Gould MPX-32™

Release 3.3

Reference Manual

Volume II

Utilities and Processors

December 1986

Publication Order Number: 323-001552-300

GOULD
Electronics

Gould Inc., Computer Systems Division
6901 West Sunrise Boulevard
Fort Lauderdale, FL 33313

HISTORY

The MPX-32 Release 3.0 Reference Manual, Publication Order Number **323-001550-000,** was printed June, 1982.

Publication Order Number **323-001552-100** (Revision 1, Release 3.2) was printed June, 1983.

Publication Order Number **323-001552-200** (Revision 2, Release 3.2B) was printed March, 1985.

Publication Order Number **323-001552-201** (Change 1 to Revision 2, Release 3.2C) was printed December, 1985.

Publication Order Number **323-001552-300** (Revision 3, Release 3.3) was printed December, 1986.

Note: For Release 3.3 of MPX-32, the reference material formerly included at the back of each MPX-32 Reference Manual volume was placed in a separate document and assigned Publication Order Number 323-001550-300.

The updated manual contains the following pages:

Title page
Copyright page
iii/iv through xiv
1-1 through 1-91/1-92
2-1 through 2-61/2-62
3-1 through 3-63/3-64
4-1 through 4-4
5-1 through 5-8
6-1 through 6-5/6-6
7-1 through 7-11/7-12

CONTENTS

CHAPTER 2 OPERATOR COMMUNICATIONS (OPCOM)

CHAPTER 3 VOLUME MANAGER (VOLMGR)

CHAPTER 4 COMPRESS

FIGURES

TABLES

Documentation Conventions

Notation conventions used in directive syntax and message examples throughout this manual are described below.

lowercase letters

In directive syntax, lowercase letters identify a generic element that must be replaced with a value. For example,

```
!ACTIVATE taskname
```

means replace taskname with the name of a task. For example,

```
!ACTIVATE DOCCONV
```

In messages, lowercase letters identify a variable element. For example,

```
**BREAK** ON:taskname
```

means a break occurred on the specified task.

UPPERCASE LETTERS

In directive syntax, uppercase letters specify a keyword must be entered as shown for input, and is printed as shown in output. For example,

```
SAVE filename
```

means enter SAVE followed by a filename. For example,

```
SAVE DOCCONV
```

In messages, uppercase letters specify status or information. For example,

```
taskname,taskno ABORTED

*YOUR TASK IS IN HOLD. ENTER CONTINUE TO RESUME IT
```

Braces { }

Elements placed one under the other inside braces specify a required choice. You must enter one of the arguments from the specified group. For example,

```
{counter  }
{startbyte}
```

means enter the value for either counter or startbyte.

Brackets [ ]

An element inside brackets is optional. For example,

```
[CURR]
```

means the term CURR is optional.

Items placed one under the other within brackets specify one of the group of options must be entered or none at all. For example,

```
[base name  ]
[progname   ]
```

means enter the base name or the program name or neither.

Items in brackets within encompassing brackets specify one item is required only when the other item is used. For example,

```
TRACE [lower address [upper address] ]
```

means both the lower address and the upper address are optional, and the lower address can be used alone. However, if the upper address is used, the lower address must also be used.

Commas between multiple brackets within an encompassing set of brackets are semi-optional; that is, they are not required unless subsequent elements are selected. For example,

```
M.DFCB fcb,LFC[,[a],[b],[c],[d],[e]]
```

could be coded as

```
M.DFCB FCB12,IN
```

or

```
M.DFCB FCB12,IN,,ERRAD
```

or

```
M.DFCB FCB13,OUT,,ERAD,,PCK
```

Horizontal Ellipsis . . .

The horizontal ellipsis indicates the previous element can be repeated. For example,

```
name [,name] . . .
```

means one or more names separated by commas can be entered.

Vertical Ellipsis

.
.
.

The vertical ellipsis specifies directives, parameters, or instructions have been omitted. For example,

```
COLLECT 1
  .
  .
  .
LIST
```

means one or more directives have been omitted between the COLLECT and LIST commands.

Numbers and Special Characters

In a syntax statement, any number, symbol, or special character must be entered as shown. For example,

(value)

means enter the proper value enclosed in parentheses; e.g., (234).

Underscore

In syntax statements, underscoring specifies the letters, numbers or characters that are typed by the user as an abbreviation. For example,

ACTIVATE taskname

means spell out the directive verb ACTIVATE or abbreviate it to ACTI.

RESET

means type either RESET or RST.

In examples, all terminal input is underscored; terminal output is not. For example,

TSM > EDIT

means TSM > was written to the terminal; EDIT is typed by the user.

