memory. Two entries are made if one data processor has read-only access to some part of GLOBAL Memory. One entry has GLOBAL_MEMORY and WRITE_OK SET, and the data processor mask is SET to show all processors except the one with read-only access. Another entry has GLOBAL_MEMORY SET, WRITE_OK RESET, and the data processor mask SET to show the processor with read-only access to that area of memory.

LINK [5] + 3 : Peripheral Information (EMS Systems)

This LINK has the following values:

LINK_TYPE	5
LINK_VERSION	0
ITEM_COUNT	Number of peripheral units
ITEM_WIDTH	3 (words)

Each peripheral information entry has the following format:

Entry Word	Description
0	Peripheral unit information [47:12] Physical unit number [35: 8] Logical type (See Table 12–22, "Peripheral Logical Types.") [27: 8] Subtype or the number 63, indicating that the peripheral subtype is stored in Word 2. For a list of subtypes, refer to Table 12–23, "Peripheral Subtypes." [19: 4] Density (See Table 12–24, "Peripheral Densities.") [15: 1] 1 if unit is saved [14: 1] 1 if unit is reserved [13: 1] 1 if unit has been seen ready since the last halt/load [12: 1] 1 if unit is currently available to group [11:12] Reserved
1	LINK to path information
2	If field [27: 8] of word 0 stores a value of 63, then this word contains the peripheral subtype in bits [16:17]. See Table 12–23, "Peripheral Subtypes."

PERIPHERAL_SEEN_RY is SET if the unit has been ready at any time since the last halt/load, even if the device is not currently ready.

PERIPHERAL_AVAIL is SET if the peripheral is currently available to the group. In a loosely coupled system, paths to a device from two or more groups can be present. In this case, one of these groups “acquires” the device and the device becomes available to that group. The other groups might have a valid path to the device, but that device is not available to those groups.

The subtype information reported for printer, pack, and disk devices is based on the information in the unit table of the MCP. The unit table is built from information in the configuration file. If a printer, pack, or disk has not been seen as ready, its subtype often might not be correct.

The I/O Processor (IOP) paths correspond to the LINK_TYPES that are in Entry Word 1 in a peripheral entry. They are defined in the following subsection.

I/O Processor Path

IOP path information is described by a LINK with the following values:

LINK_TYPE	9
LINK_VERSION	0
ITEM_COUNT	Number of paths
ITEM_WIDTH	2 (words)
START_INDEX	0

Each IOP path entry has the following format:

Entry Word	Description
0	[47: 8] PATH_ID
	[39: 8] IOP_NUMBER
	[31: 4] IOP_PORT_NUMBER
	[27: 4] LEM_PORT_NUMBER
	[23: 4] RELATIVE_DLP (behind the base)
	[19: 4] RELATIVE_UNIT (behind the DLP)
	[15:12] DLP_ID
	[3: 1] PATH_RESERVED
	[2: 3] Reserved
1	FIRMWARE_LEVEL
	(4 hex digits, right-justified)

LINK [5] + 4 : Extended Pack Information (EMS Systems)

This LINK has the following values:

LINK_TYPE	6
LINK_VERSION	0
ITEM_COUNT	Number of packs online
ITEM_WIDTH	5 (words)

Each extended pack entry has the following format:

Entry Word	Description
0	[47:12] Unit number
	[35: 8] Family index number
	[27: 1] MODE (IN = 0, I/O = 1)
	[26:27] Reserved
1	Serial number (in EBCDIC characters)
2	[47: 8] Number of characters in the pack name
	[39:40] First five characters of the pack name
3	Next six characters of the pack name
4	Last six characters of the pack name

LINK [5] + 6 : Physical Memory Configuration (A 9 and A 10 Systems)

This LINK has the following values:

LINK_TYPE	20
LINK_VERSION	1
ITEM_COUNT	9
ITEM_WIDTH	1 (word)

Each A 9 memory configuration entry has the following format:

Entry Word	Description
0	[15: 8] Number of processors (Always 1 on an A 9)
	[7: 8] Processor number
1–8	These words share the following format. Note that each of these words is 0 if the corresponding hub is not present.
	[12: 4] Module Type
	2 = A 9 memory
	[8: 1] Upper submodule present
	[7: 1] Lower submodule present

continued

continued

Entry Word	Description
	[6: 3] Configuration address
	[1: 2] Submodule online bits

Each A 10 memory configuration entry has the following format:

Entry Word	Description
0	[15: 8] Number of processors
	[7: 8] Processor number
1–4 and 6–9	These words share the following format. Note that each of these words is 0 if the corresponding hub is not present.
	[34: 4] Memory module ID interlacing bits
	[26: 1] Addr3 interlace bit submodule 1
	[25: 1] Addr2 interlace bit submodule 1
	[24: 1] Addr3 interlace bit submodule 0
	[23: 1] Addr2 interlace bit submodule 0
	[22: 3] Submodule 1 configuration address
	[12: 4] Module type (always 4)
	[8: 1] Submodule 1 present
	[7: 1] Submodule 0 present
	[6: 3] Submodule 0 configuration address
	[1: 1] Submodule 1 online
	[0: 1] Submodule 0 online
5	[7: 8] Processor number

Link Descriptions for HDU Systems

LINK [5] + 0 : Data Processor Information (HDU Systems)

This LINK has the following values:

LINK_TYPE	2
LINK_VERSION	0
ITEM_COUNT	Number of data processors
ITEM_WIDTH	7 (words)

Each data processor entry has the following format:

Entry Word	Description
0	PROCESSOR_NUMBER
1	PROCESSOR_TYPE
	7 = A 12, A 15
2	PROCESSOR_GMM_PORT_ADDRESS
	(four characters, left-justified)
3	PROCESSOR_FIRMWARE_LEVEL
	[39: 8] Stored Logic Level Major field
	[31:16] Stored Logic Level Minor field
	[15:16] Creation Date: constructed as (year – 1970) * 1000 + day_of_year
4	PROCESSOR_IOP_MASK
	I/O processors that this data processor can see; bit *i* is ON if this data processor can see I/O processor number *i*.
5	Not used
6	A15 PROCESSOR_HARDWARE_LEVEL
	[35:12] ERL Minor field
	[23: 8] ERL Major field
	[15: 4] EMODE_LEVEL: 1 = Non-ASD
	[7: 8] MACHINE_TYPE: 6 = A 15

LINK [5] + 1 : I/O Processor Information (HDU Systems)

This LINK has the following values:

LINK_TYPE	11
LINK_VERSION	0
ITEM_COUNT	Number of HDUs on system
ITEM_WIDTH	4

Each I/O processor entry has the following format:

Entry Word	Description
0	HDU information
	[47: 1] = FREEd
	[46:47] = Not used
1	HDU name
2	HDU firmware

continued

continued

Entry Word	Description
3	Message level interface (MLI) LINK

The MLI LINK word has these values:

LINK_TYPE	12
LINK_VERSION	0
ITEM_COUNT	Number of MLIs per HDU
ITEM_WIDTH	3

The MLI entry has the following format:

Entry Word	Description
0	MLI information
	[47: 1] = Reserved
	[46: 1] = No path
	[45:46] = Not used
1	MLI name
2	LEM LINK or base LINK

The LEM LINK word has these values:

LINK_TYPE	13
LINK_VERSION	0
ITEM_COUNT	1 (only 1 LEM possible per MLI)
ITEM_WIDTH	2

The LEM entry has the following format:

Entry Word	Description
0	Lem name
1	Base LINK

The base LINK word has these values:

LINK_TYPE	14
LINK_VERSION	0
ITEM_COUNT	Number of bases on LEM or MLI
ITEM_WIDTH	1

The base entry has the following format

Entry Word	Description
0	Base name

LINK [5] + 2 : Memory Information (A 12, A 15 Systems)

This LINK has the following values:

LINK_TYPE	10
LINK_VERSION	1
ITEM_COUNT	37
ITEM_WIDTH	0

Each memory entry has the following format:

Entry Word	Description
0	MEMMASK
1 – 128	PAGEINFO The format of PAGEINFO is as follows: each word contains eight 6-bit fields for a total of 1024 fields. Each field represents a page of logical memory. This logical page does not correspond to a physical page of memory in the MSM. The format of each 6-bit field is as follows: [5: 1] = To be saved [4: 5] = Page status 0 = Undefined 1 = Not present 2 = Available 3 = In use 4 = Saved 5 = Not used 6 = Bad 7 = Memory Disk
129	MSUCONFIG This word is composed of 32 1-bit fields. Each field indicates whether the MSU which it represents is physically present. MSU M of SIM S corresponds to bit 8 * S + M.
130	MSUENABLED This word is composed of 32 1-bit fields, which are indexed in the same manner as MSUCONFIG. Each field indicates whether the MSU which it represents is enabled.

continued

continued

Entry Word	Description
131	MSUSAVED This word is composed of 32 1-bit fields, which are indexed in the same manner as MSUCONFIG and MSUENABLED. Each field indicates whether the MSU which it represents is marked as *to-be-saved* for the next halt/load.
132	MSM status [35: 4] = SIM status [31: 2] = Maintenance Bus mode [27: 1] = Fan Fault [26: 1] = MSM name [25: 1] = Time-of-day valid [24: 1] = Temperature fault [23: 1] = Power fault [22: 6] = Halt/load reason [16: 6] = ERL [10: 1] = Scheduler status [9: 1] = Maintenance bus enable [8: 1] = Scrub memory enable [7: 4] = Model number [3: 4] = Box type (MSM II type = 5)

LINK [5] + 3 : Peripheral Information (HDU Systems)

This LINK has the following values:

LINK_TYPE	17
LINK_VERSION	0
ITEM_COUNT	Number of units on system
ITEM_WIDTH	3

Each peripheral information entry has the following format:

Entry Word	Description
0	Unit information [47: 8] = Peripheral type (See Table 12-22, “Peripheral Logical Types.”) [39: 8] = Peripheral subtype (See Table 12–23, “Peripheral Subtypes.”) [31: 9] = Density (See Table 12–24, “Peripheral Densities.”) [22: 1] = Saved

continued

continued

Entry Word	Description
	[21: 1] = URed
	[20: 1] = URed for maintenance
	[19: 1] = Ready
	[18: 1] = Available
	[17: 1] = No path to unit
	[16:17] = Not used
1	Unit name
2	Path LINK

The path LINK word has these values:

LINK_TYPE	18
LINK_VERSION	0
LINK_COUNT	Number of paths (DLPs) to the unit
ITEM_WIDTH	3

The entry for each path has the following format:

Entry Word	Description
0	Path information
	[47: 1] = No path to unit through this DLP: DLP is URed, UR MAINT, freed, or no path
	[46:43] = Not used
	[3:4] = Subsystem protocol
	0 = Hex
	1 = EBCDIC
	2 = Decimal
	3 = ASCII
1	DLP name
2	Firmware information
	[47: 8] = Firmware length
	[39:40] = Firmware ID

LINK [5] + 4 : Extended Pack Information (HDU Systems)

This LINK has the following values:

LINK_TYPE	19
LINK_VERSION	0

continued

continued

ITEM_COUNT	Number of packs on line
ITEM_WIDTH	6

Each extended pack entry has the following format:

Entry Word	Description
0	Extended information
	[47: 8] = Family index number
	[39: 1] = Pack mode
	0 = Read only
	1 = Read/write
	[38:39] = Not used
1	Unit name
2	Serial number (in EBCDIC characters)
3	[47: 8] Number of characters in the pack name
	[39:40] First five characters of the pack name
4	Next six characters of the pack name
5	Last six characters of the pack name

LINK [5] + 5 : I/O Base Information (HDU Systems)

The LINK word has these values:

LINK_TYPE	15
LINK_VERSION	0
ITEM_COUNT	Number of bases on the system
ITEM_WIDTH	3

The base entry has the following information:

Entry Word	Description
0	Base information
	[47: 1] = Reserved
	[46: 1] = Reserved for maintenance
	[45: 1] = No path
	[44:45] = Not used
1	Base name
2	DLP LINK

continued

continued

The DLP LINK word has these values:

LINK_TYPE	16
LINK_VERSION	0
ITEM_COUNT	Number of DLPs in the base
ITEM_WIDTH	3

The DLP entry has the following information:

Entry Word	Description
0	DLP information
	[47: 8] = DLP type
	[39: 1] = Freed
	[38: 1] = Reserved
	[37: 1] = Reserved for maintenance
	[36: 1] = No path
	[35:36] = Not used
1	DLP name
2	Firmware level

Link Descriptions for A 17 Systems

LINK [5] + 0 : Data Processor Information (A 17 Systems)

This LINK has the following values:

LINK_TYPE	2
LINK_VERSION	0
ITEM_COUNT	Number of data processors
ITEM_WIDTH	7 (words)

Each data processor entry has the following format:

Entry Word	Description
0	PROCESSOR_NUMBER
1	PROCESSOR_TYPE
	17 = A 17
2	Processor is reserved (four characters, left-justified).
3	PROCESSOR_FIRMWARE_LEVEL
	[39: 8] Stored Logic Level Major field

continued

continued

Entry Word	Description
	[31:16] Stored Logic Level Minor field
	[15:16] Creation Date: constructed as (year - 1970) * 1000 + day_of_year
4	PROCESSOR_IOP_MASK
	I/O processors that this data processor can see; bit *i* is ON if this data processor can see I/O processor number *i*.
5	Not used
6	A17 PROCESSOR_HARDWARE_LEVEL
	[35:12] Engineering release level minor field
	[23: 8] Engineering release level major field
	[15: 4] EMODE_LEVEL: 1 = Non-ASD
	[7: 8] MACHINE_TYPE: 6 = A 15

LINK [5] + 1 : I/O Processor Information (A 17 Systems)

Note: *The format of I/O information logged for A 17 systems is changing in a future release. The new format will be the same as that described under "LINK [5] + 1 : Version 2 I/O Processor Information (IOM and RMM Systems)" later in this section. A program can determine which format is being used by inspecting the LINK_VERSION value, which will change from 0 to 2 when the format changes.*

A warning is now displayed whenever a process uses the LOG_GET or LOG_READ procedures of the SDASUPPORT library to read a Hardware Configuration log entry. Programs that use the SDASUPPORT library can suppress this warning message by first invoking the LOG_SELECT procedure and passing a value of 1021 to the STYPE parameter.

This LINK has the following values:

LINK_TYPE	21
LINK_VERSION	0
ITEM_COUNT	1
ITEM_WIDTH	5

Each I/O processor entry has the following format:

Entry Word	Description
0	IOP information
	[47:44] = Not used

continued

continued

Entry Word	Description
	[3: 4] = IOP box number
1	IOP firmware word 1
2	IOP firmware word 2
3	DTU LINK
4	Port LINK

The data transfer unit (DTU) LINK word has these values:

LINK_TYPE	22
LINK_VERSION	0
ITEM_COUNT	Number of DTUs on the system
ITEM_WIDTH	3

Each DTU entry has the following format:

Entry Word	Description
0	DTU information
	[47: 1] = DTU is reserved
	[46: 1] = In use
	[45: 1] = Dead
	[44: 1] = Out of service
	[43: 1] = Online
	[42: 5] = Not present
	[41: 4] = (Reserved field)
	[37: 1] = RMM number
	[36:21] = (Reserved field)
	[15:16] = DTU identification
1	DTU firmware word 1
2	DTU firmware word 2

The Port LINK word has these values:

LINK_TYPE	23
LINK_VERSION	0
ITEM_COUNT	Number of ports on the system
ITEM_WIDTH	3

Each port entry has the following format:

Entry Word	Description
0	Port information
	[47: 1] = Port is reserved
	[46: 1] = In use
	[45: 1] = Dead
	[44: 1] = Out of service
	[43: 1] = Online
	[42: 1] = (Reserved field)
	[41: 4] = Subsystem protocol
	0 = SCSI maintenance bus (SMB)
	1 = Message level interface (MLI)
	2 = Intelligent peripheral interface (IPI)
	3 = Small-computer system interface (SCSI)
	[37: 1] = Resource management module (RMM) number
	[36:21] = (Reserved field)
	[15:16] = Port identification
1	Port firmware word 1
2	Port firmware word 2

LINK [5] + 2 : Memory Information (A 17 Systems)

This LINK has the following values:

LINK_TYPE	10
LINK_VERSION	1
ITEM_COUNT	37
ITEM_WIDTH	0

Each memory entry has the following format:

Entry Word	Description
0	MEMMASK
1–128	PAGEINFO The format of PAGEINFO is as follows: each word contains eight 6-bit fields for a total of 1024 fields. Each field represents a page of logical memory. This logical page does not correspond to a physical page of memory in the MSM. The format of each 6-bit field is as follows: [5: 1] = To be saved [4: 5] = Page status 0 = Undefined 1 = Not present 2 = Available 3 = In use 4 = Saved 5 = Not used 6 = Bad 7 = Memory Disk
129	MSUCONFIG This word is composed of 32 1-bit fields. Each field indicates whether the MSU that it represents is physically present. MSU M of SIM S corresponds to bit 8 * S + M.
130	MSUENABLED This word is composed of 32 1-bit fields, which are indexed in the same manner as MSUCONFIG is. Each field indicates whether the MSU which it represents is enabled.
131	MSUSAVED This word is composed of 32 1-bit fields, which are indexed in the same manner as MSUCONFIG and MSUENABLED are. Each field indicates whether the MSU that it represents is marked as *to-be-saved* for the next halt/load.
132	MSM status [35: 4] = SIM status [31: 2] = Maintenance bus mode [27: 1] = Fan fault [26: 1] = MSM name [25: 1] = Time-of-day valid [24: 1] = Temperature fault

continued

continued

Entry Word	Description
	[23: 1] = Power fault
	[22: 6] = Halt/load reason
	[16: 6] = ERL
	[10: 1] = Scheduler status
	[9: 1] = Maintenance bus enable
	[8: 1] = Scrub memory enable
	[7: 4] = Model number
	[3: 4] = Box type (MSM II type = 5)

LINK [5] + 3 : Peripheral Information (A 17 Systems)

This LINK has the following values:

LINK_TYPE	17
LINK_VERSION	0
ITEM_COUNT	Number of units on the system
ITEM_WIDTH	3

Each peripheral information entry has the following format:

Word	Description
0	Unit information
	[47: 8] = Peripheral type
	[39: 8] = Peripheral subtype
	[31: 9] = Density
	[22: 1] = Saved
	[21: 1] = Unit reserved
	[20: 1] = Unit reserved for maintenance
	[19: 1] = Ready
	[18: 1] = Available
	[17: 1] = No path
	[16: 1] = Mirroring
	[15: 1] = Caching
	[14: 1] = Cache mode
	[13:14] = Not used
1	Unit device number

continued

continued

Word	Description
2	Path link

The path LINK word has the following values:

LINK_TYPE	18
LINK_VERSION	0
LINK_COUNT	Number of paths to the unit
ITEM_WIDTH	3

The entry for each path has the following format:

Entry Word	Description
0	Path information [47: 1] = No path to unit through this control unit (CTL). The CTL is reserved through a UR (Unit Reserved) system command, in maintenance mode due to a UR MAINT command, freed, or has no path. [46:43] = Not used [3:4] = Subsystem protocol 0 = Hex 1 = EBCDIC 2 = Decimal 3 = ASCII
1	CTL number
2	Firmware information [47: 8] = Firmware length [39:40] = Firmware identification

LINK [5] + 4 : Extended Pack Information (A 17 Systems)

This LINK has the same format as LINK [5] + 4 under "Link Descriptions for HDU Systems" in this section.

LINK [5] + 5 : Path Group Information (A 17 Systems)

The LINK word has these values:

LINK_TYPE	25
LINK_VERSION	0
ITEM_COUNT	Number of path groups on the system
ITEM_WIDTH	2

The path group entry has the following information:

Entry Word	Description
0	Unit link or control unit link
1	Control link or unused

The unit LINK word has these values:

LINK_TYPE	26
LINK_VERSION	0
ITEM_COUNT	Number of units in the path group
ITEM_WIDTH	2

The unit entry has the following information:

Entry Word	Description
0	Unit information
	[47: 8] = Peripheral type
	[39: 1] = Freed
	[38:39] = Not used
1	Unit device number

The control unit LINK word has these values:

LINK_TYPE	27
LINK_VERSION	0
ITEM_COUNT	1
ITEM_WIDTH	5

The control unit entries are identical in format and information to the control entries. The control LINK word has these values:

LINK_TYPE	28
LINK_VERSION	0
ITEM_COUNT	Number of controls in the path group
ITEM_WIDTH	5

Each control and control unit entry has the following information:

Entry Word	Description
0	Control information
	[47: 8] = Control type
	[39: 8] = Firmware format
	[31: 1] = Freed

continued

continued

Entry Word	Description
	[30: 1] = Unit reserved for maintenance
	[29: 1] = Path broken
	[28: 1] = Path reserved
	[27: 1] = No path
	[26: 1] = Online
	[25: 1] = In use
	[24: 1] = Control unit
	[23:24] = Not used
1	External device number
2	Firmware level
3	Path status
	[23: 4] = Port selected ordinal port number; 0=none
	[19: 1] = Fourth port usable
	[18: 1] = Third port usable
	[17: 1] = Second port usable
	[16: 1] = First port usable
	[15:16] = Identification of fourth port (lowest priority)
4	Path status word 2
	[47:16] = Identification of third port
	[31:16] = Identification of second port
	[15:16] = Identification of first port (highest priority)

LINK [5] + 7 : Task Control Processor Information (A 17 Systems)

The task control processor LINK has the following values:

LINK_TYPE	29
LINK_VERSION	0
ITEM_COUNT	1
ITEM_WIDTH	3

The task control processor entry has the following format:

Entry Word	Description
0	Distinguished TCP mask
1	TCP identification word 1

continued

continued

Entry Word	Description
2	TCP identification word 2

See Tables 12–22 through 12–25 for information on peripheral types.

LINK [5] + 9 : Continuation Information (A 17 Systems)

The continuation information LINK indicates that the hardware configuration is described by multiple log entries. If the hardware configuration information fits in a single log entry, then the continuation information LINK word is 0. Otherwise, this LINK has the following values:

LINK_TYPE	207
LINK_VERSION	4
ITEM_COUNT	1
ITEM_WIDTH	1

The continuation information LINK points to a word that has one of the following values:

Value	Meaning
1	First of multiple log entries
2	An intermediate log entry
3	Last of multiple log entries

When the hardware configuration is described by multiple log entries, each log entry has the same structure. That is, words 0 through 9 appear as described in Table 12–30, "Major Type 2–Minor Type 17," and the LINK words appear at their usual offsets beginning with Word 10. However, if the information pointed to by a particular LINK cannot fit in the current log entry, that LINK word is zero and the information appears in one of the other log entries.

Link Descriptions for A 16 Systems

LINK [5] + 0 : Data Processor Information (A 16 Systems)

This LINK has the following values:

LINK_TYPE	2
LINK_VERSION	1
ITEM_COUNT	Number of data processors
ITEM_WIDTH	7 (words)

Each data processor entry has the following format:

Entry Word	Description
0	PROCESSOR_NUMBER
1	PROCESSOR_TYPE 21 = A 16
2	DOMAIN_INFORMATION [47:16] (Reserved field) [31: 8] Domain number in which requestor resides [23: 8] Requestor number [15: 8] Requestor status 0 = Undefined 1 = Ready 2 = Saved 3 = Unavailable [7: 7] Requestor status subtype If requestor status = saved, 1 = Not in use 2 = User saved 3 = MCP inhibit If requestor status = unavailable, field is not used For other requestor status values, this field is 0 [0: 1] Corrective action can be taken 0 = False 1 = True
3	PROCESSOR_FIRMWARE_LEVEL [39: 8] Stored logic level major field [31:16] Stored logic level minor field [15:16] Creation date: constructed as (year – 1970) * 1000 + day_of_year
4	PROCESSOR_IOP_MASK I/O processors that this data processor can see; bit *i* is ON if this data processor can see I/O processor number *i*.
5	(Reserved word)
6	PROCESSOR_HARDWARE_LEVEL [39: 4] SC level field [35:12] ERL minor field

continued

continued

Entry Word	Description
	[23: 8] ERL major field
	[15: 4] EMODE_LEVEL:
	3 = GAMMA
	[7: 8] MACHINE_TYPE:
	21 = A 16

LINK [5] + 1 : I/O Processor Information (A 16 Systems)

Note: *The format of I/O information logged for A 16 systems is changing in a future release. The new format will be the same as that described under "LINK [5] + 1 : Version 2 I/O Processor Information (IOM and RMM Systems)" later in this section. A program can determine which format is being used by inspecting the LINK_VERSION value, which will change from 0 to 2 when the format changes.*

A warning is now displayed whenever a process uses the LOG_GET or LOG_READ procedures of the SDASUPPORT library to read a Hardware Configuration log entry. Programs that use the SDASUPPORT library can suppress this warning message by first invoking the LOG_SELECT procedure and passing a value of 1021 to the STYPE parameter.

This LINK has the following values:

LINK_TYPE	21
LINK_VERSION	1
ITEM_COUNT	Number of IOMs
ITEM_WIDTH	5 (words)

Each I/O processor entry has the following format:

Entry Word	Description
0	IOM information
	[47: 8] IOM domain number
	[39: 1] IOU firmware valid
	[38: 1] This IOM contains the distinguished IOU.
	[37: 1] IOU is ready.
	[36: 4] IOU status (when not ready)
	1 = Not in use
	2 = User saved
	3 = MCP inhibited

continued

continued

Entry Word	Description
	4 = Unavailable
	[32:14] Not used
	[18: 8] IOM status
	0 = Freed
	1 = In use
	2 = Saved
	3 = Unavailable
	[10: 7] IOM status subtype
	If IOM Status field = 2 (saved):
	1 = Not in use
	2 = User saved
	3 = MCP inhibit
	If Status field = 3 (unavailable):
	1 = Domain powered off
	2 = Module not present
	3 = Module uninitialized
	4 = CSD interface unavailable
	5 = CSD interface saved
	6 = Memory interface unit (MIU) failed to initialize.
	7 = Peripheral configuration diagrams (PCDs) do not match.
	Otherwise, this field is not used.
	[3: 4] IOM requestor number
1	IOU firmware word 1
2	IOU firmware word 2
3	DTU LINK
4	Port LINK

The DTU LINK word has these values:

LINK_TYPE	22
LINK_VERSION	1
ITEM_COUNT	Number of DTUs on the IOM
ITEM_WIDTH	3 (words)

Each DTU entry has the following format:

Entry Word	Description
0	DTU information
	[47: 1] = DTU is reserved.
	[46: 1] = In use
	[45: 1] = Dead
	[44: 1] = Out of service
	[43: 1] = Online
	[42: 1] = Unavailable
	[41:26] = Not used
	[15:16] = DTU identification
1	DTU firmware word 1
2	DTU firmware word 2

The port LINK word has these values:

LINK_TYPE	23
LINK_VERSION	1
ITEM_COUNT	Number of ports on the IOM
ITEM_WIDTH	3 (words)

Each port entry has the following format:

Entry Word	Description
0	Port information
	[47: 1] = Port is reserved
	[46: 1] = In use
	[45: 1] = Dead
	[44: 1] = Out of service
	[43: 1] = Online
	[42: 1] = Unavailable
	[41: 4] = Subsystem protocol
	0 = SCSI maintenance bus (SMB)
	1 = Message level interface (MLI)
	2 = Intelligent peripheral interface (IPI)
	3 = Small-computer system interface (SCSI)
	[37:22] = Not used
	[15:16] = Port identification
1	Port firmware word 1

continued

continued

Entry Word	Description
2	Port firmware word 2

LINK [5] + 2 : Memory Information (A 16 Systems)

This LINK has the following values:

LINK_TYPE	10
LINK_VERSION	2
ITEM_COUNT	1
ITEM_WIDTH	17 (words)

The memory entry has the following format:

Entry Word	Description
0	MSM MASK
1	MSM 0 information
	[47: 4] Domain number associated with the MSM
	[43: 8] Mask of requestors connected to the MSM
	[35:20] (Reserved field)
	[15: 8] MSM status
	0 = Undefined
	1 = Ready
	2 = Saved
	3 = Unavailable
	[7: 7] MSM status subtype
	If Status field = 2 (saved)
	1 = Not in use
	2 = User saved
	3 = MCP inhibit
	If Status field = 3 (unavailable), this field is not used.
	Otherwise, this field stores a 0.
	[0: 1] Corrective action can be taken
2	MSM 0 engineering release level (ERL)
	[47: 8] (Reserved field)
	[39: 4] Software compatibility level
	[35:12] Engineering release level cycle
	[23: 8] Engineering release level mark

continued

continued

Entry Word	Description
	[15:16] (Reserved field)
3	MSU 0 info and SYSTEM requestor mask
	[47: 8] SYSTEM 1 requestor mask
	[39: 8] SYSTEM 0 requestor mask
	[31: 8] (Reserved field)
	[23: 4] Model number of MSU 5
	[19: 4] Model number of MSU 4
	[15: 4] Model number of MSU 3
	[11: 4] Model number of MSU 2
	[7: 4] Model number of MSU 1
	[3: 4] Model number of MSU 0
	MSU model numbers
	1 = Old style SU, 8 megaword
	(probably type used during debug)
	2 = New style SU, 8 megawords per quadrant,
	maximum 32 megawords total
4	MSM 0 MSU quadrant mask
	[47:24] Present quadrant mask
	[47: 1] MSU 5 quadrant 3
	[46: 1] MSU 5 quadrant 2
	[45: 1] MSU 5 quadrant 1
	[44: 1] MSU 5 quadrant 0
	[43: 1] MSU 4 quadrant 3
	.
	.
	.
	[28: 1] MSU 1 quadrant 0
	[27: 1] MSU 0 quadrant 3
	[26: 1] MSU 0 quadrant 2
	[25: 1] MSU 0 quadrant 1
	[24: 1] MSU 0 quadrant 0
	[23:24] In use quadrant mask
	[23: 1] MSU 5 quadrant 3
	[22: 1] MSU 5 quadrant 2

continued

continued

Entry Word	Description
	[21: 1] MSU 5 quadrant 1
	[20: 1] MSU 5 quadrant 0
	[19: 1] MSU 4 quadrant 3
	.
	.
	.
	[4: 1] MSU 1 quadrant 0
	[3: 1] MSU 0 quadrant 3
	[2: 1] MSU 0 quadrant 2
	[1: 1] MSU 0 quadrant 1
	[0: 1] MSU 0 quadrant 0
5	MSU 0 quadrant status
	Each model II MSUs can contain up to four 8 megaword quadrants. Since a model I MSU contains only 8 megawords, it is represented in this structure as having quadrant 0 present. All other quadrants are marked not present.
	Each of the four-bit quadrant status fields in words 5 and 11 has the following format:
	[3:1] Single-bit logging disabled
	[2:3] Status
	1 = Not in use
	2 = MCP inhibit
	The following are the quadrant status fields for each MSU.
	[47:16] MSU 2
	[47: 4] MSU 2 quadrant 3 status
	[43: 4] MSU 2 quadrant 2 status
	[39: 4] MSU 2 quadrant 1 status
	[35: 4] MSU 2 quadrant 0 status
	[31:16] MSU 1
	[31: 4] MSU 1 quadrant 3 status
	[27: 4] MSU 1 quadrant 2 status
	[23: 4] MSU 1 quadrant 1 status
	[19: 4] MSU 1 quadrant 0 status
	[15:16] MSU 0
	[15: 4] MSU 0 quadrant 3 status

continued

continued

Entry Word	Description
	[11: 4] MSU 0 quadrant 2 status
	[7: 4] MSU 0 quadrant 1 status
	[3: 4] MSU 0 quadrant 0 status
6	MSU 0 quadrant status, continued
	The comments for Word 5 also apply to these words.
	[47:16] MSU 5
	[47: 4] MSU 5 quadrant 3 status
	[43: 4] MSU 5 quadrant 2 status
	[39: 4] MSU 5 quadrant 1 status
	[35: 4] MSU 5 quadrant 0 status
	[31:16] MSU 4
	[31: 4] MSU 4 quadrant 3 status
	[27: 4] MSU 4 quadrant 2 status
	[23: 4] MSU 4 quadrant 1 status
	[19: 4] MSU 4 quadrant 0 status
	[15:16] MSU 3
	[15: 4] MSU 3 quadrant 3 status
	[11: 4] MSU 3 quadrant 2 status
	[7: 4] MSU 3 quadrant 1 status
	[3: 4] MSU 3 quadrant 0 status
7	MSM MSU single-bit logging disable mask
	[47:24] (Reserved field)
	[23:24] Quadrant mask
	For each bit in the quadrant mask,
	0 = Single-bit logging enabled
	1 = Single-bit logging disabled
	The following are the quadrants referred to by each bit:
	[23: 1] MSU 5 quadrant 3
	[22: 1] MSU 5 quadrant 2
	[21: 1] MSU 5 quadrant 1
	[20: 1] MSU 5 quadrant 0

continued

continued

Entry Word	Description
	[19: 1] MSU 4 quadrant 3
	.
	.
	.
	[4: 1] MSU 1 quadrant 0
	[3: 1] MSU 0 quadrant 3
	[2: 1] MSU 0 quadrant 2
	[1: 1] MSU 0 quadrant 1
	[0: 1] MSU 0 quadrant 0
8 – 13	MSU 1 information
	These words parallel the format of words 1 through 6.
14	MSM 0 To-be-saved mask
	[47:24] (Reserved field)
	[23: 1] MSU 5 quadrant 3
	[22: 1] MSU 5 quadrant 2
	[21: 1] MSU 5 quadrant 1
	[20: 1] MSU 5 quadrant 0
	[19: 1] MSU 4 quadrant 3
	.
	.
	.
	[4: 1] MSU 1 quadrant 0
	[3: 1] MSU 0 quadrant 3
	[2: 1] MSU 0 quadrant 2
	[1: 1] MSU 0 quadrant 1
	[0: 1] MSU 0 quadrant 0
14	MSM 1 To-be-saved mask
	[47:24] (Reserved field)
	[23: 1] MSU 5 quadrant 3
	[22: 1] MSU 5 quadrant 2
	[21: 1] MSU 5 quadrant 1
	[20: 1] MSU 5 quadrant 0

continued

continued

Entry Word	Description
	[19: 1] MSU 4 quadrant 3
	.
	.
	.
	[4: 1] MSU 1 quadrant 0
	[3: 1] MSU 0 quadrant 3
	[2: 1] MSU 0 quadrant 2
	[1: 1] MSU 0 quadrant 1
	[0: 1] MSU 0 quadrant 0

LINK [5] + 3 : Peripheral Information (A 16 Systems)

This LINK has the same format as LINK [5] + 3 under "Link Descriptions for A 17 Systems" in this section.

LINK [5] + 4 : Extended Pack Information (A 16 Systems)

This LINK has the same format as LINK [5] + 4 under "Link Descriptions for HDU Systems" in this section.

LINK [5] + 5 : Path Group Information (A 16 Systems)

This LINK has the same format as LINK [5] + 5 under "Link Descriptions for A 17 Systems" in this section.

LINK [5] + 7 : Task Control Processor Information (A 16 Systems)

This LINK has the same format as LINK [5] + 7 under "Link Descriptions for A 17 Systems" in this section.

LINK [5] + 8 : Domain Status Information (A 16 Systems)

This LINK has the following values:

LINK_TYPE	30
LINK_VERSION	0
ITEM_COUNT	1
ITEM_WIDTH	10 (words)

The domain entry has the following format:

Entry Word	Description
0–7	Requestor status information
	[47:16] Requestor type
	4'00' = Unknown module
	4'E1' = CPM
	4'E2' = IOM
	4'E3' = RMM
	[31: 8] Domain number requestor resides in
	[23: 8] Requestor number
	[15: 8] Requestor status
	0 = Undefined
	1 = Ready
	2 = saved
	3 = unavailable
	[7: 7] Requestor status subtype
	If requestor status = saved
	1 = Not in use
	2 = User saved
	3 = MCP inhibit
	If requestor status = unavailable, field is not used
	Otherwise, this field stores a 0
	[0: 1] Corrective action can be taken
8–9	MSM status information
	[47: 4] Domain number associated with the MSM
	[43: 8] Mask of requestors connected to the MSM
	[35:20] (Reserved field)
	[15: 8] MSM status
	0 = Undefined

continued

continued

Entry Word	Description
	1 = Ready
	2 = Saved
	3 = Unavailable
	[7: 7] MSM status subtype
	If status = saved
	1 = Not in use
	2 = User saved
	3 = MCP inhibit
	If status = unavailable, this field is not used
	Otherwise, this field stores a 0
	[0: 1] Corrective action can be taken

LINK [5] + 1 : Version 2 I/O Processor Information (IOM and RMM Systems)

Note: *The format of I/O information logged for IOM and RMM systems is changing in a future release. A program can determine which format is being used by inspecting the LINK_VERSION value, which will change from 0 to 2 when the format changes. The following pages describe the version 2 format of I/O processor information for IOM and RMM systems.*

Though the following description applies to both IOM and RMM systems, the IOM terminology is used throughout. For A 17 systems, references to IOMs should be interpreted as RMMs, and references to IOUs should be interpreted as IOPs.

This LINK has the following values:

LINK_TYPE	21
LINK_VERSION	2
ITEM_COUNT	Number of IOMs
ITEM_WIDTH	5 (words)

Each I/O processor entry has the following format:

Entry Word	Description
0	IOM information
	[47: 8] IOM domain number
	[39:21] (Reserved field)
	[18: 8] IOM status
	0 = Freed

continued

continued

Entry Word	Description
	1 = In use
	2 = Saved
	3 = Unavailable
	[10: 7] IOM status subtype
	If IOM Status field = 2 (saved):
	1 = Not in use
	2 = User saved
	3 = MCP inhibit
	If IOM Status field = 3 (unavailable):
	1 = Domain powered off
	2 = Module not present
	3 = Module uninitialized
	4 = CSD interface unavailable
	5 = CSD interface saved
	6 = Memory interface unit (MIU) failed to initialize.
	7 = Peripheral configuration diagrams (PCDs) do not match.
	8 = No peripheral configuration diagram (PCD)
	9 = No master clock
	10 = No REM in partition
	Otherwise, this field is not used.
	[3: 4] IOM requestor number
1	(Reserved word)
2	IOU LINK
3	DTU LINK
4	Port LINK

The IOU LINK word has these values:

LINK_TYPE	31
LINK_VERSION	0
ITEM_COUNT	Number of IOUs in the IOM
ITEM_WIDTH	3 (words)

Each IOU entry has the following format:

Entry Word	Description
0	IOU information
	[47: 1] = IOU is reserved.
	[46: 1] = In use
	[45: 1] = Dead
	[44: 1] = Out of service
	[43: 1] = Online
	[42: 1] = Unavailable
	[41: 1] = Distinguished
	[40: 1] = Inactive
	[39:20] = (Reserved field)
	[19:04] = IOM board number
	[15:16] = IOU number
1	IOU firmware word 1
2	IOU firmware word 2

The DTU LINK word has these values:

LINK_TYPE	22
LINK_VERSION	2
ITEM_COUNT	Number of DTUs on the IOM
ITEM_WIDTH	3 (words)

Each DTU entry has the following format:

Entry Word	Description
0	DTU information
	[47: 1] = DTU is reserved.
	[46: 1] = In use
	[45: 1] = Dead
	[44: 1] = Out of service
	[43: 1] = Online
	[42: 1] = Unavailable
	[41:22] = (Reserved field)
	[19:04] = IOM board number
	[15:16] = DTU identification
1	DTU firmware word 1
2	DTU firmware word 2

continued

continued

The port LINK word has these values:

LINK_TYPE	23
LINK_VERSION	2
ITEM_COUNT	Number of ports on the IOM
ITEM_WIDTH	3 (words)

Each port entry has the following format:

Entry Word	Description
0	Port information [47: 1] = Port is reserved. [46: 1] = In use [45: 1] = Dead [44: 1] = Out of service [43: 1] = Online [42: 1] = Unavailable [41: 4] = Subsystem protocol 0 = SCSI maintenance bus (SMB) 1 = Message-level interface (MLI) 2 = Intelligent peripheral interface (IPI) 3 = Small-computer system interface (SCSI) [37:18] = (Reserved field) [19: 4] = IOM board number [15:16] = Port identification
1	Port firmware word 1
2	Port firmware word 2

This log entry uses LINKs to store information in a tree structure. One LINK is at the top of the tree; this LINK describes a region that contains eight LINKs. Each of these LINKs describes a region that contains information about the software configuration. In some cases, these regions contain LINKs to other regions. The fixed part of the software configuration log entry consists of the information in Table 12–31, "Major Type 2–Minor Type 18."

Table 12–31. Major Type 2–Minor Type 18

Minor Type = 18 Software Configuration (SOFTWARE_CONFIG)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	General information. This word has the same layout as Word 4 of the Hardware Configuration log entry. For a description of this word, refer to Table 12–30, "Major Type 2–Minor Type 17."
5	LINK to the software information

All the actual software configuration information resides in the variable part of the software configuration entry. Accessing any of this information requires that the LINK at Word 5 be examined. For further information about the LINKs used in this log entry, refer to "Format of Hardware and Software Configuration LINK Words" earlier in this section.

The LINK at Word 5 contains the following values:

LINK_TYPE	1
LINK_VERSION	0
ITEM_COUNT	10 (LINKs)
ITEM_WIDTH	1

The region described by the LINK at Word 5 has the following format:

Offset	Contents	LINK_TYPE
0	LINK to general MCP information	2
1	LINK to support library information	3
2	LINK to disk location information	4
3	LINK to substitute backup information	5
4	LINK to ASD information. Prior to the Mark 3.9 system software release, this was a LINK to memory management parameter information	6

continued

continued

Offset	Contents	LINK_TYPE
5	Not used. Prior to Mark 3.9, this was a LINK to subsystem information	7
6	LINK to queue information	8
7	Not used. Prior to Mark 3.9, this was a LINK to SWAPPER parameter information	9
8	LINK to SEGARRAY start information	16
9	LINK to security information	18

The values contained in each LINK in the preceding table and the format of the information corresponding to each LINK are given in the following paragraphs.

All LINKs that describe an item in length-string form have the following values:

LINK_TYPE	254
LINK_VERSION	0
ITEM_COUNT	1 (name)
ITEM_WIDTH	Number of characters used to store this string

All LINKs that describe a name in standard form have the following values:

LINK_TYPE	255
LINK_VERSION	0
ITEM_COUNT	1 (name)
ITEM_WIDTH	Number of characters used to store this name

LINK [5] + 0 : General MCP Information

This LINK has the following values:

LINK_TYPE	2
LINK_VERSION	1
ITEM_COUNT	32
ITEM_WIDTH	0 (heterogeneous)

This region has the following format:

Region Word	Description
0	LINK to halt/load and backup unit information (See "Halt/Load and Backup MCP Unit Information" following this table.)
1	MCP version [47:16] MARK_LEVEL [31:16] CYCLE [15:16] PATCH_NUMBER
2	MCP compilation date and time (timestamp) [47:16] Date (Julian - 700000) [35:32] Time (2.4-microsecond increments) DIV 16
3	MCP "compiled by" information [23: 8] MARK_LEVEL (of compiler) [9:10] CYCLE (of compiler)
4	System serial number
5	LINK-to-MCP compile-time options LINK_TYPE = 14 LINK_VERSION = 0 ITEM_COUNT = 1 ITEM_WIDTH = Number of characters Each option is separated from the next by a null byte (48 "00"), and the list is terminated by two null bytes in succession.
6	LINK to GROUP ID (length-string form)
7	CATALOG_LEVEL
8	OPTIONS word The option numbers correspond to the bit numbers in this word.
9	LINK to NIF name (standard form)
10	LINK to CONFIGURATION FILE name (standard form)
11	LINK to SUPERVISOR name (standard form)
12	LINK to SYSTEM INTRINSICS name (standard form)
13	System balancing parameters [0: 1] IDLE [1: 1] IOFINISH [2: 1] WAITING [3: 1] QUEUEEMPTY
14	Compiler target

continued

continued

Region Word	Description
	[13: 1] Level indicator
	[12: 8] If [13: 1] is 1, then field [12: 8] contains the target level. Otherwise, field [12: 8] contains the target machine ID.
	[4: 5] TARGET-CLASS
	1 = Level 0
	3 = Level 1
15	Number of characters in SYSTEMLANGUAGE name
16–18	SYSTEMLANGUAGE name
19	Disk resource control (DRC) system status
	0 = Inactive
	1 = Will be started after the next halt/load
	2 = Initializing
	3 = Active
	4 = Terminating
20	Number of predefined time zones
21	Encoded time zone information word.
	[47: 1] Time zone configured
	0 = Time zone not configured.
	1 = Time zone is configured.
	[46: 1] Time zone offset direction
	0 = Local time is Universal Time (UT) – offset.
	1 = Local time is UT + offset.
	[45: 2] Not used
	[43: 1] Custom time zone
	0 = Predefined time zone. Words 22–28 are not valid.
	1 = Custom time zone. Words 22–28 store further information.
	[42: 3] If custom time zone, length of abbreviation (1–6)
	[39: 8] Hours offset (0–24)
	[31: 8] Minutes offset (0–59)
	[23: 8] Military-style mnemonic (in EBCDIC) or zero
	[15: 4] Not used
	[11:12] If predefined time zone, its number
22	Custom time zone abbreviation (left-justified) Valid only if Word 21, [43: 1] = 1.

continued

continued

Region Word	Description
23–28	Custom time zone in substandard form Valid only if Word 21, [43: 1] = 1.
29	LINK to hostname (in standard form)
30	Current CCSVERSION
	[15:16] CCSVERSION number
31	LINK to current CONVENTION (length-string form)

Halt/Load and Backup MCP Unit Information

This region is used to list each unit that is either a halt/load unit or an MCP backup unit, and the name of the MCP code file.

The LINK to halt/load and backup MCP unit information has the following values:

LINK_TYPE	15
LINK_VERSION	0
ITEM_COUNT	1
ITEM_WIDTH	5 (words)

Each entry in this region has the following format:

Entry Word	Description
0	Device number
1	Device number of BACKUP_UNIT_1 (not applicable if Word 3 is 0)
2	Device number of BACKUP_UNIT_2 (not applicable if Word 3 is 0)
3	Halt/load processor mask (bit SET for each processor that uses this unit as its halt/load unit)
4	LINK to name of MCP on this unit (standard form)

LINK [5] + 1 : Support Library Information

This LINK has the following values:

LINK_TYPE	3
LINK_VERSION	0
ITEM_COUNT	Number of libraries
ITEM_WIDTH	4 (words)

Each entry in this region has the following format:

Entry Word	Description
0–2	LIBRARY_ID (function name, length-string form)
3	LINK to library code file name (standard form)

LINK [5] + 2 : Disk Location Information

This LINK has the following format:

LINK_TYPE	4
LINK_VERSION	0
ITEM_COUNT	10
ITEM_WIDTH	1 (words)

Each entry in this region has the following format:

Entry Word	Description
0	LINK to CATALOG family (length-string form)
1	LINK to JOBS family (length-string form)
2	LINK to USERDATA family (length-string form)
3	LINK to BACKUP family (length-string form)
4	LINK to LOG family (length-string form)
5	LINK to overlay family (length-string form)
6	LINK to extra overlay families
7	LINK to DPFILES family (length-string form)
8	LINK to SORT family (length-string form)
9	LINK to IPFILES family (length-string form)

The LINK to extra overlay families has the following values:

LINK_TYPE	11
LINK_VERSION	0
ITEM_COUNT	Number of extra overlay families
ITEM_WIDTH	4 (words)

Each entry has the following format:

Entry Word	Description
0	This word stores a 0. On logs generated prior to the Mark 3.9 system software release, this word stores an ASN number. A value of 0 indicates global memory. Any other value indicates the local memory subsystem with the corresponding number.
1–3	Overlay family (length-string form)

LINK [5] + 3 : Substitute Backup Information

This LINK has the following values:

LINK_TYPE	5
LINK_VERSION	0
ITEM_COUNT	6
ITEM_WIDTH	1 (word)

Each entry in this region has the following format:

Entry Word	Description
0	SB information for DISK
1	SB information for PACK
2	SB information for TAPE
3	SB information for PETAPE
4	SB information for TAPE9
5	SB information for TAPE7

Each of the preceding six entry words has the following format:

Field	Meaning
[47: 4]	First device
[43: 4]	Second device
[39: 4]	Third device
[35: 4]	Fourth device
[31: 4]	Fifth device
[27: 4]	Sixth device
[23: 4]	Seventh device
[19:20]	Reserved

The devices are given the following code numbers:

Device Number	Device Type
1	DISK
2	PACK
3	TAPE
4	PETAPE
5	TAPE9
6	TAPE7
7	DLBACKUP

LINK [5] + 4 : Memory Management Parameters

This LINK has the following values:

LINK_TYPE	6
LINK_VERSION	1
ITEM_COUNT	9
ITEM_WIDTH	1 (word)

Each entry in this region has the following format:

Entry Word	Description
0	OLAYGOAL (percent)
1	AVAILMIN (percent)
2	FACTOR (percent)
3	PRIORITY (percent)
4	BUFFERGOAL. The absolute value of this word is the current BUFFERGOAL value. If the word contains a negative value, then the system is using the default BUFFERGOAL value. If the word contains a positive value, then the system is using a BUFFERGOAL value specified by the configuration file or by an SF (Set Factor) system command.
5	Current ASD factor
6	Next ASD factor
7	Overlay saturation
8	Overlay change

LINK [5] + 5 : Subsystem Information

This LINK is no longer used. In system logs generated prior to the Mark 3.9 system software release, this LINK has the following values:

LINK_TYPE	7
LINK_VERSION	0
ITEM_COUNT	Number of subsystems
ITEM_WIDTH	4 (words)

Each entry in this region has the following format:

Entry Word	Description
0–2	Subsystem name, length-string
3	Mask of ASNs that are in this subsystem 0 = GLOBAL ASN 1 = LOCAL 1 2 = LOCAL 2 . . .

LINK [5] + 6 : Queue Information This LINK has the following values:

LINK_TYPE	8
LINK_VERSION	0
ITEM_COUNT	3
ITEM_WIDTH	1 (words)

Each entry in this region has the following format:

Entry Word	Description
0	GLOBAL_MIXLIMIT
1	DEFAULT_QUEUE
2	LINK to individual queue information

The LINK to the individual queue information region has the following values:

LINK_TYPE	12
LINK_VERSION	0
ITEM_COUNT	Number of queues
ITEM_WIDTH	39 (words)

Each entry for an individual queue has the following information. Note than an explicit value of 0 for an integer or real attribute is represented by 4'800000000000'. Because of this representation, you can determine whether an attribute has been set by comparing the word that describes that attribute with 0.

Entry Word	Description
0	Queue number
1	MIXLIMIT
2	TASKLIMIT
3	TURNAROUND (seconds)
4	SUBSPACES (Not used on Mark 3.9 or later releases)
5–7	Family to substitute for, in length-string form. For example, for a FAMILY statement of "DISK = MCPMAST OTHERWISE SYS33", these words store "DISK".
8–10	Family to substitute, in length-string form. For example, for a FAMILY statement of "DISK = MCPMAST OTHERWISE SYS33", these words store "MCPMAST".
11–13	Alternate family to substitute, in length-string form. For example, for a FAMILY statement of "DISK = MCPMAST OTHERWISE SYS33", these words store "SYS33".
14–16	SUBSYSTEM (length-string form). These words are not used on Mark 3.9 or later releases.
17	DEFAULT PRIORITY
18	DEFAULT IOTIME (seconds)
19	DEFAULT PROCESSTIME (seconds)
20	DEFAULT LINES
21	DEFAULT CARDS
22	DEFAULT WAITLIMIT (seconds)
23	DEFAULT DISKLIMIT (segments)
24	DEFAULT ELAPSEDLIMIT (seconds)
25	LIMIT PRIORITY
26	LIMIT IOTIME (seconds)
27	LIMIT PROCESSTIME (seconds)
28	LIMIT LINES

continued

continued

Entry Word	Description
29	LIMIT CARDS
30	LIMIT WAITLIMIT (seconds)
31	LIMIT DISKLIMIT (segments)
32	LIMIT ELAPSEDLIMIT (seconds)
33	Not used
34	LIMIT TAPE7
35	LIMIT TAPE9
36	LIMIT TAPEPE
37	DEFAULT SAVEMEMORYLIMIT
38	LIMIT SAVEMEMORYLIMIT

LINK [5] + 7 : SWAPPER Parameters

This LINK is no longer used. In system logs generated prior to the Mark 3.9 system software release, this LINK has the following values:

LINK_TYPE	9
LINK_VERSION	0
ITEM_COUNT	Number of boxes with SWAPPER parameters
ITEM_WIDTH	18 (words)

Each entry in this region has the following format:

Entry Word	Description
0	ASN number 0 = GLOBAL ASN 1 = LOCAL 1 2 = LOCAL 2 . . .
1	SWAPPERSTATUS 0 = NOT RUNNING 1 = RUNNING
2	CORESIZE (990-word slots)
3	EXPMAXCORE (990-word slots)
4	EXPRESSRESERVE (990-word slots)
5	EXPMAXTIME (seconds)
6	IOBIAS (percent)

continued

continued

Entry Word	Description
7	MAXCORE (990-word slots)
8	MAXIOSIZE (990-word slots)
9	MAXSLICENUMBER
10	MEMORYBIAS (percent)
11	MINCHUNKSIZE (990-word slots)
12	MINTIMESLICE (seconds)
13	PRIORITYBIAS (percent)
14	RATIO
15	UTILIZATIONBIAS (percent)
16	[47:47] Reserved [0: 1] 0 = NOSWAPTRANSTATE 1 = SWAPTRANSTATE
17	LINK to SWAPPER family names used by the SWAPPER running in this ASN

The LINK to the SWAPPER family names region has the following format:

LINK_TYPE	13
LINK_VERSION	0
ITEM_COUNT	Number of SWAPPER families
ITEM_WIDTH	3 (words)

Each entry in this region has the following format:

Entry Word	Description
0–2	SWAPPER family name (length-string form)

LINK [5] + 8 : SEGARRAYSTART Information

This LINK has the following values:

LINK_TYPE	16
LINK_VERSION	0
ITEM_COUNT	2 In system logs generated prior to the Mark 3.9 system software release, the ITEM_COUNT equals the number of ASNs in the system + 1.
ITEM_WIDTH	1 (word)

Each entry in this region has the following format:

Entry Word	Description
0	The maximum size of a LONG array that can be overlayed by the system. Used as a limit for the SEGARRAYSTART setting.
1	SEGARRAYSTART value

In system logs generated prior to the Mark 3.9 system software release, each entry in this region has the following format:

Entry Word	Description
1	The maximum array size that can be overlayed by the system. Used as a limit for all the SEGARRAYSTART settings.
2–x	The setting for corresponding ASN (Word 1 = global, Word 2 = local 1, Word 3 = local 3, and so on)

LINK [5] + 9 : Security Information

This LINK has the following values:

LINK_TYPE	18
LINK_VERSION	0
ITEM_COUNT	2
ITEM_WIDTH	0 (Heterogeneous)

Each entry in this region has the following format:

Region Word	Description
0	Security options word [00:01] Security administrator authorized [01:01] Program dump filtering [02:01] Mandatory disk scrubbing [03:01] Nonuser file security 0 = PUBLIC 1 = PRIVATE [04:01] Restrict all hosts [05:01] Usercoded backup [06:01] (Reserved) [07:01] Tape scrubbing [11:04] Security class 0 = Class U (unspecified) 1 = Class S0 2 = Class S1

continued

continued

Region Word	Description
	3 = Class S2
	[13:02] Current tape security
	0 = NONE
	1 = AUTOMATIC
	[15:02] Future tape security
	[18:03] Password management
	0 = MINIMAL
	1 = GENERATED
	[19: 1] S1 miscellaneous restrictions
	[20: 1] S2 miscellaneous restrictions
	[23:03] Reserved for future use
	[27:04] LOGONATTEMPTS value
	[30:03] LAISSEZFILE (CANDE)
	[31:01] Reserved for future use
	[32:01] DIALLOGIN (CANDE)
	[33:01] ALLLOGIN (CANDE)
	[34:01] SECDIALIN (CANDE)
	[35:01] SECPSEUDO (CANDE)
	[36:01] SECALL (CANDE)
	[37:01] USECOMSPRIV (CANDE)
	[46:09] Reserve for future use
	[47:01] InfoGuard authorized
1	LINK to Log Options. This LINK has the following format:
	LINK_TYPE = 17
	LINK_VERSION = 0
	ITEM_COUNT = 1
	ITEM_WIDTH = length of log options array in words

The following log entry can appear only in system logs generated prior to the Mark 3.9 system software release.

Table 12–32. Major Type 2–Minor Type 19

Minor Type = 19 Environment Configuration (ENV_CONFIG) [Extended memory systems only]	
Words	**Description**
0–3	As described in Table 12-2, "First Four Log Entry Words"
4	Reason for log entry 1 = Operator requested (result of operator entered EC) 2 = System halt/loaded
5–51:	Information about ASNs 0 through 46 [32: 1] = Split ASN [31: 1] = Code environment specified in the configuration file [30: 7] = Number of code pages [23: 1] = Data environment specified in the configuration file [22: 7] = Number of data pages [15: 8] = Code page mask [7: 8] = Data page mask

Words 5 through 31 are used if Word 4 = 0 (A Series EMS system single-bit error).

Table 12–33. Major Type 2–Minor Type 20

Minor Type = 20 Internal Processor Errors (PROCESSORERRORS) [A Series EMS systems only]	
Words	**Description**
0–3	As described in Table 12-2, "First Four Log Entry Words"
4	What kind of error [7: 8] = Processor error code 0 = A Series EMS system single-bit error The remainder of Word 4 is reserved for future use; only field [7: 8] should be referenced.
5	General information [47: 1] = Reason 0 = Error threshold 1 = Hard correctable [34:35] = Sampling period duration (2.4-microsecond units)
6	General information [47:24] = Sample count [23:24] = Number of error words
7–30	Reserved for configuration expansion
31–end	Error words Single-bit error log with processor number in [47: 3]

Table 12–34. Major Type 2–Minor Type 21

Minor Type = 21 I/O Exception (MLIOEXCEPTION)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
Normal I/O Exceptions:	
4	MLAOVERALLVERSION; overall log entry version, currently 0 (zero)
5	MLAGENINFODESC; LINK to general information area
	[47:16] Version; currently 0 (zero)
	[31:16] Area size in words
	[15:16] Word index to the area
6	MLASTATINFODESC; LINK to statistical information area
	[47:16] Version; currently 0 (zero)
	[31:16] Area size in words
	[15:16] Word index to the area
7	MLAIORECORDDESC; LINK to I/O record area
	[47:16] Version; currently 0 (zero)
	[31:16] Area size in words
	[15:16] Word index to the area
8	MLACRAREADESC; LINK to command/result area
	[47:16] Version; currently 0 (zero)
	[31:16] Area size in words
	[15:16] Word index to the area
9	MLAXSTATUSDESC; LINK to extended status area
	[47:16] Version currently 0 (zero)
	[31:16] Area size in words
	[15:16] Word index to the area
10	MLAFILEINFODESC; LINK to file information area

continued

Table 12–34. Major Type 2–Minor Type 21 (cont.)

Minor Type = 21 I/O Exception (MLIOEXCEPTION)	
Words	**Description**
	[47:16] Version; currently 0 (zero)
	[31:16] Area size in words
	[15:16] Word index to the area
11	MLAJOBINFODESC; LINK to job information area
	[47:16] Version; currently 0 (zero)
	[31:16] Area size in words
	[15:16] Word index to the area
12–end	Variable-length information

The following text contains variable-length information as described previously.

Normal I/O Exceptions: General Information Area

The general information area referenced by Word 5 has the following format:

Word	**Description**
0	Control information
	[47: 8] MLALOGSTATUS
	0 = MLANOTSETUP
	1 = MLANORMAL
	2 = MLAXSTATUSONLY
	3 = MLAINLINE (for EMS systems only)
	4 = MLAERRORIOCB (for EMS systems only)
	[39: 4] MLANUMFILEITEMS
	Number of items of file information logged. File information includes the file title, the internal name, and/or the serial number.
	[35:20] Reserved
	[15: 8] MLAIOPROTOCOL
	0 = Intelligent peripheral interface (IPI) protocol
	1 = Message level interface (MLI) protocol

continued

continued

Word	Description
	2 = SCSI protocol
	[7: 8] MLASYSTEMTYPE
	For a list of the possible values, refer to the description Word 3, field [15:16] in Table 12–66, "Major Type 10–All Minor Types."
1	Device-specific information
	[47:12] MLACTLTYPE; the control type
	4'58' = 9399-H control
	4'E0' = Native SCSI disk
	[35: 8] MLALUTYPE; logical unit type
	[27:12] MLAUNITSUBTYPE; standard unit subtype
	[15:16] MLALU; logical unit number
2	I/O retry and extended status control information
	[47: 1] MLAIRRECOVERABLE
	0 = The I/O was recovered.
	1 = The I/O could not be recovered.
	[46: 1] MLARETRYMASKEDOUT
	0 = Exception was not masked by user.
	1 = Exception was masked by user.
	[45: 1] MLAMISSINGRDS
	0 = All retries are logged.
	1 = One or more retries were lost due to logging limitations. The last retry logged is the final retry. There are retries missing between the last one logged and the one preceding it.
	[44: 5] MLARETRYENTRYSIZE; size of each retry entry in words
	[39: 4] MLAXSTATUSCONTROL
	0 = MLANOXSTATUS; no extended status (error log) information is present
	1 = MLAGOODXSTATUS; extended status (error log) information is present
	2 = MLABADXSTATUS; an error occurred while attempting to obtain extended status (error log)
	3 = MLACOULDNOTLOGXSTATUS; no error occurred, but the extended status data could not be logged.
	[35:12] Unused

continued

continued

Word	Description
	[23:12] MLAXSTATUSAREASIZE; number of bytes of extended status information
	[11: 3] MLAIOPTYPE 1 = EMS 2 = RMM 4 = HDU
	[8: 9] MLANUMRETRIES; the number of retries

Words 3 through 8 of the general information area contain information from the original I/O. Their format is as follows. More information is located in entry 0 (zero) of the I/O record area.

Word	Description
3	MLAIOCW; software IOCW
4	MLAEXTU; external unit number of the unit that incurred the exception
5	MLABASEADDR; the base address of the I/O buffer
6	Stack numbers [47:24] MLAOWNERSTKNO; owner stack number [23:24] MLAINITSTKNO; initiating stack number
7	I/O length and stack numbers [47:14] MLAJOBNUM [33:14] MLATSKNUM [19:20] MLAIOUNITS; I/O length in either characters or words depending on the value in MLACW
8	MLASTIME; a 36-bit integer in units of 2.4 microseconds indicating the time of day when the MCP received the completed I/O
9	MLAFULLLRD; logical result descriptor

Normal I/O Exceptions: Statistics Area

The statistics area referenced by Word 6 has the following format:

Word	Description
0	Count of read and write operations for the logical unit [47:24] MLASTATREADERRORCOUNT; total read I/O operations to this unit since last halt/load [23:24] MLASTATWRITEERRORCOUNT; total write I/O operations to this unit since last halt/load

continued

continued

Word	Description
1	Count of I/O errors for the logical unit [47:24] HLAIOSTATREADF; total read error to this unit since last halt/load [23:24] HLAIOSTATWRITEF; total write errors to this unit since last halt/load
2	Count of bytes transferred to or from the unit since the last halt/load
3	Count of errors successfully recovered by the hardware since the last halt/load (currently not available for RMM systems). [47:24] MLASTATREADRECS; total read errors recovered by hardware since the last halt/load [47:24] MLASTATWRITERECS; total write errors recovered by hardware since the last halt/load

Normal I/O Exceptions: I/O Record Area

The I/O record area referenced by Word 7 contains a number of entries: one entry describing the original I/O plus one entry for each logged retry. The format of one entry in the I/O record area is described here. The number of retries is found in the general information area in the MLANUMRETRIES word. The size of each I/O record entry is found in the general information area in the MLARETRYENTRYSIZE word.

Word	Description
0	MLACW; hardware control word from the IOCB. [11: 1] word mode 0 = I/O lengths are in terms of characters 1 = I/O lengths are in terms of words
1	LINK to the command packet contained in the command/result area. For intelligent peripheral interface (IPI) devices, the command length includes the length bytes themselves. For a log status of MLAINLINE, only the command LINK for the original I/O is included; all retry command LINKs are zero. [47:16] Unused [47: 2] MLAPOLLDEVICETYPE; 0 = Not a polling command 1 = Poll for not ready 2 = Poll for ready [31:16] MLACOMMANDLENGTH; command length in bytes

continued

continued

Word	Description
	[15:16] MLACOMMANDOFFSET; word offset of command in command/result area
2	MLAPATHINFO1; initiating path address for EMS and HDU systems and external device numbers of the initiating path for RMM machines in the following format: [47:16] External device number of port [31:16] External device number of control [15:16] External device number of unit
3	MLAPATHINFO2; completing path address for EMS systems. For RMM systems, the external device numbers for the completing path and for HDU systems, the external device numbers of the initiating path in the following format: [47:16] External device number of port [31:16] External device number of control [15:16] External device number of unit
4	MLAIORD; hardware result word.
5	LINK to the response packet contained in the command/result area. The result length bytes for intelligent peripheral interface (IPI) results are not stored in the command/result area; therefore, the result length for these devices does not include the length bytes. [47:16] Unused [31:16] MLARESULTLENGTH; result length in bytes [15:16] MLARESULTOFFSET; word offset of result in command/result area
6	Logical RD, the XLRD exception ID, and the "actual retry" bit. [47: 1] MLAACTUALRETRY [46: 1] MLAAUTOSENSE 0 = System software is expected to issue a request sense I/O on receipt of a check condition status result. 1 = The hardware is expected to issue a request sense I/O on receipt of a check condition status result. [45: 1] MLAFIXEDBLOCKIO 0 = Not a fixed block data transfer I/O 1 = Fixed block data transfer I/O

continued

continued

Word	Description
	[44: 1] MLAROLLYOUROWN
	0 = The command was generated by System Software.
	1 = The command was generated by hardware.
	[43:16] MLAEXCEPTIONID
	[27:11] Unused
	[16:17] MLALRD
7	MLAMISCINFO1; system dependent information

Normal I/O Exceptions: Command and Result Area

Commands and results for each I/O record or retry are in this area. The I/O record area contains LINKs pointing to each command or result stored in this area. Word 1 of each I/O record area entry is a LINK to the corresponding command in the command and result area. Word 5 of each I/O record area entry is a LINK to the corresponding result in the command and result area.

Results for SCSI I/O protocol devices are logged in different formats depending on the MLAAUTOSENSE bit, indicated in Word 6 field [46: 1] of the I/O record area. The MLAAUTOSENSE bit set indicates the hardware performs the request sense on receipt of a check condition status (= 4'02'); the information stored in the command and result area includes the original status byte (4'02') followed by the request sense operation status byte (4'00') followed by up to 28 bytes of request sense data.

```
Original      Req. Sense       Req. Sense
Status        Status           Data
----------------------------------------------------
| Ø2       |   ØØ        |  <up to 28 bytes of data>|
----------------------------------------------------
```

The MLAAUTOSENSE bit reset indicates the software issues the request sense operation when a check condition status (4'02') is returned. The I/Os are then logged as two separate I/Os. The retried original I/O result consists of only its status byte (4'02'), and the actual retry bit MLAACTUALRETRY (indicated in word 6 field [47: 1] of the I/O record area) is set to TRUE. The request sense I/O result contains its status byte (4'00') followed by the request sense data for as many as 255 bytes, and its actual retry bit is set to FALSE.

Note *If the original status is not check condition (that is, not 4'02'), then a request sense operation is not performed, either by software or hardware, and the result contains only the original status. If the request rense I/O does not complete successfully (that is, the request sense status is not 4'00'), the data does not follow the request sense status byte.*

The following ALGOL example demonstrates how to access the retry commands and results stored in the command and result area. In this example, LOGBUF is the array into which the log record has been read, RETRYNUM is an integer variable, and P is a pointer to a print buffer.

```
DEFINE     % PARTIAL LAYOUT OF IO ERROR LOG RECORDS

  MLAGENINFODESCV         = 5 #,    % GENERAL AREA
    MLAGENINFOAREALOC (LA) = LA [MLAGENINFODESCV].[15:16] #,
    MLARETRYENTRYSIZE (LA) = (LA [MLAGENINFOAREALOC (LA) +2]
                              .[44: 5]) #,
    MLANUMRETRIES (LA)     = (LA [MLAGENINFOAREALOC (LA) +2]
                              .[ 8: 9]) #,

  MLAIORECORDDESCV        = 7 #,   % IO RECORD AREA
    MLAIORECORDAREALOC (LA) = LA [MLAIORECORDDESCV].[15:16] #,
  MLACRAREADESCV          = 8 #,   % IO COMMAND & RESULT AREA
    MLACRAREALOC (LA)      = LA [MLACRAREADESCV].[15:16] #,

    MLACOMMANDLENGTH (LA,N)= (LA [MLAIORECORDAREALOC (LA)
                                 + MLARETRYENTRYSIZE(LA)*(N)+1]
                              .[31:16]) #,
    MLACOMMANDOFFSET (LA,N)= LA [MLAIORECORDAREALOC (LA)
                                + MLARETRYENTRYSIZE(LA)*(N)+1]
                              .[15:16] #,

    MLACOMMAND (LA,N)      = LA [MLACRAREALOC (LA)
                                 + MLACOMMANDOFFSET (LA,N)] #,
    MLARESULTLENGTH (LA,N) = (LA [MLAIORECORDAREALOC (LA)
                                 + MLARETRYENTRYSIZE (LA)*(N)+5]
                              .[31:16]) #,
    MLARESULTOFFSET (LA,N)   = LA [MLAIORECORDAREALOC (LA)
                                  + MLARETRYENTRYSIZE (LA) * (N)+5]
                               .[15:16] #,
    MLARESULT (LA, N)        = LA [MLACRAREALOC (LA)
                                  + MLARESULTOFFSET (LA,N)] #;
```

```
FOR RETRYNUM := Ø STEP 1 UNTIL MLANUMRETRIES (LOGBUF) - 1 DO
  BEGIN
    REPLACE P BY "RETRY NUMBER ", RETRYNUM FOR 2 DIGITS,
      ", COMMAND ",
         POINTER (MLACOMMAND (LOGBUF, RETRYNUM), 4)
           FOR 2 * MLACOMMANDLENGTH (LOGBUF, RETRYNUM)
             WITH HEXTOEBCDIC,
      ", RESULT ",
         POINTER (MLARESULT (LOGBUF, RETRYNUM), 4)
           FOR 2 * MLARESULTLENGTH (LOGBUF, RETRYNUM)
             WITH HEXTOEBCDIC;

     % % %  print output buffer and then clear it with blanks

   END;
```

Normal I/O Exceptions: Extended Status Area

The extended status area referred to by Word 9 contains the number of bytes of extended status information specified by the MLAXSTATUSAREASIZE field (field [23:12] of Word 2).