Subscript Delta $_{\Delta}$

A subscript delta specifies a required space. For example,

EDT > STO$_{\Delta}$TSSPGM

means a space is required between O and T.

CHAPTER 1

JOB CONTROL LANGUAGE (JCL)

1.1 Introduction

Job Control Language (JCL) provides an interface to the facilities of the MPX-32 operating system. It is used in interactive, batch, and real-time processing environments.

JCL directives are used to:

- logon to MPX-32
- access any MPX-32 processor
- request assignment of MPX-32 resources
- specify and pass parameters to interactive and batch tasks
- automate a series of tasks into a job, or submit a stream of jobs
- specify conditional alternative actions
- communicate with on-line users or the operator
- account for the use of computer resources
- logoff of MPX-32

The Terminal Services Manager (TSM) provides interactive, time-shared access to the MPX-32 system for terminals connected through IOP, ALIM, or ACM controllers.

Many TSM directives are used for job control to accomplish the same functions as job statements used when submitting a job for batch processing. Other TSM directives activate and run tasks on-line and send messages to terminals.

Terminals must be initialized before they can be used with the MPX-32 system. If a terminal hardware interface has been disrupted, e.g., the terminal has been unplugged, it must be reinitialized.

1.2 JCL Directive Summary

The following sections summarize JCL directives by function. Each directive is described in detail in this chapter. Most JCL directives can be abbreviated. Underlining indicates accepted directive abbreviations.

Unless otherwise indicated, JCL directives are valid in the batch and interactive environments, including directive files.

1.2.1 Job Specification Directives

Job specification directives identify the beginning and end of a single job. Job specification directives are:

$JOB	Identifies the beginning of a job
$EOJ	Identifies the end of a job

One or more tasks can be grouped together in a single job. Each job is assigned a unique job number when it is initiated. The job number provides identification and control for certain aspects of job execution and spooled output.

The $JOB directive is required in batch mode and is optional in interactive mode.

1.2.2 Resource Specification Directives

Resource specification directives specify resource requirements for a task. Some are effective only for the duration of a task; others are effective until they are explicitly overridden. See individual directive descriptions. Resource specification directives are:

$ALLOCATE	Overrides cataloged memory allocation for on-line tasks
$ASSIGN	Supplies default assignments for logical file codes (LFCs) used by the task being executed/activated
$ASSIGN1	Associates a permanent disc file (optionally unblocked) with an LFC
$ASSIGN2	Associates a system SBO, SLO, SYC, or SGO file with an LFC. In the interactive mode, SYC is automatically associated with the user's terminal.
$ASSIGN3	Associates a device with an LFC. If channel/subaddress are specified, denotes specific device. User's terminal is preassigned as "UT=terminal". To assign the terminal for other LFCs, use ASSIGN4.
$ASSIGN4	Associates an LFC with another LFC. All LFCs referring to the user's terminal should be assigned as UT, such as SI=UT.
$CHANGE	Establishes new default working directory and/or project group name
$CLEAR	Clears previous $ASSIGN, $OPTION, $ALLOCATE, and $EXTDMPX directives. Also terminates processing from a directive file before end-of-file.
$CREATE	Creates a permanent file
$DELETE	Deletes a permanent file
$DISMOUNT	Informs the operator to remove a volume or an unformatted medium from a device
$EXTDMPX	Dynamically overrides the SYSGEN and Cataloger assignment for the logical starting address of extended MPX-32.
$MOUNT	Informs the operator to mount a volume on a device
$OPTION	Provides options for tasks. Options 1 to 20 are task-dependent; options 21 to 32 are system-defined and available to all tasks.
$RENAME	Changes the file name of a permanent file
$USERNAME	Modifies the current directory name for file access

1.2.3 Task Activation and Directive File Specification Directives

These directives activate tasks and select directive files for execution. Task activation and directive file specification directives are:

$ACTIVATE	Activates an independent or real-time task
$DEBUG	Loads the specified task with the interactive debugger attached and passes control to the debugger
$EXECUTE	Activates a task in the interactive or batch environment by specifying the pathname of a cataloged load module. The task is assumed to be located in the system volume and directory.
$RUN	Activates a task in the interactive or batch environment by specifying the pathname of a cataloged load module
$SELECT	Selects a file of command processor directives

1.2.4 Terminal and Batch Control Directives

Terminal and batch control directives specify terminal and job environment conditions. They are valid in the interactive mode only. Terminal and batch control directives are:

$BATCH	Submits a directive file to run in the batch stream
$CONTINUE	Resumes execution of a task that was placed in a hold state
$EXIT	Logs a user off the system
$LINESIZE	Modifies the screen width defined for a terminal
$PAGESIZE	Specifies the number of consecutive lines to display at the terminal without pressing the carriage return (CR)
$REMOVE	Terminates processing of a specified job
$SUBMIT	Submits a directive file for batch processing
$SYSOUT	Specifies how SLO assignments are interpreted (valid in interactive mode only)
$URGENT	Changes the priority of a waiting or active batch job
$WAIT	Puts a terminal in a wait state for receiving messages

1.2.5 Status and Inquiry Directives

Status and inquiry directives receive or transmit general information regarding system usage. Status and inquiry directives are:

$ACCOUNT	Displays the contents of the job accounting file to an SLO file or a terminal
$ERR	Displays the description of a valid abort code
$LIST	Displays the contents of a file to an SLO file or a terminal
$NOTE	Echoes comments to the system console or user's terminal
$PRINT	Submits a file to the output spooler
$SHOW	Displays CPU execution time for tasks activated by an owner, batch jobs waiting or active in the system, or terminal addresses of all logged on terminal users
$SIGNAL	Sends a message to another logged on user or to all terminals
$WHO	Displays all logged on terminal users.

1.2.6 Macro and Conditional Directives

Macro and conditional directives are used in macro files. They are valid in interactive and batch directive files. Directives used between $DEFM and $ENDM can be specified in any order. Macro and conditional directives are:

$DEFM	Defines parameters for a directive file
$DEFNAME	Establishes a name to branch to for conditional processing
%name	Establishes a name to branch to for conditional processing
$GOTO	Branches to a name in a directive file
$IFA	Branches to a name if the specified parameter is absent
$IFP	Branches to a name if the specified parameter is present
$IFF	Branches to a name if a condition is false
$IFT	Branches to a name if a condition is true
$RESETF	Sets flag(s) to false for conditional processing
$SET	Associates a new value with a dummy argument
$SETF	Sets flag(s) to true for conditional processing
$ENDM	Defines the end of a directive file (optional)

1.2.7 Batch Input Spooling Directives

Batch and input spooling directives are used by the input spoolers to process source files before submission to the batch stream. They are processed before any other TSM directives and are valid in batch directive files only. Batch input spooling directives are:

$OBJECT	Precedes program object records
$SELECTD	Spools batch records to the SYC file from the specified peripheral device
$SELECTF	Spools batch records to the SYC file from the specified permanent disc file
$SELECTLD	Spools batch records to the SYC file from the specified peripheral device. Data on the device is in library format.
$SELECTLF	Spools batch records to the SYC file from the specified permanent disc file. Data in the file is in library format.
$SELECTS	Resets alternate system input source levels established by previous $SELECTx statements
$$	Terminates batch input stream
$$$	Terminates batch input stream when continuous batch mode was requested

1.3 Logging On to TSM

To logon to MPX-32 from a terminal, press the wake-up character defined in the LOGONFLE for the terminal (like CNTRL E or ?). TSM responds:

```
MPX-32 RELEASE x.x patch iname TERMINAL SERVICES MANAGER
(C) COPYRIGHT 1983, GOULD INC., CSD, ALL RIGHTS RESERVED.
ENTER YOUR OWNERNAME:
```

x.x	is the release number of this version of MPX-32
patch	is the patch update level or blank
iname	is the system image name defined at SYSGEN

Respond by entering a one- to eight-character owner name. The following characters cannot be used in owner names: blanks, commas, semicolons, equal signs, line feeds, dollar signs, exclamation points, percent signs, and left or right parentheses. All other characters are valid.

If an M.KEY file containing valid owner names and owner keys was established (see MPX-32 Reference Manual, Volume III, Chapter 10), one of the following prompts is displayed:

Local Echoplex Terminals:

```
ENTER KEY:
################                Key characters echoed
```

Host/Controller Echoplex Terminals with IOQs linked to the UDT:

ENTER KEY:	Key characters not echoed

If a key was established in the M.KEY file, it must be entered to gain access to the system. To change a key, enter the old key followed by a comma and the new key. To delete a key, enter the key followed by a comma and a carriage return. To establish a key if one does not exist, enter a comma followed by the desired key. The following characters cannot be used in owner keys: blanks, commas, semicolons, equal signs, line feeds, dollar signs, exclamation points, percent signs, and left or right parentheses. All other characters are valid. If a key does not exist and one is not to be established, enter a carriage return.

If an M.KEY file does not exist, the prompt is not displayed.