Native SCSI devices have no extended status data logged in this area. The request sense information is logged with each I/O as it is encountered.

Normal I/O Exceptions: File and Job Information Areas

The number of file information items is variable, specified by MLANUMFILEITEMS. There is only one item of job information. The first word of each item specifies the type and length of the item. The following table shows the format of each item:

Word	Description
0	Item descriptor [39:20] length of item (including this word) [19:20] TYPE of item 2 = Item is FILE TITLE information 3 = Item is JOB NAME information

If field [19:20] of Word 0 is 2 (FILE TITLE), then Words 1–end have the following format:

1	Area number of the file on pack or disk. This word is all ones if no valid row index is present or if the unit is not a pack or disk.
2–end	If the length of the FILE TITLE item is greater than two, the rest of the item is the file title in standard form.

If field [19:20] of Word 0 is 3 (JOB NAME), then Words 1–end have the following format:

1–end	The job name in standard form

Error IOCBs

Error IOCBs are completed by the IOP on EMS systems when the error detected by the IOP is not associated with any particular I/O request.

Error IOCBs are logged in formats similar to those shown in Table 12–34, "Major Type 2–Minor Type 21," except that certain words are not used. A brief overview of the information present in these words follows:

Word	Description
4	MLAOVERALLVERSIONV; overall log entry version, currently 0 (zero).
5	MLAGENINFODESCV; LINK to general information area
	[47:16] Version; currently 0 (zero)
	[31:16] Area size in words; 1 for Error IOCB information
	[15:16] Word index into area
6	MLASTATINFODESCV; statistical information is not present.
7	MLAIORECORDDESCV; I/O record area is not present.
8	MLACRAREADESCV; command and result information is not present.
9	MLAXSTATUSDESCV; LINK to error IOCB extended status area
	[47:16] Version; currently 0 (zero)
	[31:16] Area size in words
	[15:16] Word index into area
10	MLAFILEINFODESCV; file information is not present.
11	MLAJOBINFODESCV; job information is not present.
12–end	Variable-length information.

Following are the variable-length areas described by the LINKs in the error IOCB log entry.

Error IOCBs: General Information Area

The general information area described by the LINK at Word 5 has the following format:

Word	Description
0	Control Information
	[47: 8] MLALOGSTATUS
	4 = MLAERRORIOCB
	[39: 4] 0
	[35:12] 0
	[23: 8] MLAEINUMTAGWORDS; number of words of information in the DLP result area of the error IOCB.

continued

continued

Word	Description
	[15: 8] MLAIOPROTOCOL
	1 = MLI protocol
	[7: 8] MLASYSTEMTYPE
	1 = A 2
	2 = A 6
	5 = A 3
	6 = A 9
	9 = A 10
	10 = A 5
	16 = A 1
	19 = A 4
	20 = Micro A

Error IOCBs: Extended Status Area

This section contains the DLP result area described by the link at word 9 of the error IOCB. Each DLP result area word is stored as 52 bits of information starting at Word 0 of this area. Figure 12–2 shows the general format of this area.

Area Word Ø		Area Word 1	
DLP result Word Ø Tag (4 bits)	DLP result Word Ø (48 bits)	DLP result Word 1 Tag (4 bits)	DLP result Word 1 (48 bits)

Figure 12–2. DLP Result Format for Error IOCBs

Refer to individual system reference manuals for a more detailed description of the information returned in these words. Note that the whole error IOCB is not returned.

Unassociated I/O Exceptions

The intelligent peripheral interface (IPI) response packet might indicate a request to read the error log for some reason not associated with the command that was issued. Therefore, much of the information found in a normal I/O exception log entry might be 0 (zero) in an unassociated I/O log entry.

Unassociated errors are logged in formats similar to those shown in Table 12–34, "Major Type 2–Minor Type 21," except that certain words are not used. A brief overview of the information present in these words follows:

Word	Description
4	MLAOVERALLVERSIONV; overall log entry version, currently 0 (zero).

continued

continued

Word	Description
5	MLAGENINFODESCV; LINK to general information area [47:16] Version; currently 0 (zero) [31:16] Area size in words [15:16] Word index to the area
6	MLASTATINFODESCV; statistical information is not present.
7	MLAIORECORDDESCV; LINK to I/O record area [47:16] Version; currently 0 (zero) [31:16] Area size in words [15:16] Word index to the area
8	MLACRAREADESCV; command and result information is not present.
9	MLAXSTATUSDESCV; LINK to extended status area [47:16] Version; currently 0 (zero) [31:16] Area size in words [15:16] Word index to the area
10	MLAFILEINFODESCV; file information is not present.
11	MLAJOBINFODESCV; job information is not present.
12–end	Variable-length information.

The following text contains variable-length information as described previously.

Unassociated I/O Exceptions: General Information Area

The general information area described by the LINK at Word 5 has the following format:

Word	Description
0	Control information [47: 8] MLALOGSTATUS 2 = MLAXSTATUSONLY [39: 4] 0 [35:20] 0 [15: 8] MLAIOPROTOCOL 0 = Intelligent peripheral interface (IPI) [7: 8] MLASYSTEMTYPE

continued

continued

Word	Description
	17 = A17
1	Device-specific information
	[47:12] MLACTLTYPE; the control type
	4'58' = 9399-H control
	[35: 8] MLALUTYPE; logical unit type or 0
	[27:12] MLAUNITSUBTYPE; standard unit subtype or 0
	[15:16] MLALU; logical unit number or 0
2	I/O retry and extended status control information
	[47: 1] 0
	[46: 1] 0
	[45: 1] 0
	[44: 5] 0
	[39: 4] MLAXSTATUSCONTROL
	1 = MLAGOODXSTATUS; extended status (error log) information is present.
	2 = MLABADXSTATUS; an error occurred while attempting to obtain extended status (error log).
	[35:12] 0
	[23:12] MLAXSTATUSAREASIZE; number of bytes of extended status information.
	[11: 3] 0
	[8: 9] 0
3	MLAIOCW
	0
4	MLAEXTU; external unit number of the unit that incurred the exception or 0 (zero).
5	MLABASEADDR
	0
6	Stack numbers
	0
7	I/O Length and stack numbers
	0
8	MLASTIME; a 36-bit integer that indicates time of day in units of 2.4 microseconds.

continued

continued

Word	Description
9	MLAFULLRD
	0

Unassociated I/O Exceptions: I/O Record Area

The I/O record area described by the link at Word 7 has the following format. Only one entry is described.

Word	Description
0	MCLACW
	0
1	Command packet information
	0
2	MLAINITPATH
	0
3	MLACOMPPATH; completing path information
	[47:16] External device number of port
	[31:16] External device number of control
	[15:16] External device number of unit
4	MLALIORD
	0
5	Result packet information
	0
6	[47: 1] MLAACTUALRETRY =1
	[46: 3] 0
	[43:16] MLAEXCEPTIONID
	[27:11] 0
	[16:17] MLALRD = 0
7	MLAMISCINFO
	0

Unassociated I/O Exceptions: Extended Status Area

The extended status area described by the LINK at Word 9 has the following format:

Table 12–35. Major Type 2–Minor Type 22

Minor Type = 22 Unit I/O Counts (UNITCOUNTS)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Entry information [47: 1] Unit counts cleared 0 = Subtotal record; unit counts intact 1 = Totals record; unit counts initialized [46: 3] Reason for logging counts 0 = Hourly record 1 = Log being transferred 2 = Prior to a scheduled power off 3 = Prior to changing the MCP [43:16] Number of units with counter entries [27: 8] LINK to unit entry information
5	Time elapsed since halt/load or last counter initialization
The following text contains unit entries of five words each. The number of unit entries (Word 4) that are logged is variable, so this section is of variable length. These words contain values since the last halt/load or since the counter was initialized. The format of each entry is as follows:	
Entry Word 0	Unit ID and bytes transferred [47:12] Physical unit number [35: 6] Logical unit type [29:30] I/O transfer volume in units of 1000 bytes
Entry Word 1	External device number
Entry Word 2	I/O counts (word 1) [47:24] Read I/O operations [23:24] Write I/O operations
Entry Word 3	I/O counts (word 2) [47:24] Read I/O errors

continued

Table 12–35. Major Type 2–Minor Type 22 (cont.)

Minor Type = 22 Unit I/O Counts (UNITCOUNTS)	
Words	**Description**
	[23:24] Write I/O errors
Entry Word 4	I/O counts (word 3)
	[47:24] Read I/O errors successfully recovered by the controller
	[23:24] Write I/O errors successfully recovered by the controller

Table 12–36. Major Type 2–Minor Type 23

Minor Type = 23 Mainframe Hardware Report (MLMFHWREPORT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	MCP version
	[47:16] MARK_LEVEL
	[31:16] CYCLE
	[15:16] PATCH_NUMBER
5	Time mainframe event occurred
6	Mainframe configuration information
	[47: 8] System type
	21 = A 16
	[39: 8] (Field reserved for future use)
	[31: 8] CPM mask
	[23: 8] I/O processor mask
	[15: 8] Memory mask
	[7: 4] Distinguished IOU
	[3: 4] Distinguished TCU

continued

Table 12–36. Major Type 2–Minor Type 23 (cont.)

Minor Type = 23 Mainframe Hardware Report (MLMFHWREPORT)	
Words	**Description**
7	System serial number
8–9	(Words reserved for future use)
10	Report entry information
	[47:40] (Field reserved for future use)
	[7: 8] Type of entry
	1 = A 16 CPM interrupt
	2 = A 16 IOM report
	3 = A 16 report via console
11–end	Variable data, depending on the value reported in the type of entry field [7: 8] of Word 10.

Major Type = 3 String Entry (LOGMAJSTR)

Table 12–37. Major Type 3–Minor Types 1 through 9

Minor Type = 1 RSVP Message Entry (RSVP)
Minor Type = 2 Fatal Error Message Entry (FATAL)
Minor Type = 3 Nonfatal Error Message Entry (NONFATAL)
Minor Type = 4 System Message Entry (SYSTEM)
Minor Type = 5 Unit Message Entry (UNIT)
Minor Type = 6 Directory Message Entry (DIRECTORY)
Minor Type = 7 Display Message Entry (DISPLAY)
Minor Type = 8 Status Message Entry (STATUS)
Minor Type = 9 Unit RSVP Message Entry (UNITRSVP)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Number of bytes in string
5–end	Character string, in EBCDIC characters

Major Type = 4 MCS Entry (LOGMAJMCS)

Table 12–38. Major Type 4–Minor Type 1

Minor Type = 1 Log-On Entry (MCSLOGON)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Log-on priority information
5	Sign-on class 0 = Unswitched 1 = New log-on 2 = HELLO 3 = New charge code 4 = SPLIT (Session number change) 5 = New access code
6	LINK to station name
7	LINK to usercode
8	LINK to chargecode
9	Represents the time of day when last connected. If log-on is a new connect and the word is valid, bit 47 is on.
10	Originator information [47: 1] Privileged user (if result indicator = relevant) [46: 1] Security administrator (if result indicator = relevant) [23: 8] MCS number [15:16] LSN
11	LINK to access code
12	LINK to privilege information
13–end	Variable-length data pointed to by the LINKs in this entry.

Table 12–39. Major Type 4–Minor Type 2

Minor Type = 2 Log-off Entry (MCSLOGOFF)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Log-off priority information
5	Log-off class
6	LINK to station name
7	LINK to user code
8	LINK to chargecode
9	Not used
10	Originator information [23: 8] MCS number [15:16] LSN
11	Processor time
12	I/O time
13	Number of lines printed
14	Number of cards read
15	Number of cards punched
16–17	Not used
18	Termination condition 0–199 = Unknown termination condition 200 = Normal log off 201 = MCS forced to EOJ (QUITted) 202 = NSP died 203 = Disconnected 204 = New log-on sequence required (Log-on status changed by system operator) 205 = Station cleared by operator or program 206 = Station released to another MCS

continued

Table 12–39. Major Type 4–Minor Type 2 (cont.)

Minor Type = 2 Log-off Entry (MCSLOGOFF)	
Words	**Description**
	207 = Error abort
	208 = MCS restart
	209 = Change of charge code
	210 = Session number returned
	211 = Split Session
	212 = Log off by HELLO
	213 = New access code
	214 = Log off because a station was detached (for example, because of a ?CLOSE command or because the MCS terminated)
	215 = Reserved for expansion
	216 = Security cleared
19–22	Not used
23	Session elapsed time (since log-on, split, or hello)
24	LINK to access code
25–end	Variable-length data pointed to by the LINKs in this entry

Table 12–40. Major Type 4–Minor Type 3

Minor Type = 3 RJE Control Card Entry (MCSRJECC)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Length of control card image (in units of characters)
5–end	Control card image as a string of EBCDIC characters

Table 12–41. Major Type 4–Minor Type 4

Minor Type = 4 MCS Message Entry (MCSMESSAGE)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Length of message (in units of characters)
5–end	MCS message as a string of EBCDIC characters

The system logs a Major Type 4, Minor Type 5 (MCSACCRUAL) entry when it is necessary to record the resource usage of an MCS session before the session ends. For example, the system issues this log entry if the user changes the charge code during a session.

Table 12–42. Major Type 4–Minor Type 5

Minor Type = 5 Session Accrual Entry (MCSACCRUAL)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4–end	As described in Table 12–39, "Major Type 4–Minor Type 2," except that the termination condition word (Word 18) is not used in an MCSACCRUAL entry.

Table 12–43. Major Type 4–Minor Type 6

Minor Type = 6 MCS Security Violation Entry (MCSSECURITY)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Error information [34: 1] Line was disconnected [33: 1] Station was saved [32: 1] Station was cleared (and logged off if necessary) [31:16] Installation-determined error code [15:16] MCS error code 0 = Installation-detected error at log-on 1 = Installation-detected error at chargecode change 2 = Installation-detected error at accesscode change 3 = Invalid usercode/password at log-on 4 = Invalid chargecode at log-on 5 = Invalid accesscode/password at log-on 6 = Invalid station name at log-on (RJE only) 7 = Invalid chargecode while changing chargecodes 8 = Invalid accesscode while changing accesscodes 9 = Invalid old password while changing password of usercode 10 = Invalid old password while changing password of accesscode 11 = Attempted to access unauthorized COMS window 12 = Attempted to log on from an MCS other than COMS with a usercode for which COMSONLYLOGON is TRUE 13 = NODEFAULTUSE set for usercode 14 = Log on attempts exceeded, station saved 15 = Log on attempts exceeded, station saved and disconnected
5	Identification [39:16] Session number if already logged on

continued

Table 12–43. Major Type 4–Minor Type 6 (cont.)

Minor Type = 6 MCS Security Violation Entry (MCSSECURITY)	
Words	**Description**
	[23: 8] MCS number [15:16] LSN
6	LINK to station identifier
7	LINK to usercode
8	LINK to chargecode
9	LINK to accesscode
	The usercode, chargecode, or accesscode is supplied only if it has already been successfully validated. For example, if the usercode/password failed validation on a new log on, words 7, 8, and 9 would be all zeros. The old chargecode or accesscode is supplied in error codes 1, 2, 7, and 8. In the case of error code 6, which is detected only by RJE with the STATIONID option set, the station identifier is the default station for that line.
10	LINK to input string that failed validation
11–end	Variable-length data pointed to by the LINKs in this entry

Table 12–44. Major Type 4–Minor Type 7

Minor Type = 7 MCS Station Application (LOGMIN_STATAPPLV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Type 0 = CANDE log station
5	Event 0 = Attach station

continued

Table 12–44. Major Type 4–Minor Type 7 (cont.)

Minor Type = 7 MCS Station Application (LOGMIN_STATAPPLV)	
Words	**Description**
	1 = Change station application
	2 = Detach station
6	LINK to originating station name
7	LINK to destination station name
8	LINK to originating usercode
9	LINK to destination usercode
10	Elapsed time attached
11	Identification
	[39: 8] MCS number
	[31:16] Destination LSN
	[15:16] Origination LSN
12	Miscellaneous information; if Type = 0 (CANDE log station) then
	[13: 1] = LGLOGSTA
	[12: 1] = AUTOINFO
	[11: 1] = LGFAULT
	[10: 1] = LGSABORT
	[9: 1] = LGUNABLE
	[8: 1] = LGSPO
	[7: 1] = LGSECURE
	[6: 1] = LGEOT
	[5: 1] = LGBOT
	[4: 1] = LGERROR
	[3: 1] = LGCHARGE
	[2: 1] = LGOFF
	[1: 1] = LGON
	[0: 1] = LGATTACH

continued

Table 12–44. Major Type 4–Minor Type 7 (cont.)

Minor Type = 7 MCS Station Application (LOGMIN_STATAPPLV)	
Words	**Description**
13–end	Variable-length data pointed to by the LINKs in this entry

Table 12–45. Major Type 4–Minor Types 8 through 10

Minor Type = 8 Remote Window Open/Close (LOGMIN_REMWINDOWV) **Minor Type = 9 MCS Window Open/Close (LOGMIN_MCSWINDOWV)** **Minor Type = 10 Direct Window Open/Close (LOGMIN_DIRWINDOWV)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Type of action 0 = Open 1 = Close 2 = Close all windows
5	LINK to program name (remote window open only)
6	LINK to window name (including dialog number)
7	Close reason 200 = Normal 202 = Pseudo deallocate 203 = Disconnect 206 = Release 212 = Hello 213 = Timeout 217 = Security

continued

Table 12–45. Major Type 4–Minor Types 8 through 10 (cont.)

Minor Type = 8 Remote Window Open/Close (LOGMIN_REMWINDOWV) Minor Type = 9 MCS Window Open/Close (LOGMIN_MCSWINDOWV) Minor Type = 10 Direct Window Open/Close (LOGMIN_DIRWINDOWV)	
Words	**Description**
8–9	Not used
10	Origination information [23: 8] MCS number [15:16] LSN
11	Not used
12–end	Variable-length data pointed to by the LINKs in this entry

Table 12–46. Major Type 4–Minor Type 11

Minor Type = 11 MCS Command (LOGMIN_MCSCCMDV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Length of command, in EBCDIC characters
5–end	Command string, in EBCDIC characters

Major Type = 5 NSP Entry (LOGMAJNSP)

Table 12–47. Major Type 5–Minor Type 1

Minor Type = 1 NSP Initiate Entry (LOGNSPINIT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4–5	Not used
6	Relative NSP number in [22: 7] with bit 23 on
7–end	DC prefix name in display form

Table 12–48. Major Type 5–Minor Type 2

Minor Type = 2 MCS Initiate Entry (LOGMCSINIT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	MCS number
5–6	Not used
7–end	MCS name in standard form

Table 12–49. Major Type 5–Minor Type 4

Minor Type = 4 MCS Error Entry (LOGMCSMSG)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	MCS number
5	Not used
6	Word 0 of error result message (result class = 99) [23: 1] 1 is DLS, 0 is LSN [22:23] LSN [22: 7] Relative NSP number for DLS [15: 8] Line for DLS [7: 8] Station for DLS
7	Word 1 of the error result message [47: 8] RSLT byte index [39: 8] Line status [31: 8] Last flag [23:24] Error flag
8	Word 2 of the error result message Always 0 (zero)
9	Word 3 of the error result message
10	Word 4 of the error result message [23: 8] Original DCWRITE type [7: 8] Variant field
11	Error count

Table 12–50. Major Type 5–Minor Type 6

Minor Type = 6 UIO-DC Unsuccessful I/O (LOGUIOLINEFAULT)	
Words	**Description**
0–3	As described in Table 12-2, "First Four Log Entry Words"
4	MCS number
5	Not used
6	NSP number
7	LSP number
8	NSP words 0 through 2 of unsuccessful I/O result
	[47:16] Result count
	[31:16] Result number
	[15: 8] Result type
	[7: 8] Start of request number
9	NSP words 3 through 5 of unsuccessful I/O result
	[47:24] Request number
	[23:16] Line number
	[7: 8] Category
10	NSP words 6 through 8 of unsuccessful I/O result
	[47: 8] Abort reason
	[39: 8] Request I/O
	[31:16] Command count
	[15:16] Data count
11	NSP words 9 through 11 of unsuccessful I/O result
	[47:16] Transfer count
	[31: 8] LEM port number
	[23: 8] Unit select
	[15: 8] Data transfer
	[7: 8] DLP number

continued

Table 12–50. Major Type 5–Minor Type 6 (cont.)

Minor Type = 6 UIO-DC Unsuccessful I/O (LOGUIOLINEFAULT)	
Words	**Description**
12	NSP words 12 through 14 of unsuccessful I/O result [47: 8] Selective clear [39: 8] Address parity [31: 8] Cleared DLP [23: 8] DLP status [15: 8] Exception [7: 8] Hung DLP
13	NSP words 15 through 17 of unsuccessful I/O result [47: 8] MLI longitudinal parity error [39: 8] MLI timeout [31: 8] MLI vertical parity error [23: 8] Nonpresent DLP [15: 8] Port busy [7: 8] Unexpected DLP status
14–end	Command and result [47:16] Number of bytes in the command. This is followed by the command for the specified length, a 16-bit quantity giving the length of the result, and the result for the specified length.

Table 12–51. Major Type 5–Minor Type 7

Minor Type = 7 UIO-DC Abnormal Termination (LOGUIOPROCFAULT)	
Words	**Description**
0–3	As described in Table 12-2, "First Four Log Entry Words"
4	MCS number
5	Not used
6	NSP number
7	LSP number
8	Adapter number
9	NSP words 0 through 2 of abnormal termination result
	[47:16] Result count
	[31:16] Result number
	[15: 8] Result type
	[7: 8] Start of request number
10	NSP words 3 through 5 of abnormal termination result
	[47:24] End of request number
	[23: 8] Process history
	[15: 8] Abort reason
	[7: 8] Start of process index
11	NSP words 6 through 8 of abnormal termination result
	[47: 8] End-of-process index
	[39:16] PC segment
	[23:16] PC offset
	[7: 8] Process type
12	NSP word 9 of abnormal termination result
	[47:16] Line number / station number. If the process type is Controller or LSP process, this field contains the line number. If the process type is Editor, this field contains the station number.

Major Type = 6 Miscellaneous Entry (LOGMAJMISC)

Table 12–52. Major Type 6–Minor Type 1

Minor Type = 1 Halt/Load Entry (LOGHL)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	MCP and system identification [47:12] System serial number [35:12] MCP mark number [23:12] MCP level number [11:12] MCP patch number
5	LINK to MCP name
6–8	Halt/load cause as a string of EBCDIC characters
9	System identification [47:16] System serial number [31:24] Not used [7: 8] Operating system type 0 = MCP 1 = MCP 2 = MCP/AS
10–end	Variable-length data pointed to by the LINK in this entry

Table 12–53. Major Type 6–Minor Type 3

Minor Type = 3 SETSTATUS Call Entry (LOGSETSTAT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Information regarding parameters to SETSTATUS call [47: 8] TYPE, the first parameter [39: 8] SUBTYPE, the second parameter [31: 8] V, the third parameter [15:16] Number of words to follow in the entry
5 for n words	Contents of the array parameter to SETSTATUS where n is the number of words specified in field [15:16] of Word 4.
If the length of the entire log record is greater than $n + 5$ words, the remaining words have the following meaning:	
$n + 5$	Identity information code [47:06] Length of fixed part of entry in words [41:10] Entire length of identity information
$n + 6$	Miscellaneous identity information [07:08] Source of SETSTATUS command 1 = From ODT 2 = From remote ODT 3 = From BNA connected host 4 = From DCKEYIN 5 = From MCS
$n + 7$	LINK to SOURCENAME task attribute value
$n + 8$	LINK to USERCODE task attribute value. This word is used only if the source of the SETSTATUS command was a BNA connected host (that is, field [07:08] of Word $n + 6$ stores a 3).
$n + 9$	The V (third parameter) value of SETSTATUS

Note that the field [31: 8] of Word 4 differs from the third parameter value (V) in the following cases:

Case	Value in Word 4 [31: 8]
AD or DD	V & (V.[10: 1]) [1: 1] & (V.[39: 1]) [2: 1]
PP or XP	IF V = –1 THEN 6 ELSE V
SW–	IF V = –1 THEN 6 ELSE V

Note: *The SETSTATUS call entry records a call to SETSTATUS, which is the MCP procedure called to carry out operator input commands. Thus, these records show most operator input that changes the status of the system, such as time and date changes, tapes purged, and tasks discontinued.*

Table 12–54. Major Type 6–Minor Type 4

Minor Type = 4 Security Violation Entry (LOGSECURITY)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to usercode of violator
5	Violation code as follows:
	1 = Attempted to open a PRIVATE file
	2 = Attempted to open INPUT on a write-only file
	3 = Attempted to open OUTPUT on a read-only file
	4 = Attempted to open I/O on a read or write only file
	5 = Attempted to rename someone else's file
	6 = Attempted to create a permanent file not under your usercode
	7 = Attempted to execute a PRIVATE code file
	8 = Use of a usercode when SYSTEM/USERDATAFILE is not defined
	9 = Invalid usercode
	10 = Invalid password
	11 = Attempted to remove someone else's file
	12 = Invalid use of file name *USERCODE
	13 = Invalid use of security file attributes

continued

Table 12–54. Major Type 6–Minor Type 4 (cont.)

Minor Type = 4 Security Violation Entry (LOGSECURITY)	
Words	**Description**
	14 = Attempted to COPY a PRIVATE file
	15 = Unauthorized user attempted to modify SYSTEM/USERDATAFILE
	16 = Not a viable usercode (no SYSTEM node in SYSTEM/USERDATAFILE entry)
	17 = Suspended usercode
	18 = Old password required but not supplied (changing passwords)
	19 = Old password incorrect (changing passwords)
	20 = Invalid chargecode
	21 = Invalid USERCODE/password
	22 = Attempted to copy a GUARDED file
	23 = Attempted to copy a CONTROLLED file
	24 = Attempted to execute a GUARDED file
	25 = Attempted to execute a CONTROLLED file
	26 = Attempted to execute a nonexecutable file
	27 = Invalid USERCODE when initiating a task
	28 = Invalid ACCESSCODE when initiating a task
	29 = Invalid task-to-task attribute: USERCODE
	30 = Invalid task-to-task attribute: ACCESSCODE
	31 = Invalid task attribute: no USERCODE
	32 = Invalid task attribute: USERCODE
	33 = Invalid task attribute: BDNAME
	34 = Invalid task attribute: ACCESSCODE
	35 = Invalid task attribute: JOBSUMMARYTITLE
	36 = Unauthorized user attempted to call SETSTATUS
	37 = Unauthorized user attempted to call GETSTATUS
	38 = Unauthorized user attempted to call DCKEYIN
	39 = Unauthorized user attempted to call ATTACHSPOQ

continued

Table 12–54. Major Type 6–Minor Type 4 (cont.)

Minor Type = 4 Security Violation Entry (LOGSECURITY)	
Words	**Description**
	40 = Unauthorized user attempted to execute WFL VOLUME statement
	41 = Attempted to execute MU(PU) system command
	42 = USERCODE no longer valid for CONTROLCARD
	43 = USERCODE no longer valid for JOBRESTART
	44 = USERCODE no longer valid for TASKRESTART
	45 = Unauthorized call to PrintS
	46 = Invalid USERCODE/PRINTCHARGE
	47 = Invalid USERCODE for PrintS Transform
	48 = Attempted to rename a GUARDED file
	49 = Attempted to rename a CONTROLLED file
	50 = Unauthorized user attempted to execute security critical function
	51 = Unauthorized user attempted to execute *??SECAD-* system primitive command
	52 = Attempted to execute a restricted file
	53 = Attempted to execute file on restricted family
	54 = Attempted to copy a restricted file
	55 = Attempted to copy file from restricted family
	56 = Attempted to CM to a restricted file
	57 = Attempted to CM to file on a restricted family
	58 = File security prevented row exchange
	59 = Expired password
	60 = Expired USERCODE
	61 = Attempted to set a file attribute without write access
	62 = Attempted to create a restricted file
	63 = Userdata function not allowed on password generating system
	64 = Makeusercode request denied
	65 = Library linkage class security violation

continued

Table 12–54. Major Type 6–Minor Type 4 (cont.)

Minor Type = 4 Security Violation Entry (LOGSECURITY)	
Words	**Description**
	66 = Cannot access transform library by title
	67 = Minimum password lifespan enforced, password change failed
	68 = Password recently used, password change failed
	69 = Program dump file copied
	70 = Attempt to copy a program dump file to a restricted destination
	71 = Attempt to open a keysfile
	72 = Attempt to open a checkpoint file
	73 = Attempt to open a program dump file
	74 = Attempt to use a catalog command on a file
	75 = Attempt to use ARCHIVE PURGE command on a file
	76 = Attempt to update the archive record for a file
	77 = Attempt to read the archive record for a file
	78 = Usercode invalid because not in correct date and time range
	79 = Attempt to copy or archive files from another user's directory
	80 = Family owner mismatch or security error on tape open
6	LINK to name (can be file name, password and so on, depending upon violation involved)
7	LINK to access code
8	Violation origin information
	[47: 1] Task originated from a WFL queue
	[46:10] Queue from which task originated
	[36:15] Unused field
	[21: 6] MCS from which task originated
	[15: 1] Task originated from data comm
	[14:15] Unused

continued

Table 12–54. Major Type 6–Minor Type 4 (cont.)

Minor Type = 4 Security Violation Entry (LOGSECURITY)	
Words	**Description**
9	If task originated from data comm, LSN of originating terminal; otherwise, unit number of originating peripheral
10–end	Variable-length data pointed to by the LINKs in this entry

Table 12–55. Major Type 6–Minor Type 5

Minor Type = 5 Deimplementation Warning Entry (LOGDEIMPWARNING)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Deimplementation warning information
	[7: 8] Warning number
5	LINK to external name of task
6	LINK to usercode
7	LINK to compiler file name
8	Task source and destination information
	[45: 6] Destination MCS
	[39: 1] Destination is remote
	[38:15] Destination UNIT
	[21: 6] Originating MCS
	[15: 1] Origin is REMOTE
	[14:15] Originating UNIT
	If bit [15: 1] is set, this field contains a logical station number (LSN). If bit [15: 1] is reset, this field contains the logical unit number.

continued

Table 12–55. Major Type 6–Minor Type 5 (cont.)

Minor Type = 5 Deimplementation Warning Entry (LOGDEIMPWARNING)	
Words	**Description**
9	Compiler information [31: 8] Language type [23: 8] Compiler mark number and level number [9:10] Compiler cycle number
10	LINK to value of itinerary chain
11–end	Variable-length data pointed to by the LINKs in this entry

Table 12–56. Major Type 6–Minor Type 6

Minor Type = 6 Log Power Off (LOGPOWEROFF)	
Words	**Description**
0–3	As described in Table 12–2, “First Four Log Entry Words”
4	Power-off indicator [47:16] System serial number [7: 8] Type of power-off 1 = Scheduled power-off 2 = Unscheduled due to thermal overload 3 = Thermal overload warning 4 = Unscheduled power-off (request for immediate power-off) 5 = Canceled power-off 6 = Actual power-off 7 = Blower failure

Table 12–57. Major Type 6–Minor Type 7

Minor Type = 7 Controller Command (LOGMIN_CONTRCMDV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Length of command, in EBCDIC characters
5 for n characters	Command string in EBCDIC characters, where n is the number of characters specified in field [15:16] of Word 4.
If the length of the entire log record extends beyond that needed to to accommodate the command string, the remaining words have the following meaning, where m is the index of the first such word:	
m	Identity information code [47:06] Length of fixed part of entry in words [41:10] Entire length of identity information
$m + 1$	Miscellaneous identity information [07:08] Source of SETSTATUS command 1 = From ODT 2 = From remote ODT 3 = From BNA connected Host 4 = From DCKEYIN 5 = From MCS
$m + 2$	LINK to SOURCENAME task attribute value
$m + 3$	LINK to USERCODE task attribute value. This word is used only if the source of the SETSTATUS command was a BNA connected host (that is, field [07:08] of Word $n + 6$ stores a 3).

This entry describes the modification or deletion of a print request as the result of an operator command.

Table 12–58. Major Type 6–Minor Type 8

Minor Type = 8 Print Subsystem Command (LOGPSCOMMAND)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Command information [47: 8] Type of PS command 1 = ADDFILES 2 = ASSOCIATE 3 = BNA 4 = CONFIGURE 5 = DEFAULT 6 = DELETE 7 = DEVICES 8 = FORCE 9 = GROUP 10 = MODIFY 11 = NOTOK 12 = OK 13 = REQUEUE 14 = SERVERS 15 = SKIP 16 = STOP [31: 4] Result information [27:28] Request number, if applicable
5	LINK to usercode of print request originator
6	LINK to station name of command originator
7	LINK to usercode of command originator
8	LINK to command text
9	Session number of command originator

continued

Table 12–58. Major Type 6–Minor Type 8 (cont.)

Minor Type = 8 Print Subsystem Command (LOGPSCOMMAND)	
Words	**Description**
10–end	Variable-length data pointed to by the LINKs in this entry

Table 12–59. Major Type 6–Minor Type 9

Minor Type = 9 USERDATA Change (LOGMIN_CHANGEUSERV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Action [12: 1] Action was for remote user [11: 4] USERDATA operation (Modify/create/delete/see entry = 7) 5 = Modify 6 = Create 7 = Delete [7: 4] USERDATA Subfunction [3: 4] USERDATA Function
5	LINK to information (specific for USERDATA function) (Password change = 6) Usercode (Modify/create/delete/see entry = 7) (Create entry using model = 8) Doings parameter to USERDATAREBUILD (Change accesscode/password = 12)

continued

Table 12–59. Major Type 6–Minor Type 9 (cont.)

Minor Type = 9 USERDATA Change (LOGMIN_CHANGEUSERV)	
Words	**Description**
	Usercode/accesscode
6–end	Variable-length data pointed to by the LINK in this entry

Note that the system creates a backup copy of the USERDATAFILE whenever an entry of Major Type 6, Minor Type 11 (New USERDATA Install) is logged. Refer to "Requesting Copies of Configuration Files" earlier in this section.

Table 12–60. Major Type 6–Minor Types 10 and 11

Minor Type = 10 New USERDATA Install & Copy (LOGMIN_NEWUSERCOPYV) **Minor Type = 11 New USERDATA Install (LOGMIN_NEWUSERV)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to installed file title
5	LINK to copied file title
6	Action 1 = USERDATA create 2 = USERDATA recall 3 = USERDATA copy new (a copy is not made for this case)
7–end	Variable-length data pointed to by the LINKs in this entry

Table 12–61. Major Type 6–Minor Type 12

Minor Type = 12 Primitive Command (LOGMIN_PRIMCMDV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Length of command, in EBCDIC characters
5–end	Command string, in EBCDIC characters

Table 12–62. Major Type 6–Minor Type 13

Minor Type = 13 Idle System Times (LOGMIN_TIMERV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"

Table 12–63. Major Type 6–Minor Type 14

Minor Type = 14 Halt/Load Reason (LOGMIN_HL2V)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	MCP and system identification [47:12] System serial number [35:12] MCP mark number [23:12] MCP level number [11:12] MCP patch number

continued

Table 12–63. Major Type 6–Minor Type 14 (cont.)