If the owner name and/or key is not valid, or if another user is logged on with that owner name, TSM displays one of the following messages:

```
UNAUTHORIZED NAME OR KEY
    or
NAME IN USE
```

The TRY AGAIN: prompt is issued until a valid owner name and key are entered. Press the carriage return to logoff the terminal.

If the owner name and key are valid, TSM checks to see if a new key was entered. If so, the new key is placed in the M.KEY file. It must be entered at the next logon session to gain access to the system. Next, the owner's logon environment is defined according to the parameters specified in the M.KEY file. This environment includes access restrictions to system services, and defaults for tab settings, project group, and the current working directory.

The project group is used for accounting purposes and to establish access restrictions for resources. The project group is changed by the $CHANGE PROJECT directive. When a project group is changed, any nonzero CPU and IPU execution time for the previous session is recorded in the M.ACCNT file under the old project group. The working directory specifies the pathname to be prefixed for file name specification. If a default directory is not established at logon, (i.e., the volume is not mounted), the user's working directory is set to @SYSTEM(SYSTEM). The working directory is changed by the $CHANGE DIRECTORY directive. However, if a nonpublic volume is specified, a $MOUNT directive must be issued before issuing the $CHANGE DIRECTORY directive.

1.4 Accessing Batch, Interactive, and Real-time Processing Environments

TSM provides access to the batch, interactive, and real-time (independent) processing environments. A task is put into execution in any of the three processing environments after logging on to TSM.

A task can be activated as part of a batch job by creating a file containing JCL statements and directives to other processors and submitting it with a TSM, OPCOM, or Text Editor (EDIT) BATCH directive:

```
TSM>    $BATCH jobfile,par1,par2,...par8

TSM>    $OPCOM
??      BATCH jobfile

TSM>    EDIT filecode
EDT>    BATCH jobfile
```

When a batch job is submitted from a terminal, messages pertaining to the job are displayed at the submitting terminal if the user remains logged on. If the user logs off, the messages are displayed at the system console. An end-of-job (EOJ) message is displayed at the submitting terminal when the job is complete:

```
jobno owner  $EOJ jobname
```

A task is activated in the interactive environment by supplying the task name in response to the TSM prompt:

```
TSM>    MYTASK
```

In the interactive mode, the task executes at the time-distribution priority of TSM and has special TSM support for breaks, wakeup characters, terminal assignments, and other functions as described in section 1.8.

A task is activated in the independent or real-time environment through the Operator Communications (OPCOM) service by using the OPCOM ACTIVATE or ESTABLISH directives. A task activated in the real-time environment executes at its base priority. TSM recognizes an exclamation point as a synonym for OPCOM. An exclamation point can enter an OPCOM directive with an automatic return to TSM. The following examples activate a task named MYTASK from OPCOM for independent processing in the real-time environment:

```
TSM>    $OPCOM                          TSM> !ACTIVATE MYTASK
??      ACTIVATE MYTASK    or
??      EXIT                            TSM>
TSM>
```

A task is also activated at its base priority in the real-time environment by the TSM $ACTIVATE statement. In batch mode, assignment and option statements can precede the $ACTIVATE statement so that the load module need not be created with all static resource requirements.

Figure 1-1 illustrates the types of processing environments used to run tasks and jobs.

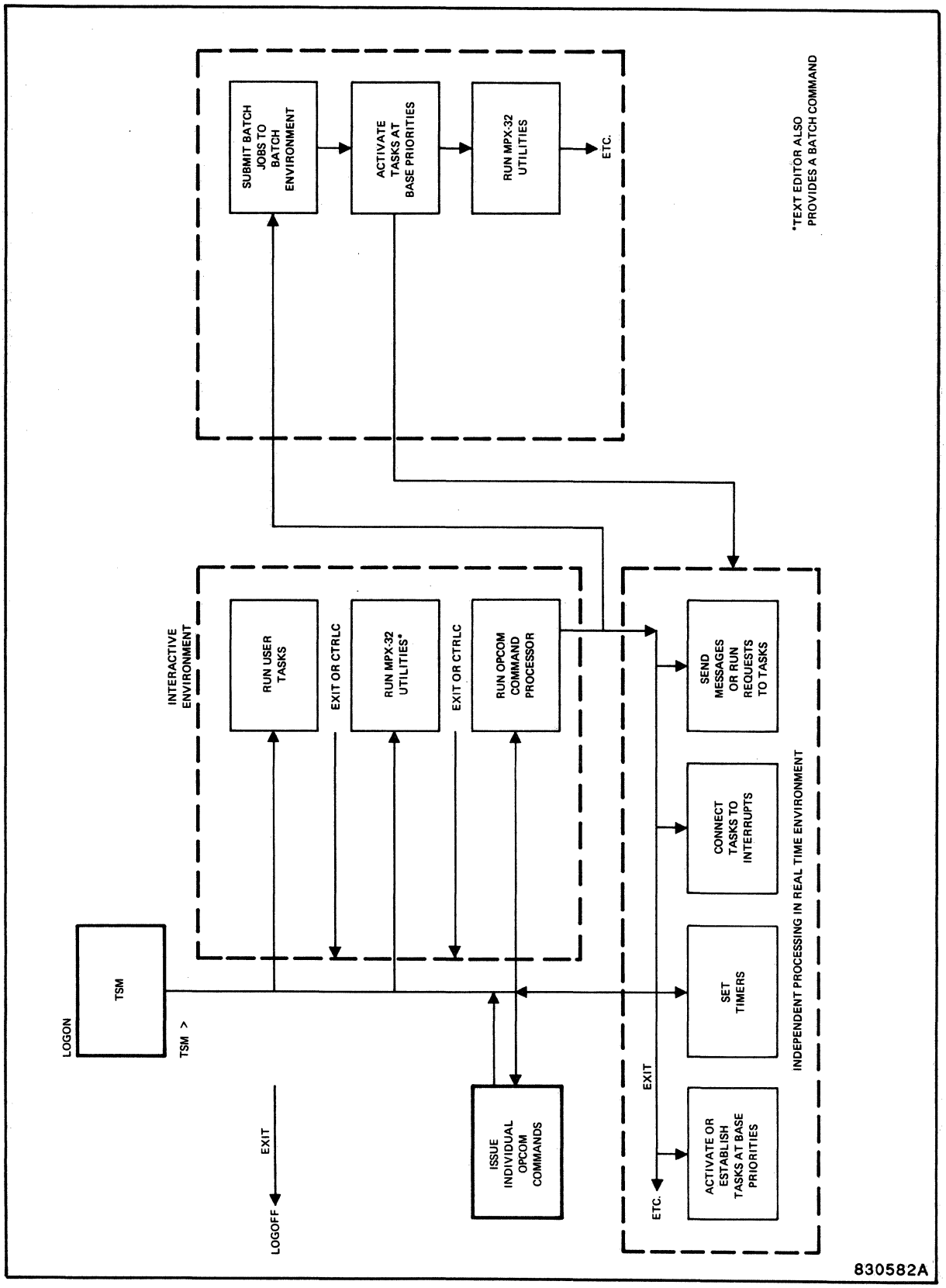


Figure 1-1. Interactive/Batch/Real-time Environments

1.5 Logging Off

The TSM prompt must be displayed to logoff the terminal. If you are in another processor or utility, specify the appropriate exit directive for the processor or use the CNTRL C key sequence to return to the TSM prompt.

To logoff, specify the TSM $EXIT directive in response to the TSM prompt. The following message is displayed:

```
CPU EXECUTION TIME    = xx HOURS-  xx MINUTES-   xx.xx SECONDS
TOTAL CONNECT TIME    = xx HOURS-  xx MINTUES-   xx.xx SECONDS

RING IN FOR SERVICE
```

xx is a decimal number indicating units of CPU execution and connect time.

When the Internal Processing Unit (IPU) and its interval timer handler are specified during SYSGEN and the IPU is used for task execution, the following message is also displayed after logging off a terminal:

```
IPU EXECUTION TIME    = xx HOURS-  xx MINUTES-   xx.xx SECONDS
```

1.6 Communicating with Other Terminals

The TSM $SIGNAL directive sends a message (80 characters maximum) to a specific terminal user or to all terminals. When a message is entered, it is identified with the owner name of the sender and the date and time sent. A message cannot be sent to an owner who is not currently logged on. If a message is sent to all terminals, it is queued for output to any terminal not currently logged on and displayed the next time someone logs on to that terminal. Only the last two messages sent are queued if a user is not logged on.

To communicate with a particular task, use the OPCOM SEND directive. SEND transmits control messages or data to tasks with message receivers.

1.7 Special Keys

The control key (CNTRL) can be used with alphabetic keys in the interactive mode to perform special functions. Table 1-1 indicates keys used to correct characters, correct a line of terminal input, and perform other interactive functions.