Minor Type = 14 Halt/Load Reason (LOGMIN_HL2V)	
Words	**Description**
5	LINK to MCP name
6–8	Halt/load cause as a string of EBCDIC characters
9	System identification [47:16] System serial number [31:24] Not used [7: 8] Operating system type 0 = MCP 1 = MCP 2 = MCP/AS
10–end	Variable-length data pointed to by the LINK in this entry

Table 12–64. Major Type 6–Minor Type 15

Minor Type = 15 DRC error logging (LOGMIN_DRCV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Error code 1 = Invalid FAMILYNAME in USERDATA FAMILYLIST entry 2 = USERDATA record overflow 3 = USERDATA error
5	LINK to usercode

continued

Table 12–64. Major Type 6–Minor Type 15 (cont.)

Minor Type = 15 DRC error logging (LOGMIN_DRCV)	
Words	**Description**
6	LINK to additional information. Error codes 1 and 2 are USERDATA FAMILYLIST entries; error code 3 is a USERDATA result.
7	Variable-length data pointed to by the LINKs in this entry.

Table 12–65. Major Type 6–Minor Type 16

Minor Type = 16 Security Relevant Warning (LOGMIN_SECWARNV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Warning code
	1 = Support Library Replaced
5	LINK to additional information. For warning code 1 it is a link to the FUNCTIONNAME library attribute value.
6	Variable-length data pointed to by the LINKs in this entry.

Major Type = 7 Installation Entry (LOGMAJINST)

Reserved for installation use.

Major Type = 10 Date/Time Reset Entry (LOGMAJDATETIMERESET)

The date/time reset log entry is a single, 30-word record that is always the first record in the log file. It is used to record the location of the first 26 date/time changes that occur while this log file is active. Only date/time changes that reset the date or time to a value earlier than the current date or time are recorded. This entry is updated each time such a change is made.

Table 12–66. Major Type 10–All Minor Types

Minor Type = 1 Log created by A 2 **Minor Type = 2 Log created by A 6** **Minor Type = 5 Log created by A 3** **Minor Type = 6 Log created by A 9** **Minor Type = 7 Log created by A 15** **Minor Type = 9 Log created by A 10** **Minor Type = 10 Log created by A 5** **Minor Type = 15 Log created by A 12** **Minor Type = 16 Log created by A 1** **Minor Type = 17 Log created by A 17** **Minor Type = 19 Log created by A 4** **Minor Type = 20 Log created by Micro A** **Minor Type = 21 Log created by A 16**	
Words	**Description**
0	Record Group Description [47: 8] = 1 [39: 8] = 1 [31:16] = Serial number of system that created log [15:16] = Serial number of SUMLOG
1	Julian date in binary when this entry was first written
2	Time of day when this entry was first written
3	Log entry type code [47: 6] = 0 [41:10] = Number of date and time resets recorded [31:16] = 10 (Major Type) [15:16] = Type of system that created the log: 1 = A 2 2 = A 6

continued

Table 12–66. Major Type 10–All Minor Types (cont.)

Minor Type = 1 Log created by A 2
Minor Type = 2 Log created by A 6
Minor Type = 5 Log created by A 3
Minor Type = 6 Log created by A 9
Minor Type = 7 Log created by A 15
Minor Type = 9 Log created by A 10
Minor Type = 10 Log created by A 5
Minor Type = 15 Log created by A 12
Minor Type = 16 Log created by A 1
Minor Type = 17 Log created by A 17
Minor Type = 19 Log created by A 4
Minor Type = 20 Log created by Micro A
Minor Type = 21 Log created by A 16

Words	Description
	5 = A 3 6 = A 9 7 = A 15 9 = A 10 10 = A 5 15 = A 12 16 = A 1 17 = A 17 19 = A 4 20 = Micro A 21 = A 16
4–29	Contain pointers (relative record numbers) to the last entry with the old time (this entry's time and date will be greater than the next entry's time and date). If bit 47 is set in a word, there are additional time breaks in the log that have not been recorded in this record. This word is the last pointer provided. The remaining words in the entry do not contain pointers.

Major Type = 11 BNA Version 1 Entry (LOGMAJBNA)

Log entries of Major Type 11 record system actions that are related to BNA Version 1.

Tables 12–67 through 12–72 are referred to in many of the Major Type 11 log entry descriptions.

Table 12–67. NS Error Codes

Number	Description
1	ILLEGAL NUMBER
2	BNA VERSION 1 COMMAND EXPECTED
3	THIS FEATURE IS NOT YET IMPLEMENTED
4	NUMBER EXPECTED
5	BOOLEAN VALUE EXPECTED
6	NUMBER IS TOO LARGE
7	RESERVED WORD EXPECTED
8	INVALID LINK RESISTANCE FACTOR
9	NUMBER IS TOO SMALL
10	THERE IS NO NEXT SCREEN
11	PRINT OPTION EXPECTED
12	NO SUSPENDED FILE
13	FILE ALREADY SUSPENDED
14	MORE INPUT EXPECTED
15	EQUAL SIGN (=) EXPECTED
16	MODULUS ATTRIBUTE MUST BE 8 OR 128
17	END OF COMMAND EXPECTED
18	ILLEGAL OUTPUT DESIGNATION
19	RIGHT PARENTHESIS EXPECTED
20	COMMAND NOT ALLOWED FROM INPUT SOURCE
21	ILLEGAL TITLE
22	I/O ERROR READING INPUT FILE
23	INPUT FILE IS SUSPENDED - CONTINUE OR DISCARD
24	NO FILE BEING LOADED
25	FILE LOAD IN PROCESS
26	DEBUG OPTION EXPECTED
27	PLUS OR MINUS EXPECTED

continued

Table 12–67. NS Error Codes (cont.)

Number	Description
28	POUND SIGN (#) EXPECTED
29	COLON (:) EXPECTED
30	ALL EXPECTED
31	AGENT NAME EXPECTED
32	YOUR-NAME OF SUBPORT EXPECTED
33	STRING EXPECTED
34	INVALID NEIGHBOR RESTART TIMEOUT VALUE
35	COMMA (,) EXPECTED
36	COMMAND IS NOT VALID DURING THE CURRENT PHASE OF THE NETWORK
37	DESTINATION HOST IS UNREACHABLE
38	SOURCE HOST IS UNREACHABLE
39	TO EXPECTED
40	INVALID MAX SEGMENT SIZE
41	INVALID NETWORK VERSION
42	INVALID NEIGHBOR GREETING TIMEOUT VALUE
43	STRING TOO LONG
44	INVALID NEIGHBOR BUSY TIMEOUT VALUE
45	SOURCE AND DESTINATION ARE SAME NODE
46	USERCODE EXPECTED
47	AGENT EXCEEDS MAX PROGRAM AGENTS
48	THIS ATTRIBUTE CANNOT BE SET WHEN PROGRAM AGENTS ARE CONNECTED
49	FAILURE DUE TO SOFTWARE ERROR
50	USERCODE IS ALLOWED ONLY WHEN PROGRAM AGENT SECURITY IS PRIVATE
51	HOST ALREADY ADDED
52	HOST NOT DELETED, HOST IS COMMUNICATING
53	HOST VALIDATE IS FALSE, COMMAND HAS NO MEANING
54	ROUTER VALIDATE IS FALSE, COMMAND HAS NO MEANING
55	HOST NAME AND/OR NODE ADDRESS REQUIRED
56	HOST NAME AND/OR NODE ADDRESS MUST FOLLOW THE WORD WITH

continued

Table 12–67. NS Error Codes (cont.)

Number	Description
57	HOST NAME REQUIRED
58	MISSING NODE ADDRESS
59	HOST NAME EXPECTED
60	NODE ADDRESS EXPECTED
61	LOCAL NODE ADDRESS IS ALREADY SET
62	INVALID HOST NAME
63	INVALID NODE ADDRESS
64	HOST NAME IS NOT DEFINED
65	NODE ADDRESS IS NOT DEFINED
66	HOST NAME - NODE ADDRESS DO NOT MATCH
67	NEIGHBOR NOT PERMITTED HERE
68	NEIGHBOR MUST BE SPECIFIED
69	LOCAL NODE NOT ALLOWED IN COMMAND
70	STATION OR ENSEMBLE NAME REQUIRED
71	STATION OR ENSEMBLE NAME MUST FOLLOW THE WORD BY
72	NO SUCH STATION
73	STATION IS INCOMING-ONLY TYPE
74	STATION NOT ABLE TO RECEIVE INCOMING CALLS
75	STATION NOT ABLE TO PLACE OUTGOING CALLS
76	STATION IS PERMANENT TYPE
77	STATION IS GLOBAL MEMORY TYPE
78	STATION IS BDLC TYPE
79	STATION IS X.25 TYPE
80	STATION NAME REQUIRED
81	STATION NAME MUST FOLLOW 'BY STATION'
82	STATION NAME IS INVALID
83	STATION IS NOT PERMANENT TYPE
84	LCPACKETSIZE ATTRIBUTE MUST BE MODULO 16
85	ENSEMBLE NAME IS NOT DEFINED
86	ENSEMBLE IS INCOMING-ONLY TYPE
87	ENSEMBLE NOT ABLE TO RECEIVE INCOMING CALLS

continued

Table 12–67. NS Error Codes (cont.)

Number	Description
88	ENSEMBLE NOT ABLE TO PLACE OUTGOING CALLS
89	ENSEMBLE IS PERMANENT TYPE
90	ENSEMBLE IS GLOBAL MEMORY TYPE
91	ENSEMBLE IS BDLC TYPE
92	ENSEMBLE IS X.25 TYPE
93	ENSEMBLE NAME REQUIRED
94	ENSEMBLE NAME MUST FOLLOW 'BY ENSEMBLE'
95	ENSEMBLE NAME IS INVALID
96	ENSEMBLE IS NOT PERMANENT TYPE
97	STATION NOT BDLC SWITCHED STATION
98	CONNECTION TO THIS NEIGHBOR HAS ALREADY BEEN ADDED
99	NO CONNECTION TO THIS NEIGHBOR HAS BEEN DEFINED
100	CONNECTION ATTRIBUTES REQUIRED
101	CALL-DATA REQUIRED
102	CALL-DATA EXPECTED
103	INIT-QUANTITY REQUIRED
104	INIT-QUANTITY EXPECTED
105	CALL-DATA NOT PERMITTED HERE
106	NEIGHBOR PHRASE REQUIRED FOR OUTGOING CALLS
107	CALL-DATA MUST BE SUPPLIED
108	CALL-DATA ALREADY SUPPLIED IN CONNECTION - NOT PERMITTED HERE
109	INIT-QUANTITY NOT PERMITTED HERE
110	STATION BELONGS TO ENSEMBLE: CONNECTION MUST SPECIFY THAT ENSEMBLE
111	CALL-DATA TOO LONG
112	INIT-QUANTITY TOO LARGE
113	STATION OR ENSEMBLE HAS ACTIVE CONNECTION TO NEIGHBOR
114	STATION HAS NON-ZERO INIT-QUANTITY IN OTHER OUT CONNECTION
115	NO SUCH CONNECTION
116	STATION OR ENSEMBLE HAS ACTIVE CONNECTION

continued

Table 12–67. NS Error Codes (cont.)

Number	Description
117	CONNECTION HAS NO ATTRIBUTES TO MODIFY
118	INVALID ADD OPTION
119	INVALID DELETE OPTION
120	INVALID MODIFY OPTION
121	INVALID SAVE OPTION
122	INVALID READY OPTION
123	INVALID SHOW OPTION
124	INVALID NODE UP TIMEOUT VALUE
125	LEFT PARENTHESIS EXPECTED
126	STATION ATTRIBUTE EXPECTED
127	LOCAL IDENTITY HAS NOT BEEN SET
128	NODE NOT DELETED, NODE IS CONNECTED AS NEIGHBOR
129	NODE IS NOT A NEIGHBOR
130	NODE ALREADY ADDED
131	PORT NAME EXPECTED
132	ILLEGAL PASSWORD
133	INVALID CALL DATA
134	MESSAGE TEXT TOO LONG
135	DIRECTION (IN, OUT, PERM) REQUIRED
136	CALLS NOT ALLOWED - NETWORK IS INITIALIZING
137	CALLS NOT ALLOWED - NETWORK IS SHUTTING DOWN
138	PASSWORD TOO LONG
139	DUPLICATE PARAMETER
140	HOST NAME DOES NOT MATCH
141	PROGRAM AGENT WAS PREVIOUSLY SPECIFIED
142	STATION NOT AVAILABLE
143	STATION IS SAVED
144	STATION IS IN MANUAL MODE
145	STATION ALREADY CLOSED
146	STATION REQUIRES CALL-DATA FOR OUTGOING CALL
147	CALL-DATA NOT ALLOWED FOR STATION

continued

Table 12–67. NS Error Codes (cont.)

Number	Description
148	STATION NOT IN CORRECT STATE
149	STATION IS NOT ATTACHED
150	ISC STATION GROUP ATTRIBUTE EXPECTED
151	STATION-TYPE ATTRIBUTE NOT MODIFIABLE
152	ISC STATION GROUP NOT ALLOWED
153	ISC STATION NOT ALLOWED
154	HUB NUMBER ALREADY IN USE BY ANOTHER ISC GROUP
155	STATION NAME APPEARS IN LIST TWICE
156	HUB NUMBER REQUIRED
157	INVALID STATION TYPE
158	CONNECTION WAS NOT SPECIFIED
159	CALL CANNOT BE MADE ON THIS STATION
160	ENSEMBLE HAS AN ACTIVE STATION
161	ISC GROUP SHUTDOWN IN PROGRESS
162	NO ACTIVE STATIONS FOR GROUP
163	ACTIVE STATIONS CURRENTLY IN GROUP
164	STATION HAS CONNECTION DEFINED
165	ENSEMBLE HAS CONNECTION DEFINED
166	STATION IN ENSEMBLE NOT OF PROPER TYPE
167	X.25 ENSEMBLES VC CANNOT BE MODIFIED
168	STATION NOT IN THAT ENSEMBLE
169	STATION ALREADY IN AN ENSEMBLE
170	CAN'T DELETE STATION FROM ENSEMBLE
171	STATIONS IN OUT ENS NEED SAME SPEED
172	NO STATIONS AVAILABLE
173	ALL STATIONS IN ENSEMBLE ARE CLOSED
174	CAN'T DELETE LAST STATION IN ENSEMBLE
175	USERCODE IS REQUIRED WHEN PROGRAM AGENT SECURITY IS PRIVATE
176	STATION ALREADY ADDED
177	ENSEMBLE ALREADY ADDED

continued

Table 12–67. NS Error Codes (cont.)

Number	Description
178	INVALID STATION ATTRIBUTE
179	INVALID ISC GROUP ATTRIBUTE
180	CAN'T ADD STATION TO ENSEMBLE WHILE STATION IS ATTACHED
181	CAN'T DELETE STATION FROM ENSEMBLE WHILE STATION IS ATTACHED
182	TIME-INTERVAL TOO SMALL
183	TIME-INTERVAL TOO LARGE
184	X.25 STATION NOT ALLOWED
185	STATION IS NOT X.25 GROUP STATION
186	STATION IS NOT AN X.25 STATION
187	STATION GROUP IS NOT OPEN
188	LCN RANGE IS GREATER THAN NUMBER OF STATION NAMES DECLARED
189	LCN RANGE IS LESS THAN NUMBER OF STATION NAMES DECLARED
190	X.25 PDN NAME EXPECTED
191	MODIFIED ATTRIBUTE ALLOWED ONLY WHEN PDN IS NULL
192	X.25 LCN ALREADY ADDED
193	STATION-TYPE ATTRIBUTE REQUIRED
194	STATION-TYPE MUST BE THE FIRST ATTRIBUTE SPECIFIED
195	STATION STATUS MUST BE CLOSED
196	THIS COMMAND IS ALLOWED ONLY WHEN BNA VERSION 1 IS COMPILED WITH DIAGNOSTICS SET
197	ALLC-REFERENCE REQUIRED
198	INVALID SECURITYTYPE
199	INVALID FILENAME
200	PORT ATTRIBUTE EXPECTED
201	INVALID YOURNAME
202	SECURITYGUARD IS USED ONLY IF SECURITYTYPE IS GUARDED
203	YOURNAME REQUIRED
204	STATION CURRENTLY UNASSIGNED
205	NO TARGET ENTRIES IN DIRECTORY
206	NO SUCH TARGET

continued

Table 12–67. NS Error Codes (cont.)

Number	Description
207	NO SUCH ALLC
208	NO ALLC ENTRIES IN DIRECTORY
209	TARGET ALREADY ADDED
210	MISSING ALLC ENTRY
211	TARGET NOT DELETED, CURRENTLY IN USE FOR ALLC ACTIVATION
212	TARGET NOT REPLACED, CURRENTLY IN USE FOR ALLC ACTIVATION
213	ALLC ALREADY ADDED
214	ALLC NOT DELETED, IS REFERENCED BY TARGET
215	YOURNAME ATTRIBUTE NOT SET
216	SECURITYGUARD MUST BE SPECIFIED, IF SECURITYTYPE IS GUARDED
217	COMMAND IS INVALID, STATION IS ASSIGNED
218	INVALID TARGET-ID
219	TARGET-ID EXPECTED
220	ACTIVATION STRING EXPECTED
221	STATION OR ENSEMBLE NOT ASSIGNABLE TYPE
222	STATION ALREADY ASSIGNED
223	ASSIGNED FUNCTION NOT IMPLEMENTED
224	TOO MANY ALLC REFERENCES
225	TOO MANY TARGETS
226	ALLC-REFERENCE INVALID
227	INVALID REPLACE OPTION
228	ILLEGAL FUNCTION REQUESTED
229	LENGTH OF TEST FRAME TOO LONG
230	MAXIMUM SEGMENT SIZE MAY NOT BE LESS THAN NETWORK MAX SEG SIZE
231	MAXIMUM SEGMENT SIZE IS TOO LARGE

Table 12–68. NSM Error Codes

Number	Description
1	INVALID PRIORITY
2	INVALID LEVEL
3	INVALID FRAME TYPE
4	TRANSIT COUNT LIMIT EXCEEDED
5	UNKNOWN ORIGIN NODE ADDRESS
6	UNKNOWN DESTINATION NODE ADDRESS
7	DESTINATION NODE UNREACHABLE
8	INVALID CONTROL TYPE
9	UNKNOWN SUBJECT NODE ADDRESS
10	INVALID HOP COUNT
11	INVALID RESISTANCE FACTOR
12	INVALID CAUSE
13	INVALID MAXIMUM HOP COUNT
14	INVALID MAXIMUM RESISTANCE FACTOR
15	INVALID RELEASE LEVEL
16	INVALID TRACE REFERENCE
17	UNKNOWN RECEIVING NODE ADDRESS
18	UNKNOWN NEIGHBOR NODE ADDRESS
19	UNKNOWN PDN-ID
20	INVALID MAXIMUM SEGMENT SIZE
21	INVALID STATION-QUEUE-ID
22	UNKNOWN STATION-QUEUE-ID
23	INVALID LINK-PENDING VALUE
24	INVALID REMOTE RESISTANCE FACTOR
25	INVALID LOCAL RESISTANCE FACTOR
26	INVALID OPERATIONS RESISTANCE FACTOR
27	INVALID PHYSICAL LINK STATION-ID
28	INVALID PHYSICAL LINK VAN-ID
29	INVALID PHYSICAL LINK SPEED
30	INVALID PHYSICAL LINK EFFICIENCY
31	INVALID PHYSICAL LINK MAXIMUM SEG SIZE

continued

Table 12–68. NSM Error Codes (cont.)

Number	Description
32	EXCEEDED MAXIMUM SEGMENT SIZE FOR LINK
33	NODE ALREADY EXISTS
34	INVALID NODE ADDRESS
35	EXCEEDED MAXIMUM NUMBER OF PARALLEL LINKS
36	STATION IS ALREADY ATTACHED
37	UNKNOWN ATTRIBUTE
38	ATTRIBUTE IS NOT SETTABLE
39	ILLEGAL ATTRIBUTE VALUE
40	INITIALIZATION SEQUENCE OUT OF PHASE
41	INCOMPATIBLE ROUTER PROTOCOL LEVEL
42	SOURCE NODE UNREACHABLE
43	NODE IS A NEIGHBOR
44	NODE IS UNKNOWN
45	INVALID FRAME LENGTH
46	NETWORK IS CLOSING
47	STATION NOT ATTACHED

Table 12–69. Station-Level Reason Codes

Number	Description
1	CONNECTION PORT DIALOG ALREADY OPEN
2	CONNECTION PORT DIALOG ALREADY CLOSED
3	CONNECTION PORT DIALOG PENDING OPEN
4	CONNECTION PORT DIALOG PENDING CLOSED
5	STATION DIALOG NOT CLOSED
6	CONNECTION PORT DIALOG NOT OPEN
7	STATION DIALOG NOT OPEN
8	STATION DIALOG ALREADY CLOSED
9	STATION DIALOG PENDING OPEN
10	STATION DIALOG PENDING CLOSED
11	CONNECTION PORT INVALID STATE

continued

Table 12–69. Station-Level Reason Codes (cont.)

Number	Description
12	DSR RESPONSE TIMER TIMEOUT
13	CTS RESPONSE TIMER TIMEOUT
14	UNEXPECTED DSR OFF
15	UNEXPECTED CTS OFF
16	DCD OFF DELAY TIMER TIMEOUT
17	DIAL HANDLER INVALID STATE
18	NO TELEPHONE NUMBER
19	UNEXPECTED PWI OFF
20	UNEXPECTED COS ON
21	UNEXPECTED DLO OFF
22	ACU RESPONSE TIMER TIMEOUT
23	ACR ON
24	RETRY COMPLETED
25	TEXT TOO LONG
26	CONNECTION PORT DIALOG CLOSED
27	TEST SENDER NOT RESET
28	FRMR RECEIVED
29	RETRY COUNT EXCEEDED FOR SABM
30	DM RECEIVED
31	DISC-1 RECEIVED
32	DISC-2 RECEIVED
33	DIAL RESPONSE TIMER TIMEOUT
34	RETRY COUNT EXCEEDED FOR DISC
35	F-RESPONSE TIMER TIMEOUT
36	NO NAME, VALUE
37	NO NAME
38	COMMAND NOT IMPLEMENTED
39	INVALID
40	CURRENT STATE OF CPDS-1
41	SYSTEM HAS NO SHARED GLOBAL MEMORY
42	INSUFFICIENT SHARED GLOBAL MEMORY

continued

Table 12–69. Station-Level Reason Codes (cont.)

Number	Description
43	CLOSE STATION DIALOG RECEIVED
44	STATION-LEVEL SOFTWARE FAILURE
45	DCWRITE ERROR
46	SLM SENDER NOT RESET
47	CP DIALOG PEND/OPEN, PEND/CLOSED
48	NSP (DCC) DSED
49	DATA COMM ERROR
50	STATION DIALOG ALREADY OPEN
51	RETRY COUNT EXCEEDED
52	UNEXPECTED UA RECEIVED
53	STATION LOST CARRIER
54	GMM FRAME XMIT FAILED
55	UNEXPECTED DM RECEIVED
56	ILLEGAL GMM STATION
57	RETRY COUNT EXCEEDED FOR FRMR
58	UNEXPECTED F=1 RECEIVED
59	DATA COMM ADAPTER FAULT
60	ISC DISC1 RECEIVED
61	ISC RETRY COUNT EXCEEDED FOR LR
62	ISC UNEXP UA RECEIVED
63	ISC MODE NOT OPEN
64	ISC STA DIALOG CLOSED
65	ISC HUB INITIALIZATION FAILED
66	ISC FATAL HC ERROR
67	ISC HUB MANAGER DSED
68	ISC UNEXPECTED SD CLOSE
69	ISC LOCAL HC ERROR
70	REMOTE HOST UNRESPONSIVE
71	CP CLOSED AT REMOTE HOST
72	ISC REMOTE HC ERROR
73	ISCUPDATEPATHS ERROR

continued

Table 12–69. Station-Level Reason Codes (cont.)

Number	Description
74	ACTIVE STATION ON HUB TO NEIGHBOR
75	REMOTE INCONSISTENT AMR INFO
76	RESUME READY
77	GLOBAL MEMORY STATION ENGAGED
78	STATION GROUP NOT OPEN

Table 12–70. Station-Level Reports

Number	Description
1	STATION DIALOG CLOSED
2	CONNECTION PORT DIALOG CLOSED
3	TEST COMMAND RECEIVED
4	FRAME RECEIVED (SLM)
5	INCOMING CALL
6	AWAIT CARRIER
7	I RESPONSE RECEIVED
8	CP DIALOG OPEN/CLOSED
9	INVALID BDLC ADDRESS
10	BDLC FRAME RECEIVED NOT OCTET MULTIPLE
11	BDLC FRAME TOO SHORT
12	UNEXPECTED CP DIALOG CLOSED
13	NON BNA VERSION 1 CALLER
14	LINK RESET LOCALLY
15	FRAME REJECT RECEIVED
16	RETRY COUNT EXCEEDED (OSD)
17	LINK RESET REMOTELY
18	FRAME REJECT SENT
19	UNEXPECTED DM/DISC RECEIVED
20	STATION DIALOG REOPENED
21	REQUEUE UNACK FRAME ERROR

continued

Table 12–70. Station-Level Reports (cont.)

Number	Description
22	FRAME TRANSMISSION ERROR
23	STATION ATTACH ERROR
24	REMOTE BUSY SET
25	REMOTE BUSY RESET
26	DM RECEIVED(OSD)
27	DISC RECEIVED(OSD)
28	MSG SIZE EXCEEDS CB AREA
29	BDLC FRAME NOT OCTET MULTIPLE (NO FRAME)
30	BDLC FRAME TOO LONG
31	PENDING RECV LR
32	ISC SHUTDOWN IN PROGRESS
33	X.25 PVC STATION AWAITING OPEN
34	ERROR
35	RECEIVED ROUTER LIU
36	RECEIVED RESTART
37	DIAGNOSTIC PACKET
38	CLEAR INDICATION RECEIVED
39	RETURN REQUESTED BY OI
40	QUERY TIMER EXPIRED
41	MAX INFO SIZE FIELD INCOMPATIBLE
42	SCM UNABLE TO CONTINUE
43	NO TARGET DIRECTORY ENTRY FOUND
44	PORT FILE WRITE ERROR
45	SUBFILE DEACTIVATED
46	GO AWAY RECEIVED
47	NO ALLC AVAILABLE FOR TARGETS
48	SCM ERROR
49	ALLC TIMER EXPIRED
50	INVALID DLL ID FRAME
51	UNRECOGNIZED ALLC FRAME
52	NOTASKAVAILABLE

continued

Table 12–70. Station-Level Reports (cont.)

Number	Description
53	OPENPORTTOALLCFAILURE
54	ALLCPORTBLOCKED
55	NEGATIVERESPONSETOOSD
56	AWAIT DATA SET READY

Table 12–71. Station Types

Number	Description
1	BDLC DEDICATED
2	BDLC SWITCHED WITH ACU
3	BDLC SWITCHED WITHOUT ACU
4	X.25 GROUP
5	X.25 VC
6	X.25 PVC
7	ISC GROUP
8	ISC STATION
9	GLOBAL MEMORY
10	BLDC SWITCHED MANUAL

Table 12–72. X-25 PDNs

Number	Description
0	NULL
1	DATAPAC
2	TRANSPAC
3	DDX
4	PSS
5	TELENET
6	TYMNET
7	INFOSWITCH

continued

Table 12–72. X-25 PDNs (cont.)

Number	Description
8	DATANET-1
9	DATEX-PIO
10	SAPONET
11	SCT
12	TELEPAC
13	BX.25
14	RETD
15	RTTX.25
16	AUSTPAC

The log entries in Table 12–73, "Major Type 11–Minor Types 1 through 4," are written when the value of a BNA Version 1 attribute is set by an operator command. The four types of log entries correspond to four different BNA Version 1 modules. These entries are written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–73. Major Type 11–Minor Types 1 through 4

Minor Type = 1 Set Network-Services-Manager (NSM) Attribute (LOGNSMATT)
Minor Type = 2 Set Port-Level-Manager (PLM) Attribute (LOGPLMATT)
Minor Type = 3 Set Router Attribute (LOGROUTATT)
Minor Type = 4 Set Station-Level-Manager (SLM) Attribute (LOGSLMATT)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to attribute name (bytes)
5	Attribute value
	[47: 2] Attribute type
	0 = INTEGER
	1 = STRING
	2 = BOOLEAN

continued

Table 12–73. Major Type 11–Minor Types 1 through 4 (cont.)

Minor Type = 1 Set Network-Services-Manager (NSM) Attribute (LOGNSMATT) **Minor Type = 2 Set Port-Level-Manager (PLM) Attribute (LOGPLMATT)** **Minor Type = 3 Set Router Attribute (LOGROUTATT)** **Minor Type = 4 Set Station-Level-Manager (SLM) Attribute (LOGSLMATT)**	
Words	**Description**
	[39:40] LINK to attribute value. The length of string-valued attributes is in bytes; other attributes are stored in one word.
6	For Minor Types 1, 3, and 4, this word is reserved. For Minor Type 2, this word holds a network services (NS) error code.
7–end	Variable-length data pointed to by the LINK in this entry

For more information, see Table 12–67, "NS Error Codes."

The log entry described in Table 12–74, "Major Type 11–Minor Type 5," is written when the phase of the NSM changes during network initialization and when the network is shutting down.

Table 12–74. Major Type 11–Minor Type 5

Minor Type = 5 Network Services Manager Phase Change (LOGNSMPHASE)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	New Phase 2 = STARTNET 3 = DATA-ENTRY 4 = WAIT-STATION 5 = WAIT-ROUTER 6 = WAIT-PORT

continued

Table 12–74. Major Type 11–Minor Type 5 (cont.)

Minor Type = 5 Network Services Manager Phase Change (LOGNSMPHASE)	
Words	**Description**
	7 = NORMAL-OPERATION
	11 = SLOW-SHUTDOWN
	12 = FAST-SHUTDOWN
	13 = ROUTER-SHUTDOWN
	14 = STATION-SHUTDOWN

The log entries in Table 12–75, "Major Type 11–Minor Types 6 and 7," are written when remote nodes and hosts are added to BNA Version 1 tables. Nodes and hosts can be added in response to operator commands, or can be added automatically as the local node learns of the existence of other nodes and hosts in the network. These entries are written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–75. Major Type 11–Minor Types 6 and 7

Minor Type = 6 Add Host (LOGADDHOST) **Minor Type = 7 Add Node (LOGADDNODE)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to hostname (bytes)
5	Host information:
	[33: 1] Reachability
	0 = Not reachable
	1 = Reachable
	[32: 1] Translate-only flag
	[31:16] Port maximum segment size

continued

Table 12–75. Major Type 11–Minor Types 6 and 7 (cont.)

Minor Type = 6 Add Host (LOGADDHOST) **Minor Type = 7 Add Node (LOGADDNODE)**	
Words	**Description**
	[15:16] Node address
6	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
7–end	Variable-length data pointed to by the LINK in this entry

The log entry in Table 12–76, "Major Type 11–Minor Type 8," is written when a remote host is cleared, saved, or made ready in response to the corresponding operator command. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–76. Major Type 11–Minor Type 8

Minor Type = 8 Host Status (LOGHOSTSTAT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to hostname (bytes)
5	Host status: [1: 2] Request 0 = Save host 1 = Ready host 2 = Clear host
6	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
7–end	Variable-length data pointed to by the LINK in this entry

The log entries in Table 12–77, "Major Type 11–Minor Types 9 and 10," are written when remote nodes and hosts are deleted from BNA Version 1 tables. Nodes and hosts are deleted in response to the DELETE HOST or DELETE NODE operator command. These log entries are written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–77. Major Type 11–Minor Types 9 and 10

Minor Type = 9 Delete Host (LOGDELHOST)
Minor Type = 10 Delete Node (LOGDELNODE)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to hostname (bytes)
5	Host information: [16: 1] Name translation 0 = Host name deleted 1 = Host name retained for address translation only [15:16] Node address
6	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
7–end	Variable-length data pointed to by the LINK in this entry

The log entry in Table 12–78, "Major Type 11–Minor Type 11," is written when the port-level-manager (PLM) module detects an error, especially in an incoming port frame. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–78. Major Type 11–Minor Type 11

Minor Type = 11 Port-Level-Manager (PLM) Error Report (LOGPLMERR)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Reserved
5	LINK to error report (bytes)
6–end	Variable-length data pointed to by the LINK in this entry

The log entry in Table 12–79, "Major Type 11–Minor Type 12," is written when the port-level-manager (PLM) module detects certain conditions and events. These are described below. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–79. Major Type 11–Minor Type 12

Minor Type = 12 Port-Level-Manager (PLM) Log Report (LOGPLMRPT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Report information
	[31:16] Report type
	Only one of the bits in this field can be set. It shows the type of PLM report.
	[21: 1] PLM-ID frame sent
	[20: 1] Received ACCEPT match frame for nonexistent port
	[19: 1] Received port frame larger than maximum segment size
	[18: 1] Valid PLM-ID frame received

continued

Table 12–79. Major Type 11–Minor Type 12 (cont.)

Minor Type = 12 Port-Level-Manager (PLM) Log Report (LOGPLMRPT)	
Words	**Description**
	[17: 1] PLM dialog terminated
	[16: 1] PLM dialog established
	[15:16] Remote node address
The format of the remainder of the entry depends on the report type.	
PLM dialog established and PLM dialog terminated:	
5	LINK to remote host name (bytes)
6	Local subport index
7	PLM dialog termination reason
	(Dialog termination report only)
	1 = Inactivity
	2 = Inactivity: network shutdown
	3 = Inactivity: host saved
	4 = Host deleted
	5 = Loss of validation
	6 = Inactivity requested
	7 = Demand received
	8 = Deactivation
	9 = Failure to establish PLM dialog
8–end	Variable-length data pointed to by the LINK in this entry
Valid PLM-ID frame received:	
5	LINK to remote host name (bytes)
6	LINK to remote incarnation ID (bytes)
7–end	Variable-length data pointed to by the LINKs in this entry
ACCEPT frame for nonexistent port:	
5	LINK to text of ACCEPT frame (bytes)

continued

Table 12–79. Major Type 11–Minor Type 12 (cont.)

Minor Type = 12 Port-Level-Manager (PLM) Log Report (LOGPLMRPT)	
Words	**Description**
6–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–80, "Major Type 11–Minor Type 13," is written when the resistance factor to a neighbor node is changed in response to the LINKRF operator command. It is also written when the node resistance factor used at the local node is changed in response to the NODERF operator command. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–80. Major Type 11–Minor Type 13

Minor Type = 13 Resistance Factor Change (LOGROUTRFCHG)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Node address (zero indicates local node)
5	Resistance factor
6	Network services manager (NSM) error code (See Table 12–68, "NSM Error Codes.")