Table 1-1
Special Keys

Key	Description
CARRIAGE RETURN <CR>	Terminates the current input or directive line. Also continues output listing after a pause at the bottom of the terminal screen. Pressing the carriage return is assumed in all cases at the end of an input or directive line, and is not discussed in the documentation.
CNTRL H	Deletes the character just typed. Repeat to delete multiple characters. Enter the correct character(s). The backspace key produces the same result.
Rub or Delete	Deletes the line just typed. Must be used before a carriage return. The cursor moves to the next line. Enter the corrected line.
CNTRL I	Displays tab spacing on input. The tab key produces the same result.
CNTRL C	Exits a processor when an EXIT directive is not available. Shown in text and examples as CNTRL C. Simulates EOF function from terminals.
Break	Terminates I/O.
Wakeup	Re-establishes communication with a task or TSM.

1.8 Executing Tasks Under TSM

Any cataloged load module runs interactively by issuing the TSM $RUN directive. Tasks that are designed to run in any environment (on-line, independent real-time, or batch) can run from terminals.

Files and devices can be assigned and options selected for a task to run interactively. Assignments and options apply only to one task. After the task is run, they are cleared automatically by TSM.

The $CLEAR directive clears assignments and options before a task is run. $CLEAR also performs the function of terminating input from a directive file.

1.8.1 System Control File

A set of job control directives stored in a file is a directive file. When a user invokes a directive file, the file is referred to as the System Control file (SYC) for that user. For read operations, the SYC is a blocked file, with a record length of 80 bytes. Columns 73 through 80 are not interpreted by the command processor and frequently contain sequence numbers. The SYC can contain both JCL directives and directives to individual processors. To distinguish the JCL directives from other data, a dollar sign ($) should precede all JCL directives (although this restriction is sometimes relaxed, it is nevertheless recommended). The general syntax of JCL directives is as follows:

```
$keyword [arguments...]
```

In most cases, the keyword can be abbreviated to four characters. The arguments can be separated by blanks, commas, equal signs, semicolons, or parentheses. Where argument ordering is implicit in the command, the absence of an argument is indicated by an extra comma. For example:

```
$A3 LFC=MT,REEL,,UNBLOCKED
```

Write operations on the SYC file are directed to the UT file which in the interactive mode is the terminal. Write operations on the SYC file cannot be performed in the batch mode.

Close and open operations on the SYC file are handled as described in Section 1.11.3. Rewind operations on the SYC file are handled as described in Section 1.11.4.

1.8.2 Options

1.8.2.1 Prompting Option

If a task running on-line is designed primarily for batch processing, the task may not establish a prompt for reads from the terminal. The TSM directive $OPTION PROMPT or $OPTION 21 can have TSM automatically precede any read from the terminal with the first three characters of the task name and a right angle bracket. $OPTION PROMPT should be used only if the task does not write the prompt itself.

1.8.2.2 Lower Case Option

When running a task (e.g., the Volume Manager utility) on-line, the task can force a translation of lower case characters to upper case. This translation is inhibited by the TSM directive $OPTION LOWER or $OPTION 22 to allow entering lower case characters. The only strings that must be entered in all upper case are file names.

1.8.2.3 Command/Nocommand Option

When JCL directives are read from an SYC file, they are echoed at the user's terminal in the interactive environment or the SLO file in the batch environment. To inhibit this operation, specify $OPTION NOCOMMAND or $OPTION 27. This option remains in effect until an end-of-job ($EOJ) in batch mode or the terminal is logged off. The listing operation can be reset at any time by specifying $OPTION COMMAND at the TSM prompt.

1.8.2.4 Text Option

When text is read from an SYC file, it can be echoed at the user's terminal in the interactive environment or the SLO file in the batch environment by specifying $OPTION TEXT or $OPTION 23. This is a one-shot option that applies only to the next executed task.

1.8.2.5 Retain Option

When activating a task, $OPTION RETAIN or $OPTION 30 forces cataloged options to be ORed with the user supplied activation time options. This overrides the clearing of cataloged options 1 to 20. This option is effective once and applies only to the next executed task.

1.8.2.6 Clear Option

When activating a task, $OPTION CLEAR or $OPTION 31 forces all cataloged options 1 to 20 to clear. Options 21 to 32 are ORed with any options supplied during task activation. This option is effective once and applies only to the next executed task.

If $OPTIONS RETAIN and CLEAR are both specified during task activation, RETAIN is reset and CLEAR remains in effect. RETAIN and CLEAR can only be specified in TSM in the batch or interactive modes. If cataloged, they are ignored.

1.8.2.7 Abort/Noabort Option

The abort option causes abort messages to display to the terminal, console, or SLO device in the interactive, batch or real-time environments. It is the default if noabort or option 29 are not set.