The log entry in Table 12–81, "Major Type 11–Minor Type 14," is written when a Routing Trace message is sent to a remote node in response to the STARTTRACE operator command. This entry is written only when the LOGGING attribute is set to the value MAXIMUM.

Table 12–81. Major Type 11–Minor Type 14

Minor Type = 14 Start Trace (LOGROUTSTTRA)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Trace source node address
5	Trace destination node address
6	Trace reference
7	Network services (NS) error code (See Table 12–67, "NS Error Codes.")

The log entry in Table 12–82, "Major Type 11–Minor Type 15," is written when the Routing Refresh function is performed in response to the ROUTINGREFRESH operator command. This entry is written only when the LOGGING attribute is set to the value MAXIMUM.

Table 12–82. Major Type 11–Minor Type 15

Minor Type = 15 Routing Refresh (LOGROUTREFRESH)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
(No additional fields)	

The log entries in Table 12–83, "Major Type 11–Minor Types 16 and 17," are written when router control frames are sent and received by the local node. Router control frames include LINKCHANGE and NETCHANGE messages that are used to inform neighbor nodes about changes in the topology of the network. The Router Control Frame Sent (Type 16) entry is written only when the LOGGING attribute is set to the value MAXIMUM. The Router Control Frame Received (Type 17) entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–83. Major Type 11–Minor Types 16 and 17

Minor Type = 16 Router Control Frame Sent (LOGROUTCFSEND) **Minor Type = 17 Router Control Frame Received (LOGROUTCFRECV)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to remote hostname (bytes)
5	LINK to text of frame (bytes)
6–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–84, "Major Type 11–Minor Type 18," is written when the Router detects an error in an incoming Router frame. This entry is written only when the LOGGING attribute is set to at least the value MINIMUM.

Table 12–84. Major Type 11–Minor Type 18

Minor Type = 18 Router Frame Error (LOGROUTERR)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Destination node address
5	Origin node address
6	Error information [15: 8] Network services manager (NSM) error code (See Table 12–68, "NSM Error Codes.") [07: 8] Router error type

continued

Table 12–84. Major Type 11–Minor Type 18 (cont.)

Minor Type = 18 Router Frame Error (LOGROUTERR)	
Words	**Description**
	1 = NSM-to-NSM error
	2 = Link-Change error
	3 = Net-Change error
	4 = Trace-Start error
	5 = Trace error
	6 = Trace-Result error
	7 = Reserved
	8 = Router header error
	9 = Router transit error
7	LINK to frame text (bytes)
8	LINK to destination host name (bytes)
9	LINK to origin host name (bytes)
10–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–85, “Major Type 11–Minor Type 19,” is written when the routing status to a remote node in the network changes. The entry is generated when at least one of the following routing factors has changed:

- Reachability
- Neighbor node address
- Router and Port maximum segment size (these always change together)

Other routing factors, such as hop-count and resistance-factor, might also have changed.

This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–85. Major Type 11–Minor Type 19

Minor Type = 19 Destination Node Status Change (LOGROUTDNASTAT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Destination node address
5	Neighbor node address
	This field indicates the neighbor node through which traffic is routed when the local node is sending frames to the destination node.
6	Routing status information #1
	[47:16] Resistance factor
	[31:16] Port frame maximum segment size
	[15:16] Router frame maximum segment size
7	Routing status information #2
	[47:06] Routing status change flags
	0 = Attribute did not change
	1 = Attribute changed
	[47:1] Neighbor node address
	[46:1] Router frame maximum segment size
	[45:1] Port frame maximum segment size
	[44:1] Reachability
	[43:1] Resistance factor
	[42:1] Hop count
	[16: 1] Node reachability
	[15:16] Hop count
8	LINK to destination host name (bytes)
9	LINK to neighbor host name (bytes)
10–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–86, "Major Type 11–Minor Type 20," is written for every incoming and outgoing Router frame when the COPY option has been set through the MONITOR operator command. This entry is written only when the LOGGING attribute is set to at least the value MINIMUM.

Table 12–86. Major Type 11–Minor Type 20

Minor Type = 20 Router Monitor Frame Copy (LOGROUTCOPYRPT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Destination node address
5	Origin node address
6	LINK to frame text (bytes)
7–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–87, "Major Type 11–Minor Type 21," is written periodically only when the TRAFFIC option has been set by the MONITOR operator command. The frequency with which this entry is generated is controlled by the Router traffic monitor interval, which is also set by the MONITOR command. This entry is written only when the LOGGING attribute is set to at least the value MINIMUM.

Table 12–87. Major Type 11–Minor Type 21

Minor Type = 21 Router Monitor Traffic Summary (LOGROUTSUMRPT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to traffic summary data (words) For more information about Word 4, see the text immediately following this table.
5	Reserved
6–end	Variable-length data pointed to by the LINK in this entry

The traffic summary data (Word 4) are composed of a series of 7-word entries, each containing a summary of the traffic that passed through the local node in transit from a particular origin node/destination node pair. Each entry has the following format:

Word	Contents
1	Node addresses: zero indicates the local node
	[31:16] destination node
	[15:16] origin node
2–7	Three entries, each two words in length: the first word in each entry shows the number of frames of a particular type, and the second word shows the number of bytes of data in those frames.
2	Number of information frames
3	Bytes of information data
4	Number of control frames
5	Bytes of control data
6	Number of frames with errors
7	Bytes of error data

The log entry in Table 12–88, "Major Type 11–Minor Type 24," is written when the Router adds or deletes a node to or from its tables. This can be in response to an ADD NODE, DELETE NODE, ADD HOST, or DELETE HOST operator command, or in response to information received from other nodes in the network. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–88. Major Type 11–Minor Type 24

Minor Type = 24 Router Node Existence (LOGROUTNODEEXIST)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Node information [16: 1] Operation 0 = Add node 1 = Delete node [15:16] Node address
5	Network services manager (NSM) error code (See Table 12–68, "NSM Error Codes.")

The log entry in Table 12–89, "Major Type 11–Minor Type 25," is written when one or more attributes of a BNA Version 1 station are modified in response to a MODIFY STATION operator command. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–89. Major Type 11–Minor Type 25

Minor Type = 25 Modify Station (LOGSLMMODIFYSTA)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	LINK to attribute list (words)
	Each word in the attribute list has the following structure:
	[47: 8] Station attribute number
	1 = Station-Type
	2 = Hardware-ID
	3 = Autoinit
	4 = Efficiency
	5 = Maximum-Segment-Size
	6 = Speed
	7 = Monitor
	8 = Hub-Number (ISC)
	11 = Windowsize (ISC)
	12 = Retry-Count-limit (ISC)
	13 = FRSP-Timer (ISC)
	14 = Checkpoint-Timer (ISC)
	15 = Maximum-Frame-Size (ISC)
	17 = First-PVC (X.25 Group)
	18 = Last-PVC (X.25 Group)
	19 = First-VC-IN (X.25 Group)
	20 = Last-VC-IN (X.25 Group)
	21 = First-VC-IO (X.25 Group)
	22 = Last-VC-IO (X.25 Group)

continued

Table 12–89. Major Type 11–Minor Type 25 (cont.)

Minor Type = 25 Modify Station (LOGSLMMODIFYSTA)	
Words	**Description**
	23 = First-VC-OUT (X.25 Group)
	24 = Last-VC-OUT (X.25 Group)
	25 = Restart-Retry-Count-Limit (X.25 Group)
	26 = Restart-Response-Timer-Limit (X.25 Group)
	27 = Timer-Limit-Value-T31 (X.25 Station)
	28 = Timer-Limit-Value-T32 (X.25 Station)
	29 = Timer-Limit-Value-T33 (X.25 Station)
	30 = LC-Retry-Count-Limit-1 (X.25 Station)
	31 = LC-Retry-Count-Limit-2 (X.25 Station)
	32 = LC-Local-Window-Size (X.25 Station)
	33 = LC-Packet-Size (X.25 Station)
	34 = Public-Data-Network (X.25 Group)
	35 = Modified (X.25 Group)
	41 = Modules (X.25 Group)
	[39:40] Attribute value
	All attributes are integer valued except for the following:
	Station type (#1): enumerated (See Table 12–71, "Station Types.")
	Autoinit (#3): Boolean
	0 = FALSE
	1 = TRUE
	Monitor (#7): enumerated
	0 = OFF
	1 = ON
	Public Data Network (#34): enumerated (See Table 12–72, "X–25 PDNs.")
	Modified (#35): Boolean
	0 = FALSE

continued

Table 12–89. Major Type 11–Minor Type 25 (cont.)

Minor Type = 25 Modify Station (LOGSLMMODIFYSTA)	
Words	**Description**
	1 = TRUE
6	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
7–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–90, "Major Type 11–Minor Type 26," is written when a BNA Version 1 station is deleted in response to a DELETE STATION operator command. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–90. Major Type 11–Minor Type 26

Minor Type = 26 Delete Station (LOGSLMDELSTA)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	Reserved
6	Network services (NS) error code
7–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–91, "Major Type 11–Minor Type 27," is written when an ensemble of BNA Version 1 stations is created in response to an ADD ENSEMBLE operator command. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–91. Major Type 11–Minor Type 27

Minor Type = 27 Add Ensemble (LOGSLMADDENS)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to ensemble name (bytes)
5	Direction 1 = PERMANENT 2 = INCOMING 3 = OUTGOING
6	LINK to station list (words) Each entry in the list is composed of a 1-byte field showing the length (in bytes) of the station name, followed by the name, and (if needed) filler to a word boundary.
7	Network services (NS) error code
8–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–92, "Major Type 11–Minor Type 28," is written when one or more BNA Version 1 stations are added to, or deleted from, an ensemble in response to a MODIFY ENSEMBLE operator command. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–92. Major Type 11–Minor Type 28

Minor Type = 28 Modify Ensemble (LOGSLMMODIFYENS)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to ensemble name (bytes)
5	Add or delete flag
	0 = Delete stations
	1 = Add stations
6	LINK to station list (words)
	Each entry in the list is composed of a 1-byte field showing the length (in bytes) of the station name, followed by the name, and (if needed) filler to a word boundary.
7	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
8–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–93, "Major Type 11–Minor Type 29," is written when an ensemble of BNA Version 1 stations is deleted in response to a DELETE ENSEMBLE operator command. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–93. Major Type 11–Minor Type 29

Minor Type = 29 Delete Ensemble (LOGSLMDELENS)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to ensemble name (bytes)
5	Reserved

continued

Table 12–93. Major Type 11–Minor Type 29 (cont.)

Minor Type = 29 Delete Ensemble (LOGSLMDELENS)	
Words	**Description**
6	Reserved
7	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
8–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–94, "Major Type 11–Minor Types 30 and 31," is written when the description of a connection to a neighbor node is added or modified in response to an ADD CONNECTION or MODIFY CONNECTION operator command. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–94. Major Type 11–Minor Types 30 and 31

Minor Type = 30 Add Connection (LOGSLMADDCONN) **Minor Type = 31 Modify Connection (LOGSLMMODIFYCONN)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Neighbor node address
5	Direction 1 = PERMANENT 2 = INCOMING 3 = OUTGOING
6	Station/ensemble name [47: 1] Station or ensemble 0 = Ensemble name 1 = Station name [39:40] LINK to name (bytes)

continued

Table 12–94. Major Type 11–Minor Types 30 and 31 (cont.)

Minor Type = 30 Add Connection (LOGSLMADDCONN) **Minor Type = 31 Modify Connection (LOGSLMMODIFYCONN)**	
Words	**Description**
7	LINK to CALLDATA (bytes) (OUTGOING only)
8	INITQUANTITY (OUTGOING only) The value –1 represents an empty field; that is, the command did not cause the value of INITQUANTITY to be set.
9	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
10	LINK to neighbor host name (bytes)
11–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–95, "Major Type 11–Minor Type 32," is written when the description of a connection to a neighbor node is deleted in response to a DELETE CONNECTION operator command. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–95. Major Type 11–Minor Type 32

Minor Type = 32 Delete Connection (LOGSLMDELCONN)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Neighbor node address
5	Direction 1 = PERMANENT 2 = INCOMING 3 = OUTGOING
6	Station/ensemble name

continued

Table 12–95. Major Type 11–Minor Type 32 (cont.)

Minor Type = 32 Delete Connection (LOGSLMDELCONN)	
Words	**Description**
	[47: 1] Station or ensemble 0 = Ensemble name 1 = Station name [39:40] LINK to name (bytes)
7	Reserved
8	Reserved
9	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
10	LINK to neighbor host name (bytes)
11–end	Variable-length data pointed to by the LINK in this entry

The log entry in Table 12–96, "Major Type 11–Minor Type 33," is written when one or more BNA Version 1 stations are closed upon receipt of a CLEARCALL operator command. The information in the entry shows whether the command requested that a particular station be cleared or if the command specified an ensemble of stations or neighbor node. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–96. Major Type 11–Minor Type 33

Minor Type = 33 Clear Call (LOGSLMCLEARCALL)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Neighbor node address
5	Station/ensemble name [47: 1] Station or ensemble

continued

Table 12–96. Major Type 11–Minor Type 33 (cont.)

Minor Type = 33 Clear Call (LOGSLMCLEARCALL)	
Words	**Description**
	0 = Ensemble name
	1 = Station name
	[39:40] LINK to name (bytes)
6	Reserved
7	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
8–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–97, "Major Type 11–Minor Type 34," is written when one or more BNA Version 1 stations are opened upon receipt of an ESTABLISHCALL operator command. The information in the entry shows whether the command requested that a particular station be opened or if the command specified an ensemble of stations. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–97. Major Type 11–Minor Type 34

Minor Type = 34 Establish Call (LOGSLMESTCALL)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Neighbor node address
5	Station/ensemble name
	[47: 1] Station or ensemble
	0 = Ensemble name
	1 = Station name
	[39:40] LINK to name (bytes)

continued

Table 12–97. Major Type 11–Minor Type 34 (cont.)

Minor Type = 34 Establish Call (LOGSLMESTCALL)	
Words	**Description**
6	LINK to CALLDATA (bytes)
7	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
8–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–98, "Major Type 11–Minor Type 35," is written when one or more BNA Version 1 stations are opened upon receipt of an AWAITCALL operator command. The information in the entry shows whether the command requested that a particular station be opened or if the command specified an ensemble of stations. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–98. Major Type 11–Minor Type 35

Minor Type = 35 Await Call (LOGSLMAWAITCALL)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Station/ensemble name [47: 1] Station or ensemble 0 = Ensemble name 1 = Station name [39:40] LINK to name (bytes)
5	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
6–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–99, "Major Type 11–Minor Type 36," is written when a test frame is sent on a BNA Version 1 station in response to a SENDTEST operator command. This entry is written only when the LOGGING attribute is set to the value MAXIMUM.

Table 12–99. Major Type 11–Minor Type 36

Minor Type = 36 Send Test (LOGSLMSENDTEST)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	LINK to text (bytes)
6	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
7–end	Variable-length data pointed to by the LINKs in this entry

The log entries in Table 12–100, "Major Type 11–Minor Types 37 and 38," are written when a test frame or test response frame is received on a BNA Version 1 station. These entries are written only when the LOGGING attribute is set to the value MAXIMUM.

Table 12–100. Major Type 11–Minor Types 37 and 38

Minor Type = 37 Test Received (LOGTESTCOMMRECV) **Minor Type = 38 Test Response Received (LOGTESTRESPRECV)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	LINK to text (bytes)
6–end	Variable-length data pointed to by the LINKs in this entry

Table 12–101. Major Type 11–Minor Type 39

Minor Type = 39 Link Reset (LOGSLMLINKRESET)	
Words	**Description**
This log-entry type is not currently used.	

The log entry in Table 12–102, "Major Type 11–Minor Type 40," is written when an Open-Connection-Port-Dialog command is sent to a BNA Version 1 station. This command is a manual BNA Version 1 used to control the opening of a station. It can also be initiated by the operator command ESTABLISHCALL or AWAITCALL, or by setting the AUTOINIT attribute for that station. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–102. Major Type 11–Minor Type 40

Minor Type = 40 Open Connection Port (LOGSLMOPENCPD)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	LINK to CALLDATA (bytes)
6	Station-level reason code (See Table 12–69, "Station-Level Reason Codes.")
7–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–103, "Major Type 11–Minor Type 41," is written when a Close-Connection-Port-Dialog command is sent to a BNA Version 1 station. This command is a manual BNA Version 1 command used to control the closing of a station. It can also be initiated by the CLEARCALL operator command, and by the shutdown of the network. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–103. Major Type 11–Minor Type 41

Minor Type = 41 Close Connection Port (LOGSLMCLOSECPD)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	Station-level reason code (See Table 12–69, "Station-Level Reason Codes.")
6–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–104, "Major Type 11–Minor Type 42," is written when a VALIDATEANDATTACH operator command is received for a BNA Version 1 station. It indicates that the operator, who is opening the station manually, wants BNA Version 1 to initiate the exchange of station-level greetings with the remote node. If successful, the station is attached to the Router level at the local node and the station becomes operational. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–104. Major Type 11–Minor Type 42

Minor Type = 42 Validate and Attach Station (LOGSLMVALATT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
6–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–105, “Major Type 11–Minor Type 43,” is written when a DETACH operator command is received for a BNA Version 1 station. It indicates that the operator, who is manually closing the station, wants BNA Version 1 to detach the station from the Router and close the Station Dialog with the remote node. This entry is written only when the LOGGING attribute is set to at least the value MINIMUM.

Table 12–105. Major Type 11–Minor Type 43

Minor Type = 43 Detach Station (LOGSLMMANDETACH)	
Words	**Description**
0–3	As described in Table 12–2, “First Four Log Entry Words”
4	LINK to station name (bytes)
5	Network services (NS) error code (See Table 12–67, “NS Error Codes.”)
6–end	Variable-length data pointed to by the LINKs in this entry

The log entries in Table 12–106, “Major Type 11–Minor Types 44 and 45,” are written when a SAVE STATION or READY STATION operator command is received for a BNA Version 1 station. It indicates that the operator wants BNA Version 1 to make the station SAVED or READY, respectively. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–106. Major Type 11–Minor Types 44 and 45

Minor Type = 44 Save Station (LOGSLMSAVESTA) Minor Type = 45 Ready Station (LOGSLMREADYSTA)	
Words	**Description**
0–3	As described in Table 12–2, “First Four Log Entry Words”
4	LINK to station name (bytes)
5	Network services (NS) error code (See Table 12–67, “NS Error Codes.”)
6–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–107, "Major Type 11–Minor Type 46," is written when an Open-Station-Dialog command is sent to a BNA Version 1 station. This is a manual BNA Version 1 command used to control the opening of stations. It can also be initiated by the ESTABLISHCALL or AWAITCALL operator commands. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–107. Major Type 11–Minor Type 46

Minor Type = 46 Open Station Dialog (LOGSLMOPENSD)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	Call data value
6	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
7–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–108, "Major Type 11–Minor Type 47," is written when a Close-Station-Dialog command is sent to a BNA Version 1 station. This command is a manual BNA Version 1 command used to control the closing of stations. It can also be initiated by the CLEARCALL operator command. These entries are written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–108. Major Type 11–Minor Type 47

Minor Type = 47 Close Station Dialog (LOGSLMCLOSESD)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	Network services (NS) error code (See Table 12–67, "NS Error Codes.")
6–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–109, "Major Type 11–Minor Type 48," is written to report that a BNA Version 1 station has successfully initialized a link and is now operational. It is generated after the station exchanges station-level greetings with the remote node and attaches to the Router level at the local node. This entry is also generated when a station successfully reinitializes a link after a Link-Reset condition. This entry is written only when the LOGGING attribute is set to at least the value MINIMUM.

Table 12–109. Major Type 11–Minor Type 48

Minor Type = 48 Station Attach Report (LOGSLMATTACH)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	Neighbor node address
6	Public Data Network ID (X.25)
7	Link information: [47: 8] Link efficiency [39:24] Link speed [15:16] Working link maximum segment size
8	LINK to neighbor host name (bytes)
9–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–110, "Major Type 11–Minor Type 49," is written to report that a BNA Version 1 station has detached from the Router level at the local node and is no longer operational. This occurrence generally indicates that the station is about to be closed. This entry is written only when the LOGGING attribute is set to at least the value MINIMUM.

Table 12–110. Major Type 11–Minor Type 49

Minor Type = 49 Station Detach Report (LOGSLMDETACH)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	Neighbor node address
6	LINK to neighbor host name (bytes)
7–end	Variable-length data pointed to by the LINKs in this entry

Table 12–111. Major Type 11–Minor Type 50

Minor Type = 50 Neighbor Restart (LOGSLMNEIRESTART)	
Words	**Description**
This log-entry type is not currently used.	

Table 12–112. Major Type 11–Minor Type 51

Minor Type = 51 Neighbor Busy (LOGSLMNEIBUSY)	
Words	**Description**
This log-entry type is not currently used.	

The log entry in Table 12–113, "Major Type 11–Minor Type 52," is written to report that BNA Version 1 has detected one of a certain set of conditions on a particular station. The various conditions that can be detected are listed in Table 12–70, "Station-Level Reports." This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–113. Major Type 11–Minor Type 52

Minor Type = 52 Station Log Report (LOGSLMLOGRPT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	Neighbor node address
6	Report information: [47: 1] Reserved [46: 1] Validity of data link control (BDLC) address and control fields (see below) 0 = BDLC fields not present 1 = BDLC fields present [45: 6] Reserved [39: 8] Station Type (See Table 12–71, "Station Types.") [31: 8] BDLC address field [23: 8] BDLC control field [15: 8] Reserved [7: 8] Station-level report (See Table 12–70, "Station-Level Reports.")
7	LINK to report text (bytes)
8	Station-level reason code (See Table 12–69, "Station-Level Reason Codes.")
9	Public Data Network (X.25) (See Table 12–72, "X–25 PDNs.")
10–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–114, "Major Type 11–Minor Type 53," is written to report that a BNA Version 1 station has failed the exchange of station-level greetings. This exchange is part of the initialization process when opening a station and making it operational. This entry is written only when the LOGGING attribute is set to at least the value MINIMUM.

Table 12–114. Major Type 11–Minor Type 53

Minor Type = 53 Station Validation Failure (LOGSLMVALFAIL)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	Neighbor node address
6	Local or remote indicator 0 = Local validation failure 1 = Remote validation failure
7	LINK to text of failed Greeting message (bytes)
8	Reason for validation failure 2 = Incompatible Station level 3 = Network-Version mismatch 4 = Network-Maximum-Segment-Size mismatch 5 = Node address validity failure 6 = Shutdown in progress 7 = Password mismatch 8 = Insufficient resources 9 = Invalid greeting
9–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–115, "Major Type 11–Minor Type 54," is written periodically when the MONITOR attribute has been set for a particular BNA Version 1 station. The MONITOR attribute is set through the MODIFY STATION operator command and causes BNA Version 1 to write information to the log file about the amount of traffic handled by the specified station and the number of errors incurred. The frequency with which this entry is written is controlled by the Station Monitor Interval attribute, which is set by the MONITOR operator command. This entry is written only when the LOGGING attribute is set to at least the value MINIMUM.

Table 12–115. Major Type 11–Minor Type 54

Minor Type = 54 Station Monitor (LOGSLMMONRPT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	Neighbor information
	[47:32] Public Data Network ID (X.25) (See Table 12–72, "X–25 PDNs.")
	[15:16] Neighbor node address
6	Station information
	[23:16] Hardware ID (LSN)
	[7: 8] Station Type (See Table 12–71, "Station Types.")
7	Total frames sent
8	Total frames received
9	Info-frames sent (BDLC, ISC)
10	FCS failures (BDLC), or link errors (ISC)
11	Memory errors (BDLC)
12	Short frames (BDLC)
13	Total bytes sent
14	Total bytes received
15	Link information
	[47: 8] Link efficiency
	[39:24] Link speed
	[15:16] Working link maximum segment size

continued

Table 12–115. Major Type 11–Minor Type 54 (cont.)

Minor Type = 54 Station Monitor (LOGSLMMONRPT)	
Words	**Description**
16	Reserved
17	Total data packets received (for X.25 Station)
18	Total data packets sent (for X.25 Station)
19–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–116, "Major Type 11–Minor Type 55," is written when a BNA Version 1 station is added in response to an ADD STATION operator command. It is also generated when a BNA Version 1 station is added automatically. For example, data link control (BDLC) stations are added automatically by BNA Version 1 when certain conditions are met. This entry is written only when the LOGGING attribute is set to at least the value STANDARD.

Table 12–116. Major Type 11–Minor Type 55

Minor Type = 55 Add Station (LOGSLMADDSTA)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to station name (bytes)
5	LINK to attribute list (words)
	Each word in the attribute list has the following structure:
	[47: 8] Station attribute number
	1 = Station-Type
	2 = Hardware-ID
	3 = Autoinit
	4 = Efficiency

continued

Table 12–116. Major Type 11–Minor Type 55 (cont.)

Minor Type = 55 Add Station (LOGSLMADDSTA)	
Words	**Description**
	5 = Maximum-Segment-Size
	6 = Speed
	7 = Monitor
	8 = Hub-number (ISC)
	11 = Window-Size (ISC)
	12 = Retry-Count-Limit (ISC)
	13 = FRSP-Timer (ISC)
	14 = Checkpoint-Timer (ISC)
	15 = Maximum-Frame-Size (ISC)
	17 = First-PVC (X.25 Group)
	18 = Last-PVC (X.25 Group)
	19 = First-VC-IN (X.25 Group)
	20 = Last-VC-IN (X.25 Group)
	21 = First-VC-IO (X.25 Group)
	22 = Last-VC-IO (X.25 Group)
	23 = First-VC-OUT (X.25 Group)
	24 = Last-VC-OUT (X.25 Group)
	25 = Restart-Retry-Count-Limit (X.25 Group)
	26 = Restart-Response-Timer-Limit (X.25 Group)
	27 = Timer-Limit-Value-T31 (X.25 Group)
	28 = Timer-Limit-Value-T32 (X.25 Group)
	29 = Timer-Limit-Value-T33 (X.25 Group)
	30 = LC-Retry-Count-Limit-1 (X.25 Group)
	31 = LC-Retry-Count-Limit-2 (X.25 Group)
	32 = LC-Local-Window-Size (X.25 Group)
	33 = LC-Packet-Size (X.25 Group)

continued

Table 12–116. Major Type 11–Minor Type 55 (cont.)

Minor Type = 55 Add Station (LOGSLMADDSTA)	
Words	**Description**
	34 = Public-Data-Network (X.25 Group) 35 = Modified (X.25 Group) 41 = Modules (X.25 Group) [39:40] Attribute value All attributes are integer valued except the following: Station type (#1): enumerated (See Table 12–71, “Station Types.”) Autoinit (#3): Boolean 0 = FALSE 1 = TRUE Monitor (#7): enumerated 0 = OFF 1 = ON Public Data Network (#34): enumerated (See Table 12–72, “X–25 PDNs.”) Modified (#35): Boolean 0 = FALSE 1 = TRUE
6	Network services (NS) error code (See Table 12–67, “NS Error Codes.”)
7–end	Variable-length data pointed to by the LINKs in this entry

The log entry in Table 12–117, "Major Type 11–Minor Type 56," is written when a BNA Version 1 operator command is entered from any one of a number of agents. It is also written when a response is sent to the operator. This entry is written only when the LOGGING attribute is set to at least the value MINIMUM.

Table 12–117. Major Type 11–Minor Type 56

Minor Type = 56 Operator Message (LOGNWOIM)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Origin of message 0 = Unknown source 1 = ODT 2 = Remote terminal 3 = Command file 4 = Operator response
5	LINK to text of message (bytes)
6–end	Variable-length data pointed to by the LINKs in this entry

Table 12–118. Major Type 11–Minor Type 57

Minor Type = 57 Operator Initiated Assign (LOGSCMOIASSIGN)	
Words	**Description**
This log entry type is not currently used.	

Table 12–119. Major Type 11–Minor Type 58

Minor Type = 58 SCM Return (LOGSCMRETURN)	
Words	**Description**
This log-entry type is not currently used.	

Table 12–120. Major Type 11–Minor Type 59

Minor Type = 59 Target Initiated Assign (LOGTRGASSIGN)	
Words	**Description**
This log-entry type is not currently used.	

Table 12–121. Major Type 11–Minor Type 60

Minor Type = 60 SCM Frame Receive (LOGSCMFRAMERECV)	
Words	**Description**
This log-entry type is not currently used.	

The log entry in Table 12–122, "Major Type 11–Minor Type 61," is for various types of BNA Version 1 bug information that occasionally need to be logged. This information is for use by Unisys A Series development centers.

Table 12–122. Major Type 11–Minor Type 61

Minor Type = 61 BNA Version 1 Debug (LOGBNADEBUG)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to text of message (bytes)
5–end	Variable-length data pointed to by the LINK in this entry. The symbol hex "FF" means end-of-line, while two of them together mean end-of-text.

The log entries in Table 12–123, "Major Type 11–Minor Types 62 through 64," are written whenever an X.25 remote terminal is added, modified, or deleted using the BNA commands NW ADD TERM, NW MOD TERM, or NW DELETE TERM, respectively. The following entries are written only when the BNA command NW LOG attribute is STANDARD or MAXIMUM.

Table 12–123. Major Type 11–Minor Types 62 through 64

Minor Type = 62 Add X.25 Terminal (LOGX25ADDTERM) **Minor Type = 63 Modify X.25 Terminal (LOGX25MODIFYTERM)** **Minor Type = 64 Delete X.25 Terminal (LOGX25DELETETERM)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to the remote terminal name (bytes)
5	LINK to the station name (bytes)
6	LINK to the terminal attribute list. The elements of this list are single words, structured in the following way: [47:08] Attribute number

continued

Table 12–123. **Major Type 11–Minor Types 62 through 64** (cont.)

Minor Type = 62 Add X.25 Terminal (LOGX25ADDTERM) **Minor Type = 63 Modify X.25 Terminal (LOGX25MODIFYTERM)** **Minor Type = 64 Delete X.25 Terminal (LOGX25DELETETERM)**	
Words	**Description**
	[39:40] For some attribute numbers, this field simply contains the attribute value. For all other attribute numbers, this field is a LINK to variable-length information.
7–end	Variable-length data pointed to by the LINKs in this entry.

The log entries in Table 12–124, "Major Type 11–Minor Types 65 through 67," are written whenever an X.25 application is added, modified, or deleted using the BNA commands NW ADD APPL, NW MOD APPL, or NW DELETE APPL, respectively. The following entries are written only when the BNA command NW LOG attribute is STANDARD or MAXIMUM.

Table 12–124. Major Type 11–Minor Types 65 through 67

Minor Type = 65 Add X.25 Application (LOGX25ADDAPPL) **Minor Type = 66 Modify X.25 Application (LOGX25MODIFYAPPL)** **Minor Type = 67 Delete X.25 Application (LOGX25DELETEAPPL)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to the application code file name (bytes)
5	LINK to the application attribute list. The elements of this list are single words, structured in the following way: [47:08] Attribute number [39:40] For some attribute numbers, this field simply contains the attribute value. For all other attribute numbers, this field is a LINK to variable-length information.

continued

Table 12–124. Major Type 11–Minor Types 65 through 67 (cont.)

Minor Type = 65 Add X.25 Application (LOGX25ADDAPPL) **Minor Type = 66 Modify X.25 Application (LOGX25MODIFYAPPL)** **Minor Type = 67 Delete X.25 Application (LOGX25DELETEAPPL)**	
Words	**Description**
6–end	Variable-length data pointed to by the LINKs in this entry.

The log entry in Table 12–125, "Major Type 11–Minor Type 68," is written whenever there is a phase transition in the X.25 MCS. The following entry is written only when the BNA command NW LOG attribute is STANDARD or MAXIMUM.

Table 12–125. Major Type 11–Minor Type 68

Minor Type = 68 X.25 MCS Phase Change (LOGX25PHASECHANGE)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	The new phase of the X.25 MCS. The following are valid X.25 MCS phases: 0 = Not running 1 = Initializing 2 = Operating 3 = Slow shutdown 4 = Fast shutdown

The log entry in Table 12–126, "Major Type 11–Minor Type 69," is written whenever an X.25 virtual call is successfully established. The following entry is written only when the BNA command NW LOG attribute is STANDARD or MAXIMUM.

Table 12–126. Major Type 11–Minor Type 69

Minor Type = 69 X.25 Call Established (LOGX25CALLESTAB)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to the name of the remote terminal for this call.
5	The X.25 station table index associated with this call.
6	LINK to the code file name of the local application that initiated this call.
7	The local endpoint token associated with this call.
8	The transmit packet size for this call.
9	The transmit window size for this call.
10	The transmit throughput for this call.
11	The receive packet size for this call.
12	The receive window size for this call.
13	The receive throughput for this call.
14–end	Variable-length data pointed to by the links in this entry.

The log entry in Table 12–127, "Major Type 11–Minor Type 70," is written whenever an X.25 virtual call terminates. The following entry is written only when the BNA command NW LOG attribute is STANDARD or MAXIMUM.

Table 12–127. Major Type 11–Minor Type 70

Minor Type = 70 X.25 Call Terminated (LOGX25CALLTERM)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to the name of the remote terminal for this call.
5	The X.25 station table index associated with this call.

continued

Table 12–127. Major Type 11–Minor Type 70 (cont.)

Minor Type = 70 X.25 Call Terminated (LOGX25CALLTERM)	
Words	**Description**
6	LINK to the code file name of the local application that initiated this call.
7	The local endpoint token associated with this call.
8	The number of packets sent during this call.
9	The number of packets received during this call.
10	The number of octets sent during this call.
11	The number of octets received during this call.
12	The encoded reason for the termination of this call.
13–end	Variable-length data pointed to by the links in this entry.

The log entry in Table 12–128, "Major Type 11–Minor Type 71," is written whenever an X.25 virtual call terminates, and there are call statistics in the X.25 clear indication/confirmation packet. The following entry is written only when the BNA command NW LOG attribute is STANDARD or MAXIMUM.

Table 12–128. Major Type 11–Minor Type 71

Minor Type = 71 X.25 Call Statistics (LOGX25CALLSTATS)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to the name of the remote terminal for this call.
5	The X.25 station table index associated with this call.
6	LINK to the code file name of the local application that initiated this call.
7	The local endpoint token associated with this call.
8	LINK to monetary unit information.
9	LINK to packet count information.

continued

Table 12–128. Major Type 11–Minor Type 71 (cont.)

Minor Type = 71 X.25 Call Statistics (LOGX25CALLSTATS)	
Words	**Description**
10	LINK to call duration information.
11–end	Variable-length information pointed to by the links in this entry.

The log entry in Table 12–129, "Major Type 11–Minor Type 72," is written whenever an X.25 local endpoint is moved due to the execution of the MOVECOMNO function. The purpose of this function is to hand off the local endpoint of a virtual call to another application program. The following entry is written only when the BNA command NW LOG attribute is STANDARD or MAXIMUM.

Table 12–129. Major Type 11–Minor Type 72

Minor Type = 72 X.25 Endpoint Moved (LOGX25ENDPTMOVED)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Old endpoint token.
5	New endpoint token.
6	LINK to the code file name of the old application.
7	LINK to the code file name of the new application.
11–end	Variable-length information pointed to by the links in this entry.