Option noabort inhibits the display of abort messages during a job or interactive session. Any tasks executed or activated while this option is in effect have OPTION 29 set. OPTION 29, which can be specified instead of OPTION NOABORT, inhibits abort messages for the next executed or activated task only.

1.8.2.8 Quiet Option

The quiet option inhibits messages from being displayed asynchronously on a full-duplex terminal. This option has no effect when specified from the console. It is the default unless overridden by option unquiet.

1.8.2.9 Unquiet Option

Option unquiet causes messages to display asynchronously on a full-duplex terminal. It is the default for the console only.

1.8.2.10 Error Option

The error option enables the expansion of abort codes. It is the default unless overridden by option noerror.

1.8.2.11 Noerror Option

The noerror option disables the expansion of abort codes. The numeric equivalent for this option is 28.

1.8.2.12 Wrap Option

The wrap option enables terminal line wrap. It is the default unless overridden by option nowrap. This option is not valid in the batch mode.

1.8.2.13 Nowrap Option

The nowrap option disables terminal line wrap, with the following exceptions:

- the output contains an escape character followed by a control character. This exception may generate a special character that causes a line wrap.
- the output contains embedded carriage return - line feeds, and the data between returns exceeds the terminal line size. This exception generates a line wrap.
- the line size setting is greater than the device line size. This exception may generate a line wrap.

This option is not valid in the batch mode.

1.8.2.14 CPU Only Option

The CPU only option specifies that the task executes only in the CPU. The numeric equivalent for this option is 25. If this option is not specified, the default is to execute the task on the first available processor (CPU or IPU).

1.8.2.15 IPU Bias Option

The IPU bias option specifies that IPU-compatible tasks are to execute on the IPU. If the IPU is not available, the task is executed on the CPU. The numeric equivalent for this option is 26.

1.8.2.16 U/C Option

The U/C option converts all input on a directive line to upper case. It is the default unless overridden by option L/C.

1.8.2.17 L/C Option

The L/C option allows characters to be read as entered on a directive line. It can override a U/C option.

1.8.2.18 Dump Option

The dump option writes the task memory area to the SLO file if an abort occurs. The SLO file is printed on the LOD device. The numeric equivalent for this option is 24.

1.8.3 Breaks

The break key terminates I/O in process. If the break key or wakeup character is pressed and a task does not have its own break receiver, TSM prompts:

```
** BREAK ** ON:taskname  AT:location  CPU TIME = n SEC.
CONTINUE, ABORT, DEBUG, OR HOLD?
```

Enter C (Continue) to continue task execution. If A (Abort) is entered, TSM aborts the task, displays the following message, and returns the TSM prompt:

```
taskname #taskno. ABORT AT:psw-bias mm/dd/yy hh/mm/ss abortcode
TSM>
```

Enter D (Debug) to attach the MPX-32 Debugger to the task. A debug prompt is then displayed. If a nonbase task has a shared CSECT, the debugger is not attached and TSM displays the following message:

```
UNABLE TO ATTACH DEBUGGER - TASK CATALOGED ENVIRONMENT NODEBUG
OR SHARED TASK.
```

To debug a shared CSECT task, request the debugger at task activation.

Enter H (Hold) to place the task in a hold state. The following message and TSM prompt are displayed:

```
*YOUR TASK IS IN HOLD. VALID COMMANDS ARE:
CONTINUE, CLEAR, LINESIZE, PAGESIZE, ERR, SHOW, SETF, RESETF, SIGNAL,
CHANGE, CREATE, DELETE, PRINT, BATCH, SUBMIT, ACTIVATE, URGENT, REMOVE
AND WAIT.
TSM>
```

If the task has its own break receiver, TSM defers to it when the break key is pressed.

1.8.4 Wakeups

If a task is executed and there is no response, it is possible the task is waiting for a resource. To communicate with a task that is waiting for resources (memory, a device, a file, etc.), enter the wakeup character used to logon. TSM responds:

```
TASK state. CONTINUE OR DELETE?
```

TSM displays the type of resource for which the task is queued. To delete the task, enter D (Delete). The task is killed and TSM displays an abort message:

```
taskname,taskno ABORTED. PSW:psw-contents  BIAS:bias REASON: errorcode
TSM>
```

To continue the task, enter C (Continue). The task continues in execution. The TSM prompt is not returned.

If the task does not have its own break receiver, TSM responds with an appropriate message. See Section 1.8.3.

The wakeup character also re-establishes communication with TSM after entering a wait state. See the TSM $WAIT directive.