The log entry in Table 12–130, "Major Type 11–Minor Type 73," is written whenever the X.25 MCS outputs an operator message. The log entry simply consists of the operator message text. The following entry is written only when the BNA command NW LOG attribute is MINIMUM, STANDARD, or MAXIMUM.

Table 12–130. Major Type 11–Minor Type 73

Minor Type = 73 X.25 REPORT (LOGX25REPORT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to the operator message text.
5–end	Variable-length data pointed to by the LINK in this entry.

Pages 12–269 through 12–276 are deleted by *–110*.

Table 12–135. Major Type 12–Minor Type 5

Minor Type = 5 Unit Ready Entry (LOG_VSID_UNIT_READY)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	DLP ID
5	VSID unit number

Table 12–136. Major Type 12–Minor Type 6

Minor Type = 6 Unit Not Ready Entry (LOG_VSID_UNIT_NOT_READY)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	DLP ID
5	VSID unit number
6	Problem code: a device-dependent integer that identifies the problem. (See the documentation for the device for an explanation of the values.)
7	LINK to the text that explains the problem
8–end	Variable-length data pointed to by LINK in this entry

Table 12–137. Major Type 12–Minor Type 7

Minor Type = 7 Unit In Use Entry (LOG_BEGIN_VSID_USE)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	DLP ID
5	VSID unit number
6	Identification of the page-writer job

continued

Table 12–137. Major Type 12–Minor Type 7 (cont.)

Minor Type = 7 Unit In Use Entry (LOG_BEGIN_VSID_USE)	
Words	**Description**
	[31:16] Page-writer job number
	[15:16] Page-writer task number
7	LINK to page-writer code file title (bytes)
8	LINK to usercode for page-writer job (bytes)
9	LINK to accesscode for page-writer job (bytes)
10	LINK to chargecode for page-writer job (bytes)
11–end	Variable-length data pointed to by the LINKS in this entry.

Table 12–138. Major Type 12–Minor Type 8

Minor Type = 8 Unit Not In Use Entry (LOG_END_VSID_USE)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	DLP ID
5	VSID unit number
6	Identification of the page-writer job
	[31:16] Page-writer job number
	[15:16] Page-writer task number
7	LINK to page-writer code file title (bytes)
8	LINK to usercode for page-writer job (bytes)
9	LINK to accesscode for page-writer job (bytes)
10	LINK to chargecode for page-writer job (bytes)

continued

Table 12–138. Major Type 12–Minor Type 8 (cont.)

Minor Type = 8 Unit Not In Use Entry (LOG_END_VSID_USE)	
Words	**Description**
11	The value of the HISTORY task attribute of the user library stack upon termination. (This is the private library stack that exports the entry points used by the page writer job.)
12–end	Variable-length data pointed to by the LINKS in this entry

Table 12–139. Major Type 12–Minor Type 9

Minor Type = 9 Begin Printing a Copy Entry (LOG_BEGIN_VSID_PRINTING)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	DLP ID
5	VSID unit number
6	Identification of the page-writer job
	[31:16] Page-writer job number
	[15:16] Page-writer task number
7	LINK to page-writer code file title (bytes)
8	LINK to usercode for page-writer job (bytes)
9	LINK to accesscode for page-writer job (bytes)
10	LINK to chargecode for page-writer job (bytes)
11	Identification of the page-creator job
	[31:16] Page-creator job number
	[15:16] Page-creator task number
12	LINK to page-creator code file title (bytes)

continued

Table 12–139. Major Type 12–Minor Type 9 (cont.)

Minor Type = 9 Begin Printing a Copy Entry (LOG_BEGIN_VSID_PRINTING)	
Words	**Description**
13	LINK to usercode for page-creator job (bytes)
14	LINK to accesscode for page-creator job (bytes)
15	LINK to chargecode for page-creator job (bytes)
16	LINK to VSID file title (bytes)
17	LINK to file title of backup file, if it exists; otherwise zero (bytes)
18	Copy count information
	[47:24] Copy number of this copy (starting at 1)
	[23:24] Number of copies to be printed
19–end	Variable-length data pointed to by the LINKS in this entry

Table 12–140. Major Type 12–Minor Type 10

Minor Type = 10 End of Printing a Copy Entry (LOG_END_VSID_PRINTING)	
Words	**Description**
0–3	As described in Table 12–2, “First Four Log Entry Words”
4	DLP ID
5	VSID unit number
6	Identification of the page-writer job
	[31:16] Page-writer job number
	[15:16] Page-writer task number
7	LINK to page-writer code file title (bytes)
8	LINK to usercode for page-writer job (bytes)

continued

Table 12–140. Major Type 12–Minor Type 10 (cont.)

Minor Type = 10 End of Printing a Copy Entry (LOG_END_VSID_PRINTING)	
Words	**Description**
9	LINK to accesscode for page-writer job (bytes)
10	LINK to chargecode for page-writer job (bytes)
11	Identification of the page-creator job [31:16] Page-creator job number [15:16] Page-creator task number
12	LINK to page-creator code file title (bytes)
13	LINK to usercode for page-creator job (bytes)
14	LINK to accesscode for page-creator job (bytes)
15	LINK to chargecode for page-creator job (bytes)
16	LINK to VSID file title (bytes)
17	LINK to file title of backup file, if it exists; otherwise zero (bytes)
18	Copy count information [47:24] Copy number of this copy (starting at 1) [23:24] Number of copies to be printed
19	Page count information [47:24] Number of "logical pages" printed [23:24] Number of physical pages (sheets of paper) printed
20–end	Variable-length data pointed to by the LINKS in this entry

Table 12–141. Major Type 12–Minor Type 11

Minor Type = 11 Special Log Information Entry (LOG_VSID_EVENT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	DLP ID
5	VSID unit number
6	Event code: a device-dependent integer that identifies the event to be logged (see the documentation for the device for an explanation of the values)
7	LINK to the text that defines, describes, or clarifies the event (bytes)
8–end	Variable-length data pointed to by the LINK in this entry

Table 12–142. Major Type 12–Minor Type 12

Minor Type = 12 Peripheral Interface Error Entry (LOG_VSID_ERROR_LOG)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	DLP ID
5	VSID unit number
6	LINK to the VSID error log (bytes)
7–end	The VSID error log pointed to by the LINK in this entry. The format and content of the VSID error log can be found in the maintenance documentation (PTD/DOC/MAINT/NIP) included with the PTD package for the DLP. The error log begins with Word 3 of the Retry Log described in that document.

Major Type = 13 BNA Version 2 Entry (LOGMAJBNAE)

The *BNA Version 2 Network Encoded Messages Programming Reference Manual, Volume 2* explains the format of the log records after Word 3.

Table 12–143. Major Type 13–Minor Types 0 and 1

Minor Type = 0 General BNA Version 2 Log Entry Minor Type = 1 Local Task Support Activity Log Entry	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4–end	Variable-length data

Major Type = 14 MLS Message Entry (LOGMAJMLS)

Major Type 14 logs MCP messages. The raw log contains the message number and the parameters used to construct the message that was displayed.

If you are writing a log analysis program, the message text can be displayed by having the program call the MCPMESSAGESEARCHER procedure. You can indicate which message is to be translated by passing the message number (Word 6) as a parameter, and having the pointer to the message parameter list directed to Word 7.

Table 12–144. Major Type 14–Minor Types 1, 4, 9 and 10

Minor Type = 1 RSVP Message Entry (LOGMLSRSVP)
Minor Type = 4 INFO Message Entry (LOGMLSINFO)
Minor Type = 9 Unit RSVP Message Entry (LOGMLSUNITRSVP)
Minor Type = 10 Special RSVP Message Entry (LOGMLSSPECIALRSVP)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Length of message parameter list in words, plus 2
5	0 (zero) or mix number
6	Message number
7–end, minus three	Message parameter list The message parameter list consists of zero or more contiguous output message parameters. Output message parameters are described in the *ALGOL Programming Reference Manual, Volume 1: Basic Implementation*. Each output message parameter is formatted as follows: • Byte 1 = parameter number (1 through 255, or 0. Zero marks the end of the message parameter.) • Byte 2 and 3 = length (in bytes) of the text that follows. The length can be from 0 to 65535 bytes. • Byte 4 through end = text of message parameter
Last three words	The identity (if any) of the object code file that displayed the message.

Major Type = 15 Volume Status Entry (LOGMAJVOL)

The log entries in Table 12–145, "Major Type 15–Minor Types 1 through 5," describe the state of disk and tape volumes on the system. When a disk or tape volume comes online or is taken offline, a volume online or offline entry is issued. The following events can cause volume online entries to be issued: halt/load, ACQUIRE, or RC. The following events can cause volume offline entries to be issued for disk: CLOSE, PO, FREE, RC, LB, PG; and for tape: LOCK, RW, FREE, taking tape offline.

Note that the SV (Save) or UR (Unit Reserved) system commands do not generate volume offline entries. These commands are used to suspend file open actions, but they do not cause the system to close out the internal information concerning the volume. Also, a subsequent RY (Ready) or *UR-* system command in such a case does not cause a reestablishment of the volume internally, so that no volume online entry is issued.

When a tape volume is purged or is assigned a new serial number, a tape volume purged entry is issued. When a tape volume comes online with a write ring, it appears as a scratch tape if the first file on the tape has expired. In this case, a tape volume expired entry is issued. When a scratch tape is opened for output or is copied to, a tape volume new file entry is issued.

Table 12–145. Major Type 15–Minor Types 1 through 5

Minor Type = 1 Volume Online (LOGVOLON) **Minor Type = 2 Volume Offline (LOGVOLOFF)** **Minor Type = 3 Tape Volume Purged (LOGVOLPG)** **Minor Type = 4 Tape Volume Expired (LOGVOLEXP)** **Minor Type = 5 Tape Volume NEWFILE (LOGVOLNEW)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	External device number
5	Peripheral Description
	[47: 8] Peripheral Type (See Table 12–22, "Peripheral Logical Types.")
	[39: 8] Peripheral subtype (See Table 12–23, "Peripheral Subtypes.")
	[31: 9] Density (See Table 12–24, "Peripheral Densities.")
6	Serial Number in EBCDIC
7	Volume Status
	[47: 1] SET if volume is scratch

continued

Table 12–145. Major Type 15–Minor Types 1 through 5 (cont.)

Minor Type = 1 Volume Online (LOGVOLON)
Minor Type = 2 Volume Offline (LOGVOLOFF)
Minor Type = 3 Tape Volume Purged (LOGVOLPG)
Minor Type = 4 Tape Volume Expired (LOGVOLEXP)
Minor Type = 5 Tape Volume NEWFILE (LOGVOLNEW)

Words	Description
	[46: 1] SET if error was detected in label (parts of log entry might be missing or invalid)
	[45: 1] SET if volume is listed in the volume library as VOLUMEd
	[44: 1] Has a write ring
	[43: 1] SET if volume is listed in the volume directory
The remainder of this log entry is not filled in if the volume is scratch (except for Words 17 and 18, which store the new serial number and new density for tape volume purged entries).	
8–10	Volume or family name (first byte is binary number of characters that follow).
The remaining words of the log entry differ depending on whether the volume is a disk or tape volume.	
For disk volumes:	
11	Status fields:
	[47: 1] SET if disk has directory (for example, base pack)
	[46: 1] SET if disk is interchange mode
	[45: 1] SET if disk is IAD
	[44: 1] SET if disk is mirrored
The remainder of the disk log entry is not filled in for interchange mode packs.	
12	Family index number
13	Family serial number in EBCDIC (serial number of original base pack for family)
14	Family creation date (Julian = YYDDD)
15	Family creation time (where time is the number of 2.4 microsecond clock ticks since midnight)
16	System serial number of site where family was originally created

continued

Table 12–145. Major Type 15–Minor Types 1 through 5 (cont.)

Minor Type = 1 Volume Online (LOGVOLON)
Minor Type = 2 Volume Offline (LOGVOLOFF)
Minor Type = 3 Tape Volume Purged (LOGVOLPG)
Minor Type = 4 Tape Volume Expired (LOGVOLEXP)
Minor Type = 5 Tape Volume NEWFILE (LOGVOLNEW)

Words	Description
For tape volumes:	
11	[23: 1] LOCKEDFILE attribute
	[20: 1] Parity
	[16: 1] SET if LABELTYPE field is valid
	[15: 4] LABELTYPE attribute
12	[47: 9] Label level
	[37:14] Cycle
	[23: 8] Version
	[14:15] FILESECTION number
13	[45:14] System serial number of system that created file
	[31: 4] GENERATION attribute
	[27:11] SAVEFACTOR attribute
	[16:17] CREATIONDATE (Julian = YYDDD)
14–16	Tape usercode or zeros (first byte is binary count of characters in the usercode)
17	New serial number of tape volume (this can differ from original serial number above for SN MT entry)
18	New density
	[31: 9] New density of tape volume. This can differ from original density above for SN MT entry. (See Table 12–24, "Peripheral Densities.")
19–21	File name identifier (for the first file on the tape volume). The first byte of the name is a binary number that counts characters in the name that follows.

continued

Table 12–145. Major Type 15–Minor Types 1 through 5 (cont.)

Minor Type = 1 Volume Online (LOGVOLON) **Minor Type = 2 Volume Offline (LOGVOLOFF)** **Minor Type = 3 Tape Volume Purged (LOGVOLPG)** **Minor Type = 4 Tape Volume Expired (LOGVOLEXP)** **Minor Type = 5 Tape Volume NEWFILE (LOGVOLNEW)**	
Words	**Description**
22–end	Variable-length data pointed to by the LINKs in this entry

The volume directory entries in Table 12–146, "Major Type 15–Minor Types 6 through 11," are issued when the TAPECHECK option of the SECOPT system command has been set to AUTOMATIC. This function is only available on an InfoGuard system. Volume directory add, change, delete, and destroy entries are issued when the corresponding VOLUME statements in WFL (or the equivalent SETSTATUS requests) are executed. Volume directory overwrite entries are issued when a volume directory entry is updated because a volume label has been overwritten. Volume directory purge entries are issued when a volume directory entry is updated because a volume label has been purged.

Because part of the contents of this log entry is a volume directory data entry, the LINKs used in this entry differ from the general format, and an offset of 3 has to be added to any index value to get the actual index value in the log entry. The LINK format is as follows:

Field	Meaning
[23:12]	Length of the item in words
[11:12]	Index to the start of the item, minus 3

The format of all these log entries is shown in Table 12–146.

Table 12–146. Major Type 15–Minor Types 6 through 11

Minor Type = 6 Volume Directory Add (LOGVSADD)
Minor Type = 7 Volume Directory Change (LOGVSCHANGE)
Minor Type = 8 Volume Directory Delete (LOGVSDELETE)
Minor Type = 9 Volume Directory Destroy (LOGVSDESTROY)
Minor Type = 10 Volume Directory Overwrite (LOGVSOVER)
Minor Type = 11 Volume Directory Purge (LOGVSPURGE)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Type of record
	[47: 8] EBCDIC "D" (data block)
	[39: 8] Volume family LABELKIND
	[31: 8] Internal MCP structure level
	[15: 4] Volume family medium type
	1 = Scratch tape
	2 = Other tape
	3 = Disk or pack volume
	[10: 1] SET if matching by serial number only
	[9:10] Index to start of variable-length items (offset –3 words)
5	Date and time of last volume directory update
6	Volume family creation date (Julian = YYDDD)
7	Creation site system serial number
8	SAVEFACTOR attribute value of first file on volume
9–11	Volume family owner usercode (substandardform)
12–14	Volume family FAMILYNAME (substandardform)
15	Security attributes
	[47: 2] Security type
	0 = Public
	1 = Guarded
	2 = Controlled

continued

Table 12–146. Major Type 15–Minor Types 6 through 11 (cont.)

Minor Type = 6 Volume Directory Add (LOGVSADD)
Minor Type = 7 Volume Directory Change (LOGVSCHANGE)
Minor Type = 8 Volume Directory Delete (LOGVSDELETE)
Minor Type = 9 Volume Directory Destroy (LOGVSDESTROY)
Minor Type = 10 Volume Directory Overwrite (LOGVSOVER)
Minor Type = 11 Volume Directory Purge (LOGVSPURGE)

Words	**Description**
	3 = Private
	[45: 2] Security use
	0 = IO
	1 = In
	2 = Out
	3 = Secured
16–18	Reserved for future use
19	Link to the serial numbers of each volume in the family. Each serial number is stored in one word in EBCDIC.
20	Link to volume status word per family member. Volume status word format is
	[47: 1] SET if the volume is permanently owned
	[46: 1] SET if the volume is not RESTRICTED
	[45: 1] SET if the volume is destroyed
	[5: 6] Volume KIND attribute value
21	Link to the volume family GUARDFILE title
22–end	Variable-length data pointed to by the links in this entry

Major Type = 16 File Status Entry (LOGMAJFILE)

The file status entries in Table 12–147, "Major Type 16–Minor Types 1 through 6," describe the creation, removal, title change, and security attribute changes of permanent disk files, and the copying of disk and tape files. Note that once a disk file is entered into the directory, the file status (FILEKIND, CYCLE, VERSION and timestamp) can be changed by programs. These changes are not subsequently logged.

When a new file is entered into the directory or an old file name in the directory is changed, a *duplicate file name* condition can occur. In order to update the directory, the system removes the duplicate file. The log entry for a file creation or file name change also describes the duplicate file removal in these cases.

When a log entry of Major Type 16, Minor Type 1 (File Creation & Copy) is logged, and the file involved is a guard file, the system creates a backup copy of the guard file. Refer to "Requesting Copies of Configuration Files" earlier in this section.

Table 12–147. Major Type 16–Minor Types 1 through 6

Minor Type = 1 File Creation & Copy (LGFLCOPYNEWV)
Minor Type = 2 File Creation (LGFLNEWV)
Minor Type = 3 File Removal (LGFLREMOVEV)
Minor Type = 4 File Title Change (LGFLCHANGEV)
Minor Type = 5 File Security Attribute Change (LGFLSECATTV)
Minor Type = 6 File Copy (LGFLCOPYV)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Miscellaneous [47: 4] Error indicator (0 = no error) [31: 8] Peripheral type (See Table 12–22, "Peripheral Logical Types.")
5	External device number of the disk drive containing the base pack for the disk family
6	The serial number (in EBCDIC) of the original base pack for the disk family; can also be the serial number of the tape for file copy entries
7–9	The disk family name, first byte contains the length; can also be the tape name for file copy entries
10	LINK to old file name, zero for file create entries
11	LINK to new file name, zero for file removal entries or file copy entries

continued

Table 12–147. Major Type 16–Minor Types 1 through 6 (cont.)

Minor Type = 1 File Creation & Copy (LGFLCOPYNEWV)
Minor Type = 2 File Creation (LGFLNEWV)
Minor Type = 3 File Removal (LGFLREMOVEV)
Minor Type = 4 File Title Change (LGFLCHANGEV)
Minor Type = 5 File Security Attribute Change (LGFLSECATTV)
Minor Type = 6 File Copy (LGFLCOPYV)

Words	Description
12	File status of old file, zero for file create entries
	[47: 1] Set if file is cataloged
	[45:14] File CYCLE attribute
	[31: 8] File VERSION attribute
	[11:12] File FILEKIND attribute
13	File status of new file, zero for file removal entries or file copy entries
	Same format as Word 12
14	File status of removed duplicate file, nonzero if creation or title change caused duplicate file to be removed, zero for file copy entries
	Same format as Word 12
15	Timestamp of old file, zero for file create entries
16	Timestamp of new file, zero for file removal entries or file copy entries
17	Timestamp of removed duplicate file, nonzero if creation or title change caused duplicate file to be removed, zero for file copy entries
18	SECURITYTYPE attribute value of new file, zero for file copy entries
19	SECURITYUSE attribute value of new file, zero for file copy entries
20	LINK to SECURITYGUARD attribute value of new file, zero for file copy entries
21	Variable-length data pointed to by the LINKs in this entry

Major Type = 17 DATA COMM Configuration Entry (LOGMAJ_DCCONFIGV)

Table 12–148. Major Type 17–Minor Type 1

Minor Type = 1 Data comm change entry (LOGMIN_DCCHANGEV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Action code 1 = Create 2 = Modify 3 = Move 4 = Add 5 = Subtract 6 = Copy
5	Structure code 1 = Line 2 = Station 3 = MCS 4 = Algorithm 5 = Editor
6	LINK to structure name
7	LINK to miscellaneous information
8–end	Variable-length data pointed to by LINKs in this entry

When a log entry of Major Type 17, Minor Type 2 (Data Comm Installation & Copy) is logged, the system creates a backup copy of the DATACOMINFO file. Refer to "Requesting Copies of Configuration Files" earlier in this section.

Table 12–149. Major Type 17–Minor Types 2 and 3

Minor Type = 2 Data comm Installation & Copy Entry (LOGMIN_DCINSTCOPYV)
Minor Type = 3 Data comm Installation Entry (LOGMIN_DCINSTV)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to new data comm configuration file name
5	LINK to copied file title
6–end	Variable-length data pointed to by LINKs in this entry

Major Type = 18 COMS Configuration Entry (LOGMAJ_COMSCONFIGV)

Table 12–150. Major Type 18–Minor Types 1 through 11

Minor Type = 1 Agenda Change (LOGMIN_AGENDAV)
Minor Type = 2 Program Change (LOGMIN_PROGRAMV)
Minor Type = 3 Processing-Item Change (LOGMIN_PROCITEMV)
Minor Type = 4 Processing-Item List Change (LOGMIN_PROCITEMLV)
Minor Type = 5 Library Change (LOGMIN_COMSLIBV)
Minor Type = 6 Database Change (LOGMIN_COMSDBSV)
Minor Type = 7 Station Change (LOGMIN_STATIONV)
Minor Type = 8 Station List Change (LOGMIN_STATIONLV)
Minor Type = 9 Window Change (LOGMIN_WINDOWV)
Minor Type = 10 Window List Change (LOGMIN_WINDOWLV)
Minor Type = 11 Usercode Change (LOGMIN_COMSUSERV)

Words	Description	
0–3	As described in Table 12–2, "First Four Log Entry Words"	
4	Update information	
	[15: 8] Type	
	1 = Create	
	2 = Modify	
	3 = Delete	
	[7: 8] Number of items affected	
5–end	Byte-oriented stream of item descriptions in the following format per item:	
	Byte	**Meaning**
	0–1	Item length in bytes
	2	Item field number
		Minor Type = 1 Agenda
		1 = Agenda name
		2 = Window name
		3 = Default input agenda
		4 = Default output agenda
		5 = Processing-item list

continued

Table 12–150. Major Type 18–Minor Types 1 through 11 (cont.)

Minor Type = 1 Agenda Change (LOGMIN_AGENDAV) **Minor Type = 2 Program Change (LOGMIN_PROGRAMV)** **Minor Type = 3 Processing-Item Change (LOGMIN_PROCITEMV)** **Minor Type = 4 Processing-Item List Change (LOGMIN_PROCITEMLV)** **Minor Type = 5 Library Change (LOGMIN_COMSLIBV)** **Minor Type = 6 Database Change (LOGMIN_COMSDBSV)** **Minor Type = 7 Station Change (LOGMIN_STATIONV)** **Minor Type = 8 Station List Change (LOGMIN_STATIONLV)** **Minor Type = 9 Window Change (LOGMIN_WINDOWV)** **Minor Type = 10 Window List Change (LOGMIN_WINDOWLV)** **Minor Type = 11 Usercode Change (LOGMIN_COMSUSERV)**	
Words	**Description**
	6 = Destination program
	Minor Type = 2 Program
	1 = Program name
	2 = Code file title
	3 = Usercode
	4 = Family
	5 = Database name
	Minor Type = 3 Processing-Item
	1 = Processing-item name
	2 = Actual name of library object
	3 = Library name
	Minor Type = 4 Processing-Item List
	1 = List name
	2 = Processing-item name
	Minor Type = 5 Library
	1 = Library name
	2 = Code file title
	Minor Type = 6 Database
	1 = Database name
	2 = Usercode name
	3 = Code file title

continued

Table 12–150. Major Type 18–Minor Types 1 through 11 (cont.)

Minor Type = 1 Agenda Change (LOGMIN_AGENDAV) **Minor Type = 2 Program Change (LOGMIN_PROGRAMV)** **Minor Type = 3 Processing-Item Change (LOGMIN_PROCITEMV)** **Minor Type = 4 Processing-Item List Change (LOGMIN_PROCITEMLV)** **Minor Type = 5 Library Change (LOGMIN_COMSLIBV)** **Minor Type = 6 Database Change (LOGMIN_COMSDBSV)** **Minor Type = 7 Station Change (LOGMIN_STATIONV)** **Minor Type = 8 Station List Change (LOGMIN_STATIONLV)** **Minor Type = 9 Window Change (LOGMIN_WINDOWV)** **Minor Type = 10 Window List Change (LOGMIN_WINDOWLV)** **Minor Type = 11 Usercode Change (LOGMIN_COMSUSERV)**	
Words	**Description**
	Minor Type = 7 Station 1 = Station name 2 = Hostname 3 = Default window name 4 = Default usercode name 5 = Control station 6 = Super user 7 = System user 8 = Privileged user 9 = Continuous log on Minor Type = 8 Station List 1 = Station list name 2 = Station name Minor Type = 9 Window 1 = Window name 2 = Virtual station name 3 = Remote file window 4 = Remote file program name 5 = MCS window 6 = MCS code file title 7 = Notify open activity

continued

Table 12–150. Major Type 18–Minor Types 1 through 11 (cont.)

Minor Type = 1 Agenda Change (LOGMIN_AGENDAV) Minor Type = 2 Program Change (LOGMIN_PROGRAMV) Minor Type = 3 Processing-Item Change (LOGMIN_PROCITEMV) Minor Type = 4 Processing-Item List Change (LOGMIN_PROCITEMLV) Minor Type = 5 Library Change (LOGMIN_COMSLIBV) Minor Type = 6 Database Change (LOGMIN_COMSDBSV) Minor Type = 7 Station Change (LOGMIN_STATIONV) Minor Type = 8 Station List Change (LOGMIN_STATIONLV) Minor Type = 9 Window Change (LOGMIN_WINDOWV) Minor Type = 10 Window List Change (LOGMIN_WINDOWLV) Minor Type = 11 Usercode Change (LOGMIN_COMSUSERV)		
Words	**Description**	
		8 = Notification text
		9 = Notify on activity
		10 = Notification text
		Minor Type = 10 Window List
		1 = Window list name
		2 = Window name
		Minor Type = 11 Usercode
		1 = Usercode name
		2 = Default window name
		3 = Valid window list name
		4 = Valid station list name
		5 = Control capability
	Byte 3	Item format
		1 = Boolean
		2 = Numeric
		3 = String
	Byte 4	List operation type (Only applicable for list items)
		1 = Add
		2 = Subtract
	Byte 5	Unused

continued

Table 12–150. Major Type 18–Minor Types 1 through 11 (cont.)

Minor Type = 1 Agenda Change (LOGMIN_AGENDAV) Minor Type = 2 Program Change (LOGMIN_PROGRAMV) Minor Type = 3 Processing-Item Change (LOGMIN_PROCITEMV) Minor Type = 4 Processing-Item List Change (LOGMIN_PROCITEMLV) Minor Type = 5 Library Change (LOGMIN_COMSLIBV) Minor Type = 6 Database Change (LOGMIN_COMSDBSV) Minor Type = 7 Station Change (LOGMIN_STATIONV) Minor Type = 8 Station List Change (LOGMIN_STATIONLV) Minor Type = 9 Window Change (LOGMIN_WINDOWV) Minor Type = 10 Window List Change (LOGMIN_WINDOWLV) Minor Type = 11 Usercode Change (LOGMIN_COMSUSERV)		
Words	**Description**	
	Byte 6–end	Item value, variable length as specified in bytes 1–2

Whenever a log entry of Major Type 18, Minor Type 12 (Load File Copy) is logged, the system creates a backup copy of the COMS load file. Refer to "Requesting Copies of Configuration Files" earlier in this section.

Table 12–151. Major Type 18–Minor Types 12 and 13

Minor Type = 12 Load File Copy (LOGMIN_COMSLOADCOPYV) Minor Type = 13 Load File (LOGMIN_COMSLOADV)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to load file title
5	LINK to load file copy title
6–end	Variable-length data pointed to by LINKs in this entry

Major Type = 19 Diagnostics Entry (LOGMAJ_DIAGNOSTICS)

The log entries in Table 12–152, "Major Type 19–Minor Types 1, 4, 5, 6, 7, 11, 12, 13, and 16" store diagnostics information for several types of data comm networks. In this table, the term Ethernet® refers to a type of local area network (LAN).

Table 12–152. Major Type 19–Minor Types 1, 4, 5, 6, 7, 11, 12, 13, and 16

Minor Type = 1 Distributed Systems Service (LOGDIAG_DSS)
Minor Type = 4 Message Handling System (LOGDIAG_MHS)
Minor Type = 5 Open Systems Interconnection (LOGDIAG_OSI)
Minor Type = 6 TCP/IP (LOGDIAG_TCPIP)
Minor Type = 7 Network-Independent Software (LOGDIAG_NIS)
Minor Type = 11 Host LAN Connection (LOGDIAG_HLCN)
Minor Type = 12 OSI Directory (LOGDIAG_DIR)
Minor Type = 13 Network Management System (LOGDIAG_NMS)
Minor Type = 16 Systems Network Architecture (LOGDIAG_SNA)

Words	Description			
0–3	As described in Table 12–2, "First Four Log Entry Words"			
4	[47:13]	Length of entry in bytes or lines, depending on the data format. See the description for Word 5–end.		
	[34:19]	Reserved for future use		
	[15: 4]	Data format		
		0 = Raw binary data		
		1 = Formatted text		
		2 through 15 = Reserved for future formats		
	[11:12]	Subtype indicator. Each subtype contains log entries from particular network diagnostic options. The meaning of the subtype indicator depends on the minor type the log entry is for. The following are the possible subtype indicator values for each minor type, and the network diagnostic options that are logged under each subtype:		
		Minor Type 1 – Distributed Systems Service (DSS)		
		Subtype	**JOBFORMATTER Define**	**Meaning**
		6	LOGDIAG_DSS_FTAM	File Transfer, Access, and Management (FTAM)

continued

Ethernet is a registered trademark of Xerox Corp.

Table 12–152. Major Type 19–Minor Types 1, 4, 5, 6, 7, 11, 12, 13, and 16 (cont.)

Minor Type = 1 Distributed Systems Service (LOGDIAG_DSS)
Minor Type = 4 Message Handling System (LOGDIAG_MHS)
Minor Type = 5 Open Systems Interconnection (LOGDIAG_OSI)
Minor Type = 6 TCP/IP (LOGDIAG_TCPIP)
Minor Type = 7 Network-Independent Software (LOGDIAG_NIS)
Minor Type = 11 Host LAN Connection (LOGDIAG_HLCN)
Minor Type = 12 OSI Directory (LOGDIAG_DIR)
Minor Type = 13 Network Management System (LOGDIAG_NMS)
Minor Type = 16 Systems Network Architecture (LOGDIAG_SNA)

Words	Description		
	7	LOGDIAG_DSS_FTP	File Transfer Protocol (FTP)
	8	LOGDIAG_DSS_SMTP	Simple Mail Transfer Protocol (SMTP)
	10	LOGDIAG_DSS_HLCNTERM	Host LAN Connection (HLCN) Terminal Service
	11	LOGDIAG_DSS_RES	Resolver (RES)
	Minor Type 4 – Message Handling System (MHS)		
	Subtype	**JOBFORMATTER Define**	**Meaning**
	0	LOGDIAG_MHS_OTHER	Other MHS entries
	1	LOGDIAG_MHS_RTS	MHS reliable transfer service
	2	LOGDIAG_MHS_MTA	MHS message transfer agent
	3	LOGDIAG_MHS_AS	MHS addressing services
	4	LOGDIAG_MHS_UI	MHS user interface
	Minor Type 5 – Open Systems Interconnection (OSI)		
	Subtype	**JOBFORMATTER Define**	**Meaning**
	0	LOGDIAG_OSI_OTHER	Other OSI entries
	2	LOGDIAG_OSI_PRESENTATION	OSI presentation layer

continued

Table 12–152. **Major Type 19–Minor Types 1, 4, 5, 6, 7, 11, 12, 13, and 16** (cont.)

Minor Type = 1 Distributed Systems Service (LOGDIAG_DSS)
Minor Type = 4 Message Handling System (LOGDIAG_MHS)
Minor Type = 5 Open Systems Interconnection (LOGDIAG_OSI)
Minor Type = 6 TCP/IP (LOGDIAG_TCPIP)
Minor Type = 7 Network-Independent Software (LOGDIAG_NIS)
Minor Type = 11 Host LAN Connection (LOGDIAG_HLCN)
Minor Type = 12 OSI Directory (LOGDIAG_DIR)
Minor Type = 13 Network Management System (LOGDIAG_NMS)
Minor Type = 16 Systems Network Architecture (LOGDIAG_SNA)

Words	**Description**		
	3	LOGDIAG_OSI_SESSION	OSI session layer
	4	LOGDIAG_OSI_TRANSPORT	OSI transport layer
	5	LOGDIAG_OSI_NSI	OSI network services interface
	6	LOGDIAG_OSI_ACSE	OSI application control service element
	7	LOGDIAG_OSI_ENVIRONMENT	OSI environment manager
	19	LOGDIAG_OSI_CM	OSI connection manager
	20	LOGDIAG_OSI_PIE	OSI process intercommunication element
	28	LOGDIAG_OSI_EAM	OSI endpoint address manager
	Minor Type 6 – TCP/IP (TCPIP)		
	Subtype	**JOBFORMATTER Define**	**Meaning**
	0	LOGDIAG_TCPIP_OTHER	Miscellaneous
	1	LOGDIAG_TCPIP_PORTDEBUG	MCP – PORTDEBUG
	2	LOGDIAG_TCPIP_IP	Internet Protocol layer
	3	LOGDIAG_TCPIP_ARP	Address Resolution Protocol function

continued

Table 12–152. **Major Type 19–Minor Types 1, 4, 5, 6, 7, 11, 12, 13, and 16** (cont.)

Minor Type = 1 Distributed Systems Service (LOGDIAG_DSS)
Minor Type = 4 Message Handling System (LOGDIAG_MHS)
Minor Type = 5 Open Systems Interconnection (LOGDIAG_OSI)
Minor Type = 6 TCP/IP (LOGDIAG_TCPIP)
Minor Type = 7 Network-Independent Software (LOGDIAG_NIS)
Minor Type = 11 Host LAN Connection (LOGDIAG_HLCN)
Minor Type = 12 OSI Directory (LOGDIAG_DIR)
Minor Type = 13 Network Management System (LOGDIAG_NMS)
Minor Type = 16 Systems Network Architecture (LOGDIAG_SNA)

Words	**Description**		
	4	LOGDIAG_TCPIP_RIP	Route Information Protocol function
	5	LOGDIAG_TCPIP_UDP	User Datagram Protocol layer
	6	LOGDIAG_TCPIP_ICMP	Internet Control Message Protocol function of IP
	7	LOGDIAG_TCPIP_TCPMGR	TCP Manager function
	8	LOGDIAG_TCPIP_PIM	Connection open and close reports
	9		Reserved
	10	LOGDIAG_TCPIP_SNMP	Entries generated during the process of an SNMP command
	11	LOGDIAG_TCPIP_TCP	Transmission Control Protocol layer
	Minor Type 7 – Network-Independent Software (NIS)		
	Subtype	**JOBFORMATTER Define**	**Meaning**
	0	LOGDIAG_NIS_OTHER	Other NIS entries
	11	LOGDIAG_NIS_MSG	NIS message module
	12	LOGDIAG_NIS_INPUT	NIS input module
	13	LOGDIAG_NIS_OUTPUT	NIS output module
	14	LOGDIAG_NIS_ERROR	NIS error module

continued

Table 12–152. Major Type 19–Minor Types 1, 4, 5, 6, 7, 11, 12, 13, and 16 (cont.)

Minor Type = 1 Distributed Systems Service (LOGDIAG_DSS)
Minor Type = 4 Message Handling System (LOGDIAG_MHS)
Minor Type = 5 Open Systems Interconnection (LOGDIAG_OSI)
Minor Type = 6 TCP/IP (LOGDIAG_TCPIP)
Minor Type = 7 Network-Independent Software (LOGDIAG_NIS)
Minor Type = 11 Host LAN Connection (LOGDIAG_HLCN)
Minor Type = 12 OSI Directory (LOGDIAG_DIR)
Minor Type = 13 Network Management System (LOGDIAG_NMS)
Minor Type = 16 Systems Network Architecture (LOGDIAG_SNA)

Words	Description		
	15	LOGDIAG_NIS_DEBUG	NIS debug module
	16	LOGDIAG_NIS_LOCK	NIS lock module
	17	LOGDIAG_NIS_QUEUE	NIS queue module
	18	LOGDIAG_NIS_KIO	NIS KEYEDIO module
	19	LOGDIAG_NIS_NS	NIS network servant module
	21	LOGDIAG_NIS_ROUTER	NIS router module
	26	LOGDIAG_NIS_TOKEN	NIS token module
	27	LOGDIAG_NIS_PARSE	NIS parse module
	28	LOGDIAG_NIS_REG	NIS registration facility
	Minor Type 11 – Host LAN Connection (HLCN)		
	Subtype	**JOBFORMATTER Define**	**Meaning**
	0	LOGDIAG_HLCN_PIM	PIM module
	1	LOGDIAG_HLCN_PIE	PIE module
	2	LOGDIAG_HLCN_NM	Network manager module
	3	LOGDIAG_HLCN_OI	Operator interface module
	4	LOGDIAG_HLCN_DM	Dialog manager module
	5	LOGDIAG_HLCN_NWMGR	NWMGR module

continued

Table 12–152. Major Type 19–Minor Types 1, 4, 5, 6, 7, 11, 12, 13, and 16 (cont.)

Minor Type = 1 Distributed Systems Service (LOGDIAG_DSS)
Minor Type = 4 Message Handling System (LOGDIAG_MHS)
Minor Type = 5 Open Systems Interconnection (LOGDIAG_OSI)
Minor Type = 6 TCP/IP (LOGDIAG_TCPIP)
Minor Type = 7 Network-Independent Software (LOGDIAG_NIS)
Minor Type = 11 Host LAN Connection (LOGDIAG_HLCN)
Minor Type = 12 OSI Directory (LOGDIAG_DIR)
Minor Type = 13 Network Management System (LOGDIAG_NMS)
Minor Type = 16 Systems Network Architecture (LOGDIAG_SNA)

Words	Description		
	6	LOGDIAG_HLCN_STREAMS	STREAMS module
	7	LOGDIAG_HLCN_SLOOP	SLOOP module
	8	LOGDIAG_HLCN_BOUNCE	BOUNCE module
	9	LOGDIAG_HLCN_IPX	IPX module
	10	LOGDIAG_HLCN_NETBIOS	NetBIOS module
	11	LOGDIAG_HLCN_NBIX	NBIX module
	12	LOGDIAG_HLCN_ETH	Ethernet driver
	13	LOGDIAG_HLCN_SPX	SPX module
	14	LOGDIAG_HLCN_NBDG	NetBIOS datagram module
	15	LOGDIAG_HLCN_INTSWP	Integer swap module
	16	LOGDIAG_HLCN_MACSWP	Medium access control swap module
	17	LOGDIAG_HLCN_SWPTEST	Swap test module
	Minor Type 12 – OSI Directory (DIR)		
	Subtype	**JOBFORMATTER Define**	**Meaning**
	0	LOGDIAG_DIR_DUA	Directory user agent
	1	LOGDIAG_DIR_DSA	Directory system agent
	2	LOGDIAG_DIR_DUACTL	DUA control modules
	3	LOGDIAG_DIR_DUACMN	DUA common modules

continued

Table 12–152. Major Type 19–Minor Types 1, 4, 5, 6, 7, 11, 12, 13, and 16 (cont.)

Minor Type = 1 Distributed Systems Service (LOGDIAG_DSS)
Minor Type = 4 Message Handling System (LOGDIAG_MHS)
Minor Type = 5 Open Systems Interconnection (LOGDIAG_OSI)
Minor Type = 6 TCP/IP (LOGDIAG_TCPIP)
Minor Type = 7 Network-Independent Software (LOGDIAG_NIS)
Minor Type = 11 Host LAN Connection (LOGDIAG_HLCN)
Minor Type = 12 OSI Directory (LOGDIAG_DIR)
Minor Type = 13 Network Management System (LOGDIAG_NMS)
Minor Type = 16 Systems Network Architecture (LOGDIAG_SNA)

Words	Description		
	4	LOGDIAG_DIR_DUARMT	DUA remote modules
	5	LOGDIAG_DIR_DSAPRS	DSA parsing modules
	6	LOGDIAG_DIR_DSACNT	DSA control modules
	7	LOGDIAG_DIR_DSADIB	DSA database modules
	8	LOGDIAG_DIR_CSERVER	Communication server
	9	LOGDIAG_DIR_DIBACCESS	Database access modules
	Minor Type 13 – Network Management System (NMS)		
	Subtype	**JOBFORMATTER Define**	**Meaning**
	0	LOGDIAG_NMS_OTHER	Miscellaneous entries
	1	LOGDIAG_NMS_LCF	Local Control Facility entries
	2	LOGDIAG_NMS_NCF	Network Control Facility entries
	3	LOGDIAG_NMS_SNMP	Entries from the SNMP agent
	Minor Type 16 – Systems Network Architecture (SNA)		
	Retrieve all diagnostic information by using the LOG DIAG SNA command of LOGANALYZER.		

continued

Table 12–152. Major Type 19–Minor Types 1, 4, 5, 6, 7, 11, 12, 13, and 16 (cont.)

Minor Type = 1 Distributed Systems Service (LOGDIAG_DSS) Minor Type = 4 Message Handling System (LOGDIAG_MHS) Minor Type = 5 Open Systems Interconnection (LOGDIAG_OSI) Minor Type = 6 TCP/IP (LOGDIAG_TCPIP) Minor Type = 7 Network-Independent Software (LOGDIAG_NIS) Minor Type = 11 Host LAN Connection (LOGDIAG_HLCN) Minor Type = 12 OSI Directory (LOGDIAG_DIR) Minor Type = 13 Network Management System (LOGDIAG_NMS) Minor Type = 16 Systems Network Architecture (LOGDIAG_SNA)	
Words	**Description**
5–end	Variable-length data. Field [15:4] of Word 4 specifies the format of the variable-length data. If the format is raw binary data, then field [47:13] of Word 4 specifies the length of the data in bytes. If the format is formatted text, then field [47:13] of Word 4 specifies the total number of lines of text. The first byte of each line stores a value indicating the length of the line in bytes, not including the length byte itself.

Major Type = 25 Host LAN Connection (LOGMAJ_HLCN)

Log entries of Major Type 25 record system actions that are related to the Host LAN Connection (HLCN) product.

The LINK words used in Major Type 25 log entries follow the same format as standard LINK words, except that the length field can be expressed in either bytes or words. In Tables 12–153 through 12–158, the individual log entry descriptions specify the units used for each LINK word. The following is the format:

Field	Meaning
[39:20]	Length of this item, in bytes or words
[19:20]	Index of word in log entry where this item starts. It is found in the variable part of the log entry.

Table 12–153. Major Type 25–Minor Type 1

Minor Type = 1 Change Network Processor (LOGHLCN_CHANGE_NP);	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Network processor unit number
5	LINK to firmware file title. The LINK length is in units of bytes. If the firmware file title has not been changed, the length is 0. The file title is in display form.
6	LINK to node address. The LINK length is in units of bytes. If the node address has not been changed, the length is 0. The node address is in EBCDIC form.
7	LINK to network number. The LINK length is in units of bytes. If the network number has not been changed, the length is 0. The network number is in EBCDIC form.
8	LINK to internal network number. The LINK length is in units of bytes. If the internal network number has not been changed, the length is 0. The network number is in EBCDIC form.
9	[3: 4] FRAME –1 = Frame was not changed. 1 = ETHERNET_802.3 2 = ETHERNET_II
10–end	Variable-length data pointed to by the LINKs in this record

Table 12–154. Major Type 25–Minor Type 2

Minor Type = 2 Network Processor Report (LOGHLCN_NP_REPORT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Network processor unit number
5	LINK to report. The LINK length is in units of bytes. The report is in EBCDIC form.
6–end	Variable-length data pointed to by the LINKs in this record

Table 12–155. Major Type 25–Minor Types 3 and 4

Minor Type = 3 Add NETACCESSPOINT (LOGHLCN_ADD_NETACCESSPOINT) Minor Type = 4 Change NETACCESSPOINT (LOGHLCN_CHANGE_NETACCESSPOINT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to NETACCESSPOINT name. The LINK length is in units of bytes. The NETACCESSPOINT name is in EBCDIC form.
5	LINK to network processor unit number list. The LINK length is in units of words. Within the unit number list, each word stores the unit number of a single network processor.
6–end	Variable-length data pointed to by the LINKs in this record

Table 12–156. Major Type 25–Minor Type 5

Minor Type = 5 Delete NETACCESSPOINT (LOGHLCN_DELETE_NETACCESSPOINT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to NETACCESSPOINT name. The LINK length is in units of bytes. The NETACCESSPOINT name is in EBCDIC form.
5–end	Variable-length data pointed to by the LINKs in this record

Table 12–157. Major Type 25–Minor Type 6

Minor Type = 6 Reset Router (LOGHLCN_RESET_ROUTER)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	Network processor identification, expressed as an integer

Table 12–158. Major Type 25–Minor Type 7

Minor Type = 7 Network Manager Report (LOGHLCN_NM_REPORT)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4	LINK to network manager report. The LINK length is in units of bytes. The report is in EBCDIC form.

Major Type = 27 TCP/IP (LOGMAJ_TCPIP)

Log entries of Major Type 27 record all nondiagnostic commands, responses, and reports generated by the TCP/IP network provider.

The *BNA Version 2 Network Encoded Messages Programming Reference Manual, Volume 2* explains the format of the log records after Word 3.

Table 12–159. Major Type 27–Minor Types 1 through 11

Minor Type = 1 Transmission Control Protocol (TCP) layer
Minor Type = 2 Internet Protocol (IP) layer
Minor Type = 3 Address Resolution Protocol (ARP) function
Minor Type = 4 Route Information Protocol (RIP) function
Minor Type = 5 User Datagram Protocol (UDP) layer
Minor Type = 6 Internet Control Message Protocol (ICMP) function
Minor Type = 7 TCP Manager (TCPMGR) function
Minor Type = 8 Connection Open and Close (PIM) reports
Minor Type = 9 Reserved
Minor Type = 10 Entries Generated During the Process of an SNMP Command
Minor Type = 11 Miscellaneous (OTHER)

Words	Description
0–3	As described in Table 12–2, "First Four Log Entry Words"
4– end	Variable-length data.

Major Type = 28 Network Management (LOGMAJ_NMS)

Log entries of Major Type 28 record all nondiagnostic commands, responses, and reports generated for Network Management.

The *BNA Version 2 Network Encoded Messages Programming Reference Manual, Volume 2* explains the format of the log records after Word 3.

Table 12–160. Major Type 28–Minor Types 1 through 4

Minor Type = 1 Local Control Facility (LCF) **Minor Type = 2 Network Control Facility (NCF)** **Minor Type = 3 SNMP Agent (SNMP)** **Minor Type = 4 Miscellaneous (OTHER)**	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4– end	Variable-length data.

Major Type = 30 Systems Network Architecture (LOGMAJ_SNA)

Log entries of Major Type 30 record all nondiagnostic commands, responses, and reports generated for Systems Network Architecture (SNA).

Table 12–161. Major Type 30–Minor Types 1 through 3

Minor Type = 1 Noteworthy network event (REPORT) Minor Type = 2 Network frame reported as a result of the TRACE + command Minor Type = 3 Unexpected fault condition (ERROR) Minor Type = 4 Solicited status request and response (INQUIRY) Minor Type = 5 Solicited action request and response (COMMAND) Minor Type = 6 Miscellaneous (OTHER)	
Words	**Description**
0–3	As described in Table 12–2, "First Four Log Entry Words"
4– end	Variable-length data.

Section 13
System Stability Reporting

One of the fundamental performance characteristics of a computing system is its stability. Stability is a measure of how consistently a system provides service to users. System stability is of concern to customers because system downtime can affect the business value of time-critical applications. Computer vendors in turn are concerned about system stability because it affects customer satisfaction.

To promote system stability, it is essential that the computer vendor collect reliable information about the stability of systems at customer sites. This information can be used to identify areas in the hardware and software where reliability could be enhanced.

System stability reporting (SSR) is a process by which Unisys Corporation collects system stability information from customer sites. The information collected is known as SSR data. As of the current Mark release, SSR has been implemented for A 12, A 15, A 16, and A 17 systems.

The following are the elements of the SSR process:

- Automatic system stability logging, which stores information about service interruptions in the stability log. The stability log is discussed under "Understanding the Stability Log" in this section.
- RSVP messages, which prompt an operator to enter information after each halt/load. These are discussed under "Recording the Initial Halt/Load Description" in this section.
- Customer and Unisys evaluation. The customer and the Unisys site representative each use the SSR interactive support tool (ISTUTILITY) to review and comment on the contents of the stability log. ISTUTILITY is discussed under "Viewing and Commenting On Stability Data" in this section.
- Data transfers. In the United States, the customer periodically transmits a copy of the stability log to the Unisys customer service center (CSC), using a data communications link that exists in the maintenance subsystem of the customer system. On A 16 systems, customers can use ISTUTILITY to manually initiate data transfers or to schedule automatic data transfers at specified intervals. If you use ISTUTILITY on an A 12, A 15, or A 17 system, contact your Unisys site representative for information about how to transfer stability data.

Understanding the Stability Log

The stability log is a file dedicated solely to storing stability information about the system. The stability log is separate from the system summary log (SUMLOG). Because the stability log is dedicated to stability information, it can store information covering a long period of time while still remaining a manageable size.

Structure and Capacity

The stability log is of a fixed size, and is divided into several sections that also have a fixed size. After each section becomes full, the system begins overwriting the oldest entries in the section. The stability log has the following capacities for the various types of information stored:

- Up to 300 halt records can be stored. For information about the contents of halt records, refer to "Reviewing and Entering Halt Data" later in this section.
- Up to 365 days' worth of internal failure summaries can be stored. For information about the contents of internal failure summaries, refer to "Viewing Internal Failure Summaries" later in this section.
- Up to 30 different daily comments can be stored. The comments can be spread over a period of more than 30 days if comments are not entered every day. For information about daily comments, refer to "Reviewing and Entering Daily Comments" later in this section.

Where the Log Resides

The stability log resides on the halt/load family that was in effect at the time of the most recent halt/load. The stability log file is titled *SYSTEM/STABILITYLOG. The stability log file cannot be removed.

If the HLUNIT (Specify Halt/Load Unit) system command is used to specify a new halt/load family, then a new stability log is opened on the new halt/load family after the next halt/load. As a result, any family that has been used as a halt/load family might contain a stability log file. If, following a halt/load, the system finds an invalid stability log file titled *SYSTEM/STABILITYLOG on the halt/load family, then the system renames the file *SYSTEM/STABILITYLOG/BAD and opens a new stability log file titled *SYSTEM/STABILITYLOG.

When the system opens a new stability log file on a new halt/load family after a halt/load, the system is unable to complete the halt record for the halt/load until a subsequent HLUNIT system command and halt/load cause the stability log file on the original family to be used again. If the halt/load that returns the original stability log to use takes place within 12 hours of the original halt/load, the system displays the RSVP questions described under "Recording the Initial Halt/Load Description," later in this section. If the elapsed time has been greater than 12 hours, however, the system displays no RSVP messages, because the operator might be unsure which halt/load the RSVP messages refer to.

Before changing to a new halt/load family, transfer the data in the stability log to the customer support center (CSC). On A 16 systems, you can initiate a manual data transfer from the Transfer Schedule Manual screen of ISTUTILITY, as discussed under "Initiating and Scheduling Data Transfers" later in this section.

If you create a halt/load family for a new system by copying all SYSTEM files from the halt/load family of an existing system, remove the *SYSTEM/STABILITYLOG file from the new family before that family is used as a halt/load family. This step is advisable because the old *SYSTEM/STABILITYLOG file contains stability data for the old system. By removing this file, you ensure that the MCP creates a new stability log when the MCP is first initialized from the new halt/load family.

Recording the Initial Halt/Load Description

Following a halt/load, the system uses RSVP messages to prompt the operator to enter information about whether the halt/load was scheduled, and whether the halt/load was attributed to the customer or to a problem in a Unisys product. The system records the operator responses in the stability log. The customer and the Unisys representative can use ISTUTILITY later to expand or modify this basic information, as discussed under "Reviewing and Entering Halt Data" later in this section.

The RSVP messages are as follows:

- After the halt/load, a process with the following RSVP message appears in the W (Waiting Mix Entries) system command display:

  ```
  MLSACCEPT:WAS THIS HALTLOAD A SCHEDULED INTERRUPTION ? (YES OR NO)
  ```

  Scheduled interruptions include interruptions for maintenance purposes, such as changing the microcode or MCP. Unscheduled interruptions include fatal memory dumps and interruptions caused by power failures. The operator can respond to this message with an AX (Accept) system command of the form *<mix number> AX YES* or *<mix number> AX NO*.

- A process with the following RSVP message then appears in the W command display:

  ```
  MLSACCEPT:SHOULD THE HALTLOAD BE ALLOCATED TO UNISYS ? (YES OR NO)
  ```

  The operator must enter his or her perception of whether the interruption is due to a Unisys hardware or software failure or due to some other factor that is peculiar to the site, such as a power failure. Once again, the operator can respond with a *<mix number> AX YES* or a *<mix number> AX NO* command.

The operator can cause the system to skip the second RSVP message by entering a complex response to the first RSVP message. The complex response contains two YES or NO answers, separated by a space or a comma (,). For example, if the operator answers the first RSVP message with *<mix number> AX YES YES*, then the system records a YES response to both questions but never actually displays the second RSVP message.

Viewing and Commenting On Stability Data

The interactive support tool (ISTUTILITY) is a utility program that allows you to review automatically collected SSR data and to add comments of your own. For example, you can use ISTUTILITY to do any of the following:

- Record whether a system halt is caused by an unplanned system failure or a planned halt/load to reload the operating system.
- Record whether a system halt is the result of an equipment failure or the result of losing electrical power to the system.
- Record observed symptoms relating to a particular halt.
- Record the amount of time the system is unavailable for production use.
- Identify a particular system element whose failure is suspected of having caused a system halt.
- Correlate two or more halts that have similar symptoms.

Installing ISTUTILITY

ISTUTILITY is part of the general A Series system software release. You can install ISTUTILITY using the Simple Installation (SI) program described in the *A Series Software Release Installation Guide.* Simple Installation (SI) installs ISTUTILITY as part of the System Software Utilities (SSU) product.

SI installs the following files, which must both be present in order for ISTUTILITY to function: SYSTEM/ISTUTILITY and SYSTEM/SSRSUPPORT. SI also marks SYSTEM/SSRSUPPORT as a privileged program, and associates SYSTEM/SSRSUPPORT with the SL function name SSRSUPPORT.

Installing the Transfer Schedule Feature

A 16 systems support automatic scheduled transfers of system stability data to the Unisys customer support center (CSC). To make use of the transfer schedule feature, you must first configure the system console and the A Series host properly. You can use the following steps to configure the system console to support the transfer schedule feature:

1. Install console software of level 12.008 or greater. Refer to the *A 16/A 19 System Console Operations Guide* for details.
2. Create a console network description file for system stability reporting. To do this, contact a Unisys customer support representative, who will assist with the creation of a network description file. The customer support representative might be able to provide a floppy disk with an appropriate network description file. You can use the Transfer function of the Installer console screen to copy this file onto the fixed disk.

You might have to make minor changes to an existing network description file, such as changes to the data terminal equipment (DTE) address. Alternatively, you might have to create a new network description file. You can modify or create network description files by using the Configure Network field on the RSC Connect console screen.

3. Declare the SSR network description file name by using the Change SSR Network field on the RSC Connect screen. The SSR network name does *not* have to match a configured remote supervisory console (RSC) network name.
4. Set the Auto Dial Enable field on the Network-Layer 2 console screen to a value of *2* for SSR network description files.
5. Declare the system console transport (SCX) unit in the peripheral configuration diagram (PCD). Declare the SCX unit with unit position 14 on the CON1 control. Then attach the new PCD to the IOM or IOMs by using the IOM Features console screen.

You can use the following steps to configure the A Series host to support the transfer schedule feature:

1. Load the system from the console.
2. Acquire the SCX unit into the group by using the *ACQUIRE SC <unit number>* form of the ACQUIRE (Acquire Resource) system command. If the SCX is not ready, then ready the unit by using the *RY SC <unit number>* form of the RY (Ready) system command.
3. Verify that the SCX unit is present by entering the *PER SC* form of the PER (Peripheral Status) system command. The SCX unit, if it is present, is labeled *CMPLINK*.
4. Scan the MSG (Display Messages) system command output until you see the following message:

   ```
   CONSOLE TRANSPORT SERVICE IS NOW AVAILABLE.
   ```

5. Initiate a manual data transfer to verify that the console and the system hardware and software are functioning properly. You can initiate the manual data transfer from the Transfer Schedule Manual screen of ISTUTILITY. The following message appears in the MSG display within one or two minutes of the initiation:

   ```
   STABILITYLOG MANUAL TRANSFER INITIATED
   ```

6. If the transfer completes successfully, the following message appears in the MSG display:

   ```
   STABILITYLOG TRANSFER SUCCESSFUL
   ```

   Once a manual transfer has completed successfully, define a schedule for automatic data transfers with the Auto Transfer Schedule menu of ISTUTILITY.

7. If the transfer does not complete successfully, the following message appears in the MSG display:

   ```
   STABILITYLOG TRANSFER FAILED
   ```

   Contact the CSC for assistance in remedying the problem.

Initiating ISTUTILITY

You can use ISTUTILITY at any T27 or compatible terminal. To run ISTUTILITY, log on to a CANDE or MARC session and enter the following command:

```
RUN *SYSTEM/ISTUTILITY ON <family>
```

If you enter this command in MARC, you should enter it in the Action field rather than the Choice field.

Using ISTUTILITY Screens

Every screen used in ISTUTILITY contains an Action field and an XMIT Here field. These fields are illustrated in Figure 13–1.

```
Interactive Support Tool                            Tue Nov 19 17:37:17 1991
                                  Main Menu
   [ ] Action
       Redisplay  Help  Exit


                              [1] Select Function
                               ** 1 - Review/Evaluate Halt Data
                                  2 - View Summary Data
                                  3 - Review/Enter Daily Comment
                                  4 - Collection Configuration
                                  5 - Transfer Schedule









                                                             XMIT here [@]
```

Figure 13–1. Main Menu

The Action field is a 1-character field that accepts the first character of the actions listed immediately below it.

Note: *On most screens, you can leave the Action field blank, in which case the next screen in the menu sequence appears when you press the Transmit key. However, you must make an entry in the Action field to exit comment screens such as the Daily Comment screen.*

The following are the actions available from most screens, and their effects:

Action	Effect
Cancel	Ignores any data that you have typed into fields on the screen. Cancels the activity that is normally initiated by the current screen and redisplays the screen that initiated the current activity. This screen might be the previous screen, or it might be one several screens back.
Exit	Inspects the data on the screen, accepts the data if valid, and exits ISTUTILITY. The user is returned to the environment from which ISTUTILITY was executed (for example, CANDE or MARC).
Help	Temporarily saves any data you have typed on the screen and displays help text for the current screen. You can return to the current screen by transmitting the help screen with a C (for cancel) or a blank in the Action field.
Previous	Inspects the data on the current screen. If the data is valid, accepts the data and returns to the screen displayed prior to this one. If the data is invalid, redisplays the current screen with an error message so that you can correct the unacceptable data.
Redisplay	Ignores the data on the screen. Redisplays the current screen as it was originally displayed.

Below the Action field, many of the screens include one or more special-purpose fields. For example, the Main Menu includes the Select Function field.

After making all your entries to the Action field and any special-purpose fields, you should place the cursor in the XMIT Here field before transmitting. This cursor position ensures that a valid screen is transmitted to the system.

Using the Main Menu Screen

When ISTUTILITY initiates, it displays the Main Menu screen shown in Figure 13–1. From the Main Menu screen, you can choose any of the following functions:

Selection	Meaning
Review/Evaluate Halt Data	Review or add information to halt/load descriptions automatically collected by the system.
View Summary Data	Review internal fault descriptions automatically collected by the system.
Review/Enter Daily Comment	Enter free-formatted comments describing any issues related to system stability for a particular day.

continued

continued

Selection	Meaning
Collection Configuration	Specify which types of halts the customer is asked to review when using ISTUTILITY. This menu selection is intended for use by the Unisys representative.
Transfer Schedule	Initiate a manual transfer of stability data to a CSC, or create a schedule for automatic data transfers. This selection is currently available only on A 16 systems.

The following subsections describe the functions provided by these menu selections in more detail.

Reviewing and Entering Halt Data

You can use the Review/Evaluate Halt Data selection on the Main Menu to review halt item descriptions. This selection causes ISTUTILITY to display the Review Halt Data screen shown in Figure 13–2.

```
Interactive Support Tool                              Fri Jul 27 12:10:45 1990
                                  Review Halt Data
   [ ] Action
       Redisplay  Cancel  Help  Exit

            [1] Select review type
                1 - Review site evaluation incomplete halt records.
                2 - Review Unisys evaluation incomplete halt records.
                3 - Review halts attributed to the site.
                4 - Review halts attributed to Unisys.
                5 - Manual selection of halts for review.

                    Halt data is available for the period of

                           03/19/1990 to 07/25/1990

                                  (mm/dd/yyyy)

                           Incomplete halt records:
                152  Site evaluations    153  Unisys evaluations
                                                                XMIT here [@]
```

Figure 13–2. Review Halt Data Screen

Selecting Criteria for Halt Record Display

You can use the Select Review Type field of this screen to specify the types of halt records to be reviewed. You can enter *1* to select those records that require customer evaluation, or *2* to select those records that require Unisys evaluation. Not all halt records require evaluation; refer to "Configuring the Halt Review Criteria" in this section for further information.

Entering *3* or *4* selects halt records based on whether they are attributable to the customer or to Unisys. Entering *5* (for Manual Selection of Halts for Review) causes the display of the Halt Selection Items screen, which allows you to specify additional criteria for the selection of halt records.

Reviewing the Halt Records

After you finish selecting the criteria for halt record display, ISTUTILITY begins displaying a series of halt records. Each halt record display begins with two Collected Halt Status screens that display the information automatically collected by the system. This information includes the following items:

- The date and time the system recovered from the halt.
- The halt type. The halt type indicates whether the halt is initiated internally by the hardware or software, initiated by manual intervention by the operator, or initiated by an unknown cause.
- The system command used to reload the system.
- Whether the halt was scheduled or unscheduled, and whether the halt is attributable to Unisys or the site. These items of information are based on the operator responses discussed under "Recording the Initial Halt/Load Description" in this section.
- Whether or not the ODT was responding to the mainframe at the time of the halt.
- Whether or not the system running indicator was present at the time of the halt.
- The number of processors operating in the system at the time of the halt.
- The MCP level in use at the time of the halt.
- Whether or not the halt has been evaluated by Unisys.
- Whether or not the halt has been evaluated by the site.
- Whether or not halt comments have been entered by either Unisys or the site.
- The number of minutes the MCP was running as a single-processor system before the system halted.
- The number of minutes the MCP was running as a multiprocessor system before the system halted.
- The primary reload time. This is the time taken to restart the MCP.
- The secondary reload time. This is the time taken to restart and recover user applications necessary for production to begin.
- The general, and if known, the specific system element that caused the halt. This might be a hardware or software component.
- Whether or not a halt is believed to be related to another halt.

After viewing the first Collected Halt Status screen for a halt, you can advance to the next screen by transmitting the current screen with a blank in the Action field. After viewing the second Collected Halt Status screen, you can can again transmit the blank action to advance to the Halt Evaluations screen.

Reviewing and Entering the Halt Evaluation Data

After the last of the Collected Halt Status screens for a halt record, ISTUTILITY displays the Halt Evaluations screen. This screen enables you to review or enter site evaluation information for the halt, and to review Unisys evaluation information for the halt. Once you transmit the Halt Evaluations screen, ISTUTILITY displays several screens including fields for the following types of evaluation information:

- Whether the halt is scheduled or not scheduled.
- Whether the halt is attributable to Unisys, the customer site, or an unknown source.
- The impact of the halt on the site. Measures of impact are
  - *None*. The halt occurred when the system was not in production.
  - *Minimal*. The halt caused very little disruption.
  - *Moderate*. The halt occurred while the system was not in prime production time.
  - *Severe*. The halt occurred while the system was in prime production time.
- Whether the halt was deferred or not. A deferred halt occurs when, in response to a fault condition in the system, a halt is postponed to a less disruptive time.
- The system activity occurring when the halt happened. For example, this activity could be any of the following:
  - Normal production
  - Changing the physical hardware configuration
  - Changing the logical configuration of the software
  - Performing maintenance of system elements
- The general, and if known, the specific system element that caused the halt. This might be a hardware or software component.
- The estimated time that the system is down. Downtime is the elapsed time that the system is unavailable for processing.
- The RESPOND reference number assigned by Unisys.
- The CSC CONTACT number that Unisys assigns if the support center is contacted.
- The UCF number that Unisys assigns if a UCF is submitted against the halt.
- Identification of a related halt, if there is one. The related halt identification consists of the site identifier assigned by Unisys, the system identifier, and the halt number.

When you are first entering evaluation information for a particular halt, you should fill out the required fields on each screen and leave the Action field blank. When you transmit the screen, ISTUTILITY displays the next screen in the sequence.

When you complete the last screen in the sequence, ISTUTILITY returns you to the Halt Evaluations screen for the current halt. You can then step through the sequence of screens again to review the information you previously entered. Simply transmit each screen with the Action field left blank to review the next screen in the sequence.

While you are entering halt evaluation information, you might wish to also enter a halt comment. This procedure is described in the following subsection.

When you have entered all the information you wish to for a given halt, you can advance to the next halt by using the Next Halt action, or by entering *3* (for Continue) in the Select Function field of the Halt Evaluations screen.

To stop evaluating halts, you can use the Cancel action or enter *4* (for Complete) in the Select Functions field of the Halt Evaluations screen. Or, you can use the Exit action to exit ISTUTILITY.

Entering Halt Comments

You might wish to enter free-formatted comments about the halt. Such comments can describe specific conditions under which the halt occurred. Comments can also describe any symptoms of the problem causing the halt or any trouble encountered in bringing the system back up.

To enter free-formatted comments about a halt, you must enter an *A* for Add Comment in the Action field of the Collected Halt Status screen, the Halt Evaluations screen, or any of the subsequent screens in the sequence. This action causes the Halt Comment screen to be displayed. You can enter your comments on this screen.

If you leave the Action field on the Halt Comment screen blank, then ISTUTILITY simply redisplays the Halt Comment screen when you transmit. You can then revise your comments if desired. When you are satisfied with your comments, you should transmit the screen with a *D* for Done in the Action field. The Done action returns you to the previous screen.

Viewing Internal Failure Summaries

You can use the View Summary Data selection on the Main Menu to view summaries of internal system failures. This selection causes ISTUTILITY to display the View Summary Data Screen shown in Figure 13–3.

```
Interactive Support Tool                          Fri Jul 27 12:11:06 1990
                                     View Summary Data
   [ ] Action
       Redisplay  Cancel  Help  Exit


                        Summary data is available for the period of

                                  03/18/1990  to  07/27/1990


                            View summary data for the period of

                                  [          ] to [          ]

                                          (mm/dd/yyyy)



                                                                  XMIT here [@]
```

Figure 13–3. View Summary Data Screen

System delays are the only type of failure currently recorded. A delay is a time interval in which no programs are being processed except those programs supporting the diagnostic or recovery work. Delays are recorded when they exceed 10 seconds, as in the case of a nonfatal memory dump. Delays are also recorded when the accumulation of multiple delays exceeds 3 minutes in a given hour.

You can use the View Summary Data screen to select the range of dates for which you want to view summaries. After you transmit this information, ISTUTILITY displays a series of Summary Data screens. Each Summary Data screen displays information about all failures of a given type that occurred on a particular day. For example, all system delays caused by the same system element on the same day are summarized on a single Summary Data screen.

The Summary Data screens display the following items:

- The date for which the failures are summarized.
- The failure type being summarized.
- If applicable, the system element that caused the failure.
- If known, the reason for the failure. For delays, the reason for the delay is given.
- The number of failure occurrences in this category that occurred on the indicated date.
- For the failure type of delay, the accumulated duration of delays, measured in seconds, for the specified date.

Reviewing and Entering Daily Comments

Daily comments are free-formatted comments that you can use to record any system activity or circumstances related to system stability. You can use the Review/Enter Daily Comment selection on the Main Menu to review, add, or update daily comments. This selection causes ISTUTILITY to display the Review Daily Comments screen shown in Figure 13–4.

```
Interactive Support Tool                          Fri Jul 27 12:11:17 1990
                                   Review Daily Comments
   [ ] Action
       Add comment  Redisplay  Cancel  Help  Exit


                       Daily comments are available for the period of

                                03/18/1990  to  03/20/1990


                         View/enter daily comments for the period of

                               [         ] to [         ]

                                       (mm/dd/yyyy)



                                                                XMIT here [@]
```

Figure 13–4. Review Daily Comments Screen

If you are interested in adding or modifying daily comments, you should

- Enter *A* (for Add comment) in the Action field.
- Enter a range of dates in the fields below the *View/enter daily comments for the period of* prompt.
- Transmit the screen.

ISTUTILITY responds by displaying a series of Daily Comment screens, one for each date in the range. On each screen, an input field appears that enables you to enter comments or modify existing comments. When you transmit a Daily Comment screen with a D (for Done) in the Action field, ISTUTILITY displays the Daily Comment screen for the next date.

If you are interested in viewing comments, and are sure that you do not want to modify any of the comments, then you should transmit the Review Daily Comments screen with a date range filled in and the Action field left blank. ISTUTILITY displays a series of Daily Comment screens that list the comments without allowing the comments to be modified.

Configuring the Halt Review Criteria

A Unisys representative can specify the criteria for halt evaluation collection by entering the Collection Configuration selection on the Main Menu. This entry causes the display of the Collection Configuration screen shown in Figure 13–5.

```
Interactive Support Tool                        Fri Jul 27 12:11:28 1990
                             Collection Configuration
   [ ] Action
       Redisplay  Cancel  Help  Exit

                 Site review will be requested for:

                   [*] Scheduled halts attributed to Unisys.
                   [ ] Scheduled halts attributed to the site.
                   [*] Unscheduled halts attributed to Unisys.
                   [*] Unscheduled halts attributed to the site.

                 Unisys review will be requested for:

                   [ ] Scheduled halts attributed to Unisys.
                   [ ] Scheduled halts attributed to the site.
                   [ ] Unscheduled halts attributed to Unisys.
                   [ ] Unscheduled halts attributed to the site.

                                                          XMIT here [@]
```

Figure 13–5. Collection Configuration Screen

A Unisys representative uses the Collection Configuration screen to select the types of halts you are asked to review when you enter *2* (Review Site Evaluation Incomplete Halt Records) on the Review Halt Data screen. Additionally, you can use the Collection Configuration screen to add to the types of halts that you review. The types of halts to select from are the following:

- Scheduled halts attributable to Unisys
- Scheduled halts attributable to the customer site
- Unscheduled halts attributable to Unisys
- Unscheduled halts attributable to the customer site

Initiating and Scheduling Data Transfers

On A 16 systems with the transfer schedule feature installed, you can use the Transfer Schedule selection on the Main Menu to transfer data to a CSC. For information about how to install the Transfer Schedule feature, refer to "Installing the Transfer Schedule Feature" earlier in this section. The Transfer Schedule selection causes ISTUTILITY to display the Transfer Schedule screen shown in Figure 13–6.

```
Interactive Support Tool                              Wed Nov 20 14:14:50 1991
                                    Transfer Schedule
   [ ] Action
       Redisplay  Cancel  Help  Exit

                              [1] Select transfer action
                                  1 - Set up auto transfer schedule
                                  2 - Display transfer status
                                  3 - Set up manual transfer






                                                                 XMIT here [@]
```

Figure 13–6. Transfer Schedule Screen

The following are the values you can enter in the Select Transfer Action field, and their effects:

Value	Description
1	ISTUTILITY displays the Transfer Schedule Auto screen. You can use the Transfer Schedule Auto screen to enable or disable automatic data transfers, to specify the time of day that transfers are to occur, and to specify whether the intervals between transfers are to be measured in units of weeks or months. ISTUTILITY displays further screens that ask you to specify the number of weeks or months between transfers, and the days of the week or month when data should be transferred.
2	ISTUTILITY displays the Transfer Schedule Status screen. The Transfer Schedule Status screen provides information about the data transfer currently in progress (if any), the last data transfer to be completed, and the next data transfer that is scheduled. You can also use this screen to specify whether data transfer failures result in a waiting entry being displayed at the ODT. On the Transfer Schedule Status screen, the Calendar Date of Last Completed Auto Transfer field shows the date of the last completed automatic transfer. This field is updated after each successful automatic transfer. You can also manually alter the date in this field if you want to retransmit some of the data, or if you want to compensate because the system date was recently reset backwards. If this date has been manually modified, two asterisks (**) appear next to it on the screen. The asterisks disappear after the next successful automatic transfer.

continued

continued

Value	Description
3	ISTUTILITY displays the Transfer Schedule Manual screen. The Transfer Schedule Manual screen enables you to initiate the immediate transfer of stability data to the CSC. You specify the oldest stability data to be transferred by entering a date in the Enter the Starting Date field. If you fill this field with blanks, the entire stability log is transferred.

If an automatic transfer has already been attempted on the current day, then no additional automatic transfers are initiated on the current day. This restriction applies even if the automatic transfer failed or if you use ISTUTILITY to change the schedule for automatic transfers.

Automatic transfers can occur within a transfer window. This window begins at the scheduled transfer time and ends 2 hours and 50 minutes later. If the first attempt at an automatic transfer fails, then the system might wait for a while and retry at a later time within the transfer window. The system does not attempt an automatic transfer or a retry of an automatic transfer at any time outside this window.

You can initiate a manual transfer at any time, except when a manual or automatic transfer is already in progress, or when an automatic transfer is waiting to retry after a failure. A manual transfer terminates immediately if it fails, and is not retried.

If a manual transfer is in progress at the time that an automatic transfer is scheduled to begin, then the automatic transfer is initiated immediately after the manual transfer completes.

If a transfer is currently in progress, any attempt to modify the calendar date of the last completed automatic transfer is rejected.

If an automatic transfer fails because it is unable to connect to the customer support center, an automatic transfer is scheduled for the following day at the regularly scheduled time. An automatic transfer continues to be scheduled every day until one of the following occurs:

- An automatic transfer succeeds.
- An automatic transfer fails for some reason other than a connect error.
- A new automatic transfer schedule is entered by way of the ISTUTILITY.

Note: *Reentering the current automatic transfer schedule has the side effect of stopping the daily rescheduling after a connect failure. However, there is no way to permanently turn off the mechanism that does this daily rescheduling.*

Appendix A
Understanding Railroad Diagrams

What Are Railroad Diagrams?

Railroad diagrams are diagrams that show you the rules for putting words and symbols together into commands and statements that the computer can understand. These diagrams consist of a series of paths that show the allowable structure, constants, and variables for a command or a statement. Paths show the order in which the command or statement is constructed. Paths are represented by horizontal and vertical lines. Many railroad diagrams have a number of different paths you can take to get to the end of the diagram. For example:

```
— REMOVE ─┬──────────┬──────────────────────────────────────┤
          ├─ SOURCE ─┤
          └─ OBJECT ─┘
```

If you follow this railroad diagram from left to right, you will discover three acceptable commands. These commands are

- REMOVE
- REMOVE SOURCE
- REMOVE OBJECT

If all railroad diagrams were this simple, this explanation could end here. However, because the allowed ways of communicating with the computer can be complex, railroad diagrams sometimes must also be complex.

Regardless of the level of complexity, all railroad diagrams are visual representations of commands and statements. Railroad diagrams are intended to

- Show the mandatory items.
- Show the user-selected items.
- Present the order in which the items must appear.
- Show the number of times an item can be repeated.
- Show the necessary punctuation.

To familiarize you with railroad diagrams, this explanation describes the elements of the diagrams and provides examples.

Some of the actual railroad diagrams you will encounter might be more complex. However, all railroad diagrams, simple or complex, follow the same basic rules. They all consist of paths that represent the allowable structure, constants, and variables for commands and statements.

By following railroad diagrams, you can easily understand the correct syntax for commands and statements. Once you become proficient in the use of railroad notation, the diagrams serve as quick references to the commands and statements.

Constants and Variables

A constant is an item that cannot be altered. You must enter the constant as it appears in the diagram, either in full or as an allowable abbreviation. If a constant is partially underlined, you can abbreviate the constant by entering only the underlined letters. In addition to the underlined letters, any of the remaining letters can be entered. If no part of the constant is underlined, the constant cannot be abbreviated. Constants can be recognized by the fact that they are never enclosed in angle brackets (< >) and are in uppercase letters.

A variable is an item that represents data. You can replace the variable with data that meets the requirements of the particular command or statement. When replacing a variable with data, you must follow the rules defined for the particular command or statement. Variables appear in railroad diagrams enclosed in angle brackets.

In the following example, BEGIN and END are constants while <statement list> is a variable. The constant BEGIN can be abbreviated since it is partially underlined. Valid abbreviations for BEGIN are BE, BEG, and BEGI.

```
— BEGIN —<statement list>— END ———————————————————————————————|
```

Constraints

Constraints are used in a railroad diagram to control progression through the diagram. Constraints consist of symbols and unique railroad diagram line paths. They include

- Vertical bars
- Percent signs
- Right arrows
- Required items
- User-selected items
- Loops
- Bridges

A description of each item follows.

Vertical Bar

The vertical bar symbol (|) represents the end of a railroad diagram and indicates the command or statement can be followed by another command or statement.

```
— SECONDWORD — ( —<arithmetic expression>— ) ————————————————————┤
```

Percent Sign

The percent sign (%) represents the end of a railroad diagram and indicates the command or statement must be on a line by itself.

```
— STOP ——————————————————————————————————————————————————————————%
```

Right Arrow

The right arrow symbol (>) is used when the railroad diagram is too long to fit on one line and must continue on the next. A right arrow appears at the end of the first line, and another right arrow appears at the beginning of the next line.

```
— SCALERIGHT — ( —<arithmetic expression>— , ————————————————————>
>—<arithmetic expression>— ) ————————————————————————————————————┤
```

Required Items

A required item can be either a constant, a variable, or punctuation. A required item appears as a single entry, by itself or with other items, on a horizontal line. Required items can also exist on horizontal lines within alternate paths or nested (lower-level) diagrams. If the path you are following contains a required item, you must enter the item in the command or statement; the required item cannot be omitted.

In the following example, the word EVENT is a required constant and <identifier> is a required variable:

```
— EVENT —<identifier>————————————————————————————————————————————┤
```

User-Selected Items

User-selected items appear one below the other in a vertical list. You can choose any one of the items from the list. If the list also contains an empty path (solid line), none of the choices are required. A user-selected item can be either a constant, a variable, or punctuation. In the following railroad diagram, either the plus sign (+) or the minus sign (–) can be entered before the required variable <arithmetic expression>, or the symbols can be disregarded because the diagram also contains an empty path.

```
——┬—————┬—<arithmetic expression>————————————————————————————————┤
  ├— + —┤
  └— - —┘
```

Loop

A loop represents an item or group of items that you can repeat. A loop can span all or part of a railroad diagram. It always consists of at least two horizontal lines, one below the other, connected on both sides by vertical lines. The top line is a right-to-left path that contains information about repeating the loop.

Some loops include a return character. A return character is a character – often a comma (,) or semicolon (;) – required before each repetition of a loop. If there is no return character, the items must be separated by one or more blank spaces.

```
  ┌←───── ; ─────┐
──┴─<field value>─┴──────────────────────────────────────────┤
```

Bridge

Sometimes a loop also includes a bridge, which is used to show the maximum number of times the loop can be repeated. The bridge can precede the contents of the loop, or it can precede the return character (if any) on the upper line of the loop.

The bridge determines the number of times you can cross that point in the diagram. The bridge is an integer enclosed in sloping lines (/ \). Not all loops have bridges. Those that do not can be repeated any number of times until all valid entries have been used.

In the first bridge example, you can enter LINKAGE or RUNTIME no more than two times. In the second bridge example, you can enter LINKAGE or RUNTIME no more than three times.

```
  ┌←─────── , ───────┐
──┴─/2\─┬─ LINKAGE ─┬┴──────────────────────────────────────┤
        └─ RUNTIME ─┘
```

```
  ┌←─/2\──────────┐
──┴┬─ LINKAGE ─┬──┴─────────────────────────────────────────┤
   └─ RUNTIME ─┘
```

In some bridges an asterisk (*) follows the number. The asterisk means that you must cross that point in the diagram at least once. The maximum number of times that you can cross that point is indicated by the number in the bridge.

```
  ┌←─────── , ──────────┐
──┴┬─/2*\─ LINKAGE ─┬───┴──────────────────────────────────┤
   └─ RUNTIME ──────┘
```

In the previous bridge example, you must enter LINKAGE at least once but no more than twice, and you can enter RUNTIME any number of times.

The following figure shows the types of constraints used in railroad diagrams.

SYMBOL/PATH	EXPLANATION
`————————┤`	Vertical bar. Indicates that the command or statement can be followed by another command or statement.
`——————— %`	Percent sign. Indicates that the command or statement must be on a line by itself.
`————————>` `>———————`	Right arrow. Indicates that the diagram occupies more than one line.
`—< required >—`	Required items. Indicates the constants, variables, and punctuation that must be entered in a command or statement.
`┬— YES —┬` `└— NO ——┘`	User-selected items. Indicates the items that appear one below the other in a vertical list. You select which item or items to include.
`┌<————┐` `┴—————┴`	A loop. Indicates an item or group of items that can be repeated.
`┌<————┐` `┴— /2\ —┴`	A bridge. Indicates the maximum number of times a loop can be repeated.

Figure A–1. Railroad Constraints

Following the Paths of a Railroad Diagram

The paths of a railroad diagram lead you through the command or statement from beginning to end. Some railroad diagrams have only one path, while others have several alternate paths. The following railroad diagram indicates there is only one path that requires the constant LINKAGE and the variable <linkage mnemonic>:

```
— LINKAGE —<linkage mnemonic>————————————————————————————————┤
```

Alternate paths provide choices in the construction of commands and statements. Alternate paths are provided by loops, user-selected items, or a combination of both. More complex railroad diagrams can consist of many alternate paths, or nested (lower-level) diagrams, that show a further level of detail.

For example, the following railroad diagram consists of a top path and two alternate paths. The top path includes an ampersand (&) and the constants (that are user-selected items) in the vertical list. These constants are within a loop that can be repeated any number of times until all options have been selected. The first alternate path requires the ampersand and the required constant ADDRESS. The second alternate path requires the ampersand followed by the required constant ALTER and the required variable <new value>.

```
        ┌←───────── , ───────┐
— & ──┬─┴─┬─ TYPE ──────────┬┴──┬──────────────────────────────────────────┤
      │   ├─ ASCII ─────────┤   │
      │   ├─ BCL ───────────┤   │
      │   ├─ DECIMAL ───────┤   │
      │   ├─ EBCDIC ────────┤   │
      │   ├─ HEX ───────────┤   │
      │   └─ OCTAL ─────────┘   │
      ├─ ADDRESS ───────────────┤
      └─ ALTER ──<new value>────┘
```

Railroad Diagram Examples with Sample Input

The following examples show five railroad diagrams and possible command and statement constructions based on the paths of these diagrams.

Example 1

<lock statement>

```
— LOCK — ( — <file identifier> — ) ──────────────────────────────┤
```

Sample Input	Explanation
LOCK (FILE4)	LOCK is a constant and cannot be altered. Because no part of the word is underlined, the entire word must be entered. The parentheses are required punctuation, and FILE4 is a sample file identifier.

Example 2

<open statement>

```
— OPEN ─┬─────────────┬─<database name>──────────────────────────┤
        ├─ INQUIRY ───┤
        └─ UPDATE ────┘
```

Sample Input	Explanation
OPEN DATABASE1	The constant OPEN is followed by the variable DATABASE1, which is a database name. The railroad diagram shows two user-selected items, INQUIRY and UPDATE. However, because there is an empty path (solid line), these entries are not required.

continued

continued

Sample Input	Explanation
OPEN INQUIRY DATABASE1	The constant OPEN is followed by the user-selected constant INQUIRY and the variable DATABASE1.
OPEN UPDATE DATABASE1	The constant OPEN is followed by the user-selected constant UPDATE and the variable DATABASE1.

Example 3

<generate statement>

```
— GENERATE —<subset>— = ┬─ NULL ─────────────────────┬──────────────┤
                        └<subset>┬───────────────────┘
                                 ├ AND ─┬<subset>────┘
                                 ├ OR ──┤
                                 ├ + ───┤
                                 └ - ───┘
```

Sample Input	Explanation
GENERATE Z = NULL	The GENERATE constant is followed by the variable Z, an equal sign (=), and the user-selected constant NULL.
GENERATE Z = X	The GENERATE constant is followed by the variable Z, an equal sign, and the user-selected variable X.
GENERATE Z = X AND B	The GENERATE constant is followed by the variable Z, an equal sign, the user-selected variable X, the AND command (from the list of user-selected items in the nested path), and a third variable, B.
GENERATE Z = X + B	The GENERATE constant is followed by the variable Z, an equal sign, the user-selected variable X, the plus sign (from the list of user-selected items in the nested path), and a third variable, B.

Example 4

<entity reference declaration>

```
              ┌←───────────────────── , ────────────────┐
— ENTITY REFERENCE ─┴─<entity ref ID>— ( —<class ID>— ) ─┴──────────┤
```

Sample Input	Explanation
ENTITY REFERENCE ADVISOR1 (INSTRUCTOR)	The required item ENTITY REFERENCE is followed by the variable ADVISOR1 and the variable INSTRUCTOR. The parentheses are required.
ENTITY REFERENCE ADVISOR1 (INSTRUCTOR), ADVISOR2 (ASST_INSTRUCTOR)	Because the diagram contains a loop, the pair of variables can be repeated any number of times.

Example 5

```
— PS — MODIFY ─────────────────────────────────────────────────→

    ┌←───────────────────── , ────────────────────┐
→─┬─┴─┬─<request number>──────────────────────────┴───┬───────────→
  │   └─<request number>— - —<request number>─────────┤
  └─ ALL ─┬─────────────────────────────────────────────┤
          └─ EXCEPTIONS ──────────────────────────────┘

→─┬──────────────────────────────────────────┬───────────────────┤
  │ ┌←──────────────── , ─────────────────┐  │
  └─┴─┬─────┬─<file attribute phrase>─────┴──┘
      │└ - ┘│
      └──┬──┴─<print modifier phrase>─┘
         └ - ┘
```

Sample Input	Explanation
PS MODIFY 11159	The constants PS and MODIFY are followed by the variable 11159, which is a request number.
PS MODIFY 11159,11160,11163	Because the diagram contains a loop, the variable 11159 can be followed by a comma, the variable 11160, another comma, and the final variable 11163.
PS MOD 11159–11161 DESTINATION = "LP7"	The constants PS and MODIFY are followed by the user-selected variables 11159–11161, which are request numbers, and the user-selected variable DESTINATION = "LP7", which is a file attribute phrase. Note that the constant MODIFY has been abbreviated to its minimum allowable form.
PS MOD ALL EXCEPTIONS	The constants PS and MODIFY are followed by the user-selected constants ALL and EXCEPTIONS.

Glossary

A

actual segment descriptor (ASD)

A pointer to the location of a data or code item in memory or on a disk.

ALGOL

Algorithmic language. A structured, high-level programming language that provides the basis for the stack architecture of the Unisys A Series systems. ALGOL was the first block-structured language developed in the 1960s and served as a basis for such languages as Pascal and Ada. It is still used extensively on A Series systems, primarily for systems programming.

ASD

See actual segment descriptor.

B

BDLC

See Data Link Control.

BNA

The network architecture used on A Series, B 1000, and V Series systems as well as CP9500 and CP 2000 communications processors to connect multiple, independent, compatible computer systems into a network for distributed processing and resource sharing.

BNA Version 1

The original version of BNA, which uses data link processors (DLPs) within the host and supports communications between Unisys A Series, B 1000 Series, V Series, and CP9500 systems.

BNA Version 2

A second-generation version of BNA, which uses communications processors such as the CP 2000 for enhanced performance, provides facilities for centralized network administration, and supports multiple types of industry-standard and proprietary networks.

C

CANDE

See Command and Edit.

central processing module (CPM)
The central processor for data processing on host data unit (HDU) and resource management module (RMM) systems.

central processing unit (CPU)
The computer hardware unit that controls and executes the instructions contained in object code files.

COBOL
Common Business-Oriented Language. A widely used, procedure-oriented language intended for use in solving problems in business data processing. The main characteristics of COBOL are the easy readability of programs and a considerable degree of machine independence. COBOL is the most widely used procedure-oriented language.

COBOL74
A version of the COBOL language that is compatible with the American National Standard X3.23-1974.

Command and Edit (CANDE)
A time-sharing message control system (MCS) that enables a user to create and edit files, and to develop, test, and execute programs, interactively.

control (CTL)
The hardware that provides control functions for a peripheral device or string of peripherals. Control (CTL) is a preferred synonym for data link processor (DLP) on resource management module (RMM) systems.

copy descriptor
A duplicate of a mom descriptor except the copy bit is set to 1. A copy descriptor is derived from a mom descriptor, and multiple copy descriptors can reference the same data segment.

CPM
See central processing module.

CPU
See central processing unit.

CSC
See customer service center.

CTL
See control.

customer service center
A subsidiary-level Unisys support office that provides secondary support to customer service personnel.

D

data communications controller (DCC)
The subset of the master control program (MCP) operating as a group of independent tasks, each associated with one network support processor (NSP) or data communications data link processor (DCDLP).

Data Link Control (BDLC)
The bit-synchronous transmission mode used on Unisys A Series systems. BDLC is compatible with the international standard high-level data link control (HDLC) protocol.

data link processor (DLP)
A processor that serves as the system interface to a specific peripheral device, controller, or communications network.

Data Management System II (DMSII)
A specialized system software package used to describe a database and maintain the relationships among the data elements in the database.

data terminal equipment (DTE)
The functional unit of a data station that establishes, maintains, and releases a connection and provides code and signal conversion between the data station and the transmission line. A DTE can serve as a data source, a data sink, or both and can provide for the data communications control functions to be performed in accordance with link protocol. Packet-mode DTEs divide the data into packets. Non-packet-mode DTEs require packet assemblers/disassemblers (PADs) to operate in an X.25 MCS environment.

data transfer unit (DTU)
A hardware unit used on an A 16 or A 17 system that provides support for the disk cache feature by transferring data between disk cache pages in memory and the user data buffers.

DCC
See data communications controller.

disk resource control (DRC) system
An optional feature of the disk subsystem that provides the ability to control disk space on a per user basis. The DRC system does not support interchange (IC) packs or installation-allocated disk (IAD) usage. DRC is not a security system, but normal security checking occurs.

distributed systems service (DSS)
One of a collection of services provided on Unisys hosts to support communications across multihost networks. DSSs can be services such as file handling, station transfer, and mail transfer.

DLP
See data link processor.

DMSII
See Data Management System II.

DRC
See disk resource control system.

DSS
See distributed systems service.

DTE
See data terminal equipment.

DTU
See data transfer unit.

E

EMS
See Entry and Medium Systems.

end of job (EOJ)
The termination of processing of a job.

Entry and Medium Systems (EMS)
A designation referring to the Micro A and A 1 through A 10 systems.

EOJ
See end of job.

F

family name
The name, consisting of up to 17 alphanumeric characters, assigned by an installation to identify a family of disks.

FIB
See file information block.

file information block (FIB)
A data structure in an object code file that contains information describing a file.

file name
A unique identifier for a file, consisting of name constants separated by slashes. Each name constant consists of letters, digits, and selected special characters. A file name can be optionally preceded by an asterisk (*) or usercode, and optionally followed by ON and a family name.

file title
The complete identifier for a file that consists of the file name, the word ON, and the family name.

File Transfer, Access, and Management (FTAM)

The standard developed by the International Standards Organization (ISO) for file exchange and management across an Open Systems Interconnection (OSI) network. FTAM systems can access file attributes (for example, password information) and the contents of files (including individual records, as well as entire files). *See also* OSI File Transfer, Access, and Management.

FORTRAN

Formula Translation. A high-level, structured programming language intended primarily for scientific use.

FTAM

See File Transfer, Access, and Management.

H

halt/load

A system-initialization procedure that temporarily halts the system and loads the master control program (MCP) from a disk to main memory.

HDU

See host data unit.

host data unit (HDU)

The A 12 and A 15 system host interface to the I/O subsystem. An HDU is configured with up to three host-dependent ports (HDPs), each of which supports two message level interface (MLI) cables.

I

I/O

Input/output. An operation in which the system reads data from or writes data to a file on a peripheral device such as a disk drive.

I/O control block (IOCB)

A data structure used for communication between the host system and the I/O subsystem.

I/O module (IOM)

A functional module for connecting a system to a peripheral device. The I/O module is the interface between the processor memory and the I/O subsystem.

I/O processor (IOP)

A specialized processor for moving data between system memory and the I/O subsystem.

InfoGuard

The Unisys security-enhancement software for A Series systems. InfoGuard provides such features as password management, selective logging and auditing, tape volume security, and simplified system-security configuration.

initialization, verification, relocation (IVR)

A maintenance procedure used to write sector boundaries and a blank label on a disk. You can use the IVR procedure to make a new disk pack usable by the system or to make a damaged disk reusable by eliminating defective sectors. The end product of an IVR procedure is a master available table (MAT) of available disk segments with all old files on the disk erased.

intelligent peripheral interface (IPI)

An industry-recognized standard for the specification of the interface between a computer and a peripheral device.

IOCB

See I/O control block.

IOM

See I/O module.

IOP

See I/O processor.

IPI

See intelligent peripheral interface.

IVR

See initialization, verification, relocation.

J

job

An independent process. The job of a particular task is the independent process that is the eldest ancestor of that task.

L

label

(1) The first 28 sectors on a disk, on which information about the disk is stored. This information includes the family name and serial number, the master available table (MAT), the family index number, information about the family base pack, and a pointer to the system directory if the disk contains a directory. (2) An area on a magnetic tape (MT) that contains permanent attributes associated with the tape volume or with individual files on the volume, such as the volume serial number and the file name.

line support processor (LSP)

The data communications subsystem processor that manages communication with the host and initiates processes that control the input of messages to and the output of messages from data communications lines.

LSP

See line support processor.

M

MARC

See Menu-Assisted Resource Control.

mark stack control word (MSCW)

A special control word used by the central processor unit (CPU) to define environments and stacks. The MSCW is generated when compilers issue the mark stack (MKST) command to call subroutines or procedures.

master available table (MAT)

A table stored on each disk that lists the valid sectors on the disk that were successfully processed by the initialization, verification, relocation (IVR) procedure. Pointers to defective sectors are deleted from the MAT so that these sectors cannot be accessed. Normally, the MAT shows the entire disk as being available, minus any defective sectors.

master control program (MCP)

The central program of the A Series operating system. The term applies to any master control program that Unisys might release for A Series systems.

Master Control Program/Advanced Systems (MCP/AS)

The version of the master control program (MCP) that supports the Actual Segment Descriptor (ASD) memory architecture. As of the Mark 3.9 release, MCP/AS is the only master control program available on A Series.

MAT

See master available table.

MCP

See master control program.

MCP/AS

See Master Control Program/Advanced Systems.

MCS

See message control system.

memory interface unit (MIU)

A device that enables a user to interface with memory.

memory storage unit (MSU)

The smallest unit of memory that can be readied or saved on an A Series system. An MSU contains 1 million to 4 million words, depending on the type of MSU. MSU types cannot be mixed on a system.

Menu-Assisted Resource Control (MARC)

A menu-driven interface to A Series systems that also enables direct entry of commands.

message control system (MCS)
A program that controls the flow of messages between terminals, application programs, and the operating system. MCS functions can include message routing, access control, audit and recovery, system management, and message formatting.

Message Handling System (MHS)
A distributed systems service (DSS) that implements the Consultative Committee on International Telegraphy and Telephony (CCITT) X.400 standards for store-and-forward message handling systems.

message level interface (MLI)
The interface between the host system, the I/O subsystem, and the data communications subsystem.

message level interface processor (MLIP)
See I/O processor, Entry and Medium Systems.

MHS
See Message Handling System.

MIU
See memory interface unit.

MLI
See message level interface.

MLIP
An acronym for the obsolete term message level interface processor; *see* I/O processor, Entry and Medium Systems.

mom descriptor
The original descriptor for a data segment. For a given data segment in memory, there is one mom descriptor, but there can be many copy descriptors. A mom descriptor is a data descriptor that has 0 (zero) in the copy bit.

MSCW
See mark stack control word.

MSU
See memory storage unit.

N

network-independent software (NIS)
System software that supports network administration functions common to most types of multihost data communications networks. These functions include network selection, initiation, termination, and command processing.

network information file II (NIFII)
The file generated when a Network Definition Language II (NDLII) program is compiled. This file contains line support processor (LSP) and network support

processor (NSP) code, data structures, and other information. A NIFII is also generally referred to as a network information file (NIF).

network support processor (NSP)
A data communications subsystem processor that controls the interface between a host system and the data communications peripherals. The NSP executes the code generated by the Network Definition Language II (NDLII) compiler for line control and editor procedures. An NSP can also control line support processors (LSPs).

NIFII
See network information file II.

NIS
See network-independent software.

NSP
See network support processor.

O

ODT
See operator display terminal.

op-code
An instruction that is executable by the machine. Op-codes are either created by a compiler and stored in an object code file or created at run time by an interpreter.

Open Systems Interconnection (OSI)
A set of data communications standards defined by the International Organization for Standardization (ISO) that provide for communications between different types of computer systems. The application services defined under OSI include File Transfer, Access, and Management (FTAM) and the Message Handling System (MHS).

operator display terminal (ODT)
A terminal or other device that is connected to the system in such a way that it can communicate directly with the operating system. The ODT allows operations personnel to accomplish system operations functions through either of two operating modes: system command mode or data comm mode.

OSI
See Open Systems Interconnection.

OSI File Transfer, Access, and Management (OSI FTAM)
An A Series distributed systems service (DSS) that supports Open Systems Interconnection (OSI) standards for file management functions such as reading, writing, and copying files on remote hosts.

OSI FTAM
See OSI File Transfer, Access, and Management.

P

PCD

See peripheral configuration diagram.

peripheral

A device used for input, output, or file storage. Examples are magnetic tape drives, disk drives, printers, or operator display terminals (ODTs).

peripheral configuration diagram (PCD)

A list of units on a system, coupled with a diagram that shows how the units are connected to the system.

peripheral test driver (PTD)

A module of the master control program (MCP) that executes maintenance tests for peripheral devices.

PIB

See process information block.

PL/I

Programming Language I. A high-level, structured programming language designed primarily for scientific and commercial use.

process

The execution of a program or of a procedure that was initiated. The process has its own process stack and process information block (PIB). It also has a code segment dictionary, which can be shared with other processes that are executions of the same program or procedure.

process information block (PIB)

A memory structure that is associated with each process stack and code segment dictionary. The PIB contains control information that is visible only to the operating system. The PIB for a process stack also contains a reference to a task attribute block (TAB).

programmable read-only memory (PROM)

A type of memory that can be modified once for specific purposes, and then can only be read.

PROM

See programmable read-only memory.

PTD

See peripheral test driver.

R

RAM

See random-access memory.

random-access memory (RAM)

A type of memory that allows the reading and writing of a memory cell without regard to the location of the preceding read or write operation on the memory. An important characteristic of RAM is that it is volatile; that is, the data stored in it remains there only as long as the computer is not turned off or rebooted.

remote supervisory console (RSC)

A terminal released to remote job entry (RJE) that provides a remote operator and system interface through input and output messages.

remote support center (RSC)

A facility such as a customer service center that provides remote diagnostic capability for hardware and software problems.

resource management module (RMM)

A hardware module that interfaces with the I/O subsystem and schedules tasks on the E-mode processor (EMP) by way of a message protocol.

RMM

See resource management module.

RSC

See remote supervisory console, remote support center.

S

s-code file

In the peripheral test driver (PTD), a file storing semicompiled instructions called s-ops.

s-op

In the peripheral test driver (PTD), a semicompiled instruction that cannot be directly executed by the machine. S-ops are stored in s-code files and are translated into op-codes at run time by the PTD interpreter.

SCSI

See small-computer system interface.

SCSI maintenance bus (SMB)

Proprietary SCSI protocol used to communicate between the maintenance processor and the mainframe hardware.

SCX

See system console transport.

small-computer system interface (SCSI)

An interface adopted by the computer industry as a standard that allows the connection of low-cost peripherals to computer systems.

SMB

See SCSI maintenance bus.

stability log

A file on A 12, A 15, A 16, and A 17 systems that is dedicated solely to storing stability information about the system, such as processing interruptions, halt/loads, and other software or hardware problems encountered during daily processing. The stability log is separate from the system summary log (SUMLOG).

system console transport (SCX)

On the A 11, A 16, and A 19 systems, a device that enables the operating system, in cooperation with the system console, to send the stability log from an A series machine to the Unisys remote support center by means of an automatic transport service.

T

TAB

See task attribute block.

task

(1) A dependent process. (2) Any process, whether dependent or independent. *See also* process.

task attribute block (TAB)

A memory structure that stores the values of task attributes associated with a given task variable. Before the Mark 3.9 release, this information was part of the process information block (PIB).

task control processor (TCP)

A special purpose processor that schedules tasks on the E-mode processor (EMP) by way of a message protocol.

tasking program

A program that has been marked with tasking status by the MP (Mark Program) system command.

tasking status

A type of security status that permits a program to perform most of the actions that normally require message control system (MCS) privileges. A process receives tasking status if it is running without a usercode and is executing code from a tasking program.

TCP

See task control processor.

U

usercode

An identification code used to establish user identity and control security, and to provide for segregation of files. Usercodes can be applied to every task, job, session, and file on the system. A valid usercode is identified by an entry in the USERDATAFILE.

W

WFL

See Work Flow Language.

Work Flow Language (WFL)

A Unisys language used for constructing jobs that compile or run programs on A Series systems. WFL includes variables, expressions, and flow-of-control statements that offer the programmer a wide range of capabilities with regard to task control.

Bibliography

A Series ALGOL Programming Reference Manual, Volume 1: Basic Implementation (8600 0098). Unisys Corporation.

A Series CANDE Operations Reference Manual (8600 1500). Unisys Corporation.

A Series DCALGOL Programming Reference Manual (8600 0841). Unisys Corporation.

A Series DiagnosticMCS Reference Manual (1169596). Unisys Corporation.

A Series File Attributes Programming Reference Manual (8600 0064). Unisys Corporation.

A Series GETSTATUS/SETSTATUS Programming Reference Manual (8600 0346). Unisys Corporation.

A Series I/O Subsystem Programming Guide (8600 0056). Unisys Corporation.

A Series Print System (PrintS/ReprintS) Administration, Operations, and Programming Guide (8600 1039). Unisys Corporation.

A Series Security Administration Guide (form 8600 0973). Unisys Corporation.

A Series Software Release Installation Guide (8600 0981). Unisys Corporation.

A Series System Administration Guide (8600 0437). Unisys Corporation.

A Series System Commands Operations Reference Manual (8600 0395). Unisys Corporation.

A Series System Management Facility II (SMFII) Query Operations Guide (7831 1867). Unisys Corporation.

A Series System Management Facility II (SMFII) Resource Management Operations Reference Manual (7831 1628). Unisys Corporation.

A Series System Operations Guide (8600 0387). Unisys Corporation.

A Series System Software Utilities Operations Reference Manual (8600 0460). Unisys Corporation.

A Series SYSTEMSTATUS Programming Reference Manual (8600 0452). Unisys Corporation.

A Series Task Attributes Programming Reference Manual (8600 0502). Unisys Corporation.

A Series Task Management Programming Guide (8600 0494). Unisys Corporation.

A Series Work Flow Language (WFL) Programming Reference Manual (8600 1047). Unisys Corporation.

A 16/A 19 System Console Operations Guide (3734 9339). Unisys Corporation.

BNA Version 2 Network Encoded Messages Programming Reference Manual,Volume 2 (3787 7529), (3787 7598), and (3787 5127). Unisys Corporation.

Index

A

B

C

D

E

F

G

J

K

L

M

N

Q

R

S

T

UNISYS

Help Us To Help You

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

O Addition O Deletion O Revision O Error

Publication title

Document number Date

Comments:

Name Telephone number ( )

Title Company

Address

City State ZIP code

Cut along dotted line

Tape

Do Not Staple

Please Fold and Fasten

Tape

Fold here

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 817 DETROIT, MI 48232

Postage Will Be Paid By Addressee

Hardware Documentation
Unisys Corporation
25725 Jeronimo Road
Mission Viejo, CA 92691-9826 USA