1.8.5 Tabs

The M.KEY file can contain default tab settings for each logon owner name as described in the MPX-32 Reference Manual Volume III, Chapter 10. If no tabs are set in M.KEY, TSM sets system default tabs. Tab settings (M.KEY or default) can be overridden by using the Text Editor (EDIT) utility SET TABS directive.

The most recent tabs set with the Text Editor SET TABS are effective while the user remains on the system. When exiting TSM, the Text Editor SET TABS are not saved. When the user logs on again, either the M.KEY tabs or the default tabs are set.

During formatted input, the tab character (CNTRL I) is interpreted by the TSM device handlers and replaced by the appropriate number of blanks. The cursor is adjusted by echoing the spaces to the terminal.

1.8.6 Upper and Lowercase Sensitivity

Under normal conditions, J.TSM recognizes upper and lowercase characters while processing directives. If the input is from the terminal and OPTION U/C is set, the terminal or console handler translates all lowercase alphabetical characters to uppercase. If the input is from the terminal and OPTION L/C is set, the terminal or console handler inhibits case translation. If the input is from a command file, case translation is not processed. This prevents alterations of disc data as the data is read.

1.9 TSM Screen Logic

TSM end-of-screen logic is automatically available for an interactive task. TSM uses the SYSGEN-defined screen length for a terminal. During a write, TSM displays the following prompt at the logical bottom of the screen:

```
ENTER CR FOR MORE
```

If a carriage return is entered, TSM writes the next line on the screen and returns to the task. If any character other than a carriage return is entered, TSM throws away the line that caused it to go to the end-of-screen. It sets bit seven of word three in the FCB that indicates end-of-medium before returning to the task. If the task contains the logic to check the situation, the appropriate action can be taken. If the task does not have the logic to check and stop output, TSM continues output to the next bottom-of-screen.

If the screen length is not defined at SYSGEN (implying a hardcopy terminal), the TSM end-of-screen logic is not used.

1.10 Logical File Codes

Logical file codes (LFCs) for input and output are assigned by the $ASSIGN, $ASSIGN1, $ASSIGN2, $ASSIGN3, and $ASSIGN4 directives. The following sections describe LFC assignments.

1.10.1 System Control File (SYC)

When a directive file is invoked, the file is referred to as the System Control (SYC) file. The SYC file may contain both JCL directives and directives to other processors. To

distinguish JCL directives from other data, a dollar sign should precede all JCL directives.

For read operations, the SYC is a blocked file with a record length of 80 bytes. Columns 73 to 80 are not interpreted by command processors because they frequently contain sequence numbers.

1.10.2 User Terminal (UT)

Another LFC used for input is the user terminal (UT). If no directive file is present, TSM assigns logical file code SYC to UT.

As TSM reads a directive file, it displays directives and responses at the terminal as if it were operating interactively. TSM turns control over to a task when it encounters a directive to activate the task. Whether directives and responses to the task are displayed on the terminal depends on if the task uses the UT assignment for I/O.

TSM returns to the terminal for input when all directives are processed, a $CLEAR directive is entered, an $ENDM directive is read, or a task reads from the terminal.

Write operations on the SYC file are directed to the UT file, which is the terminal in the interactive mode. Write operations on the SYC file cannot be performed in the batch mode.

1.10.3 Spooled Output Files

Output for interactive and batch jobs can be directed to three types of system files:

- System General Output (SGO)
- System Listed Output (SLO)
- System Binary Output (SBO)

The print and punch spools (SLO and SGO) are available to all tasks. SGO, and certain options of SLO and SGO, are available only within jobs.

1.10.3.1 System General Output (SGO)

System General Output (SGO) files are used for accumulating object records. A separate SGO file is automatically allocated for each job.

Various assemblers and compilers write object records to SGO. Object records are accumulated until either the entire contents are read or the end of the job is reached. The SGO file is then deleted. Any job-oriented task can write records to or read records from the job's SGO file.

If SGO assignments are issued in the interactive environment, they are redirected to the NULL device.

1.10.3.2 System Listed Output (SLO) and System Binary Output (SBO)

System Listed Output (SLO) files are used for listed output. System Binary Output (SBO) files are used for punched output. Both are temporary, automatically extendible files created by the J.TSM command processor. Access restrictions of SLO and SBO files allow tasks to read from, write to, rewind, or write an end-of-file (EOF) without restriction.

SLO files are blocked files with a maximum record size of 133 bytes. SBO files are blocked with a maximum binary record size of 120 bytes (1.5 bytes per card column), or a maximum ASCII record size of 80 bytes (one byte per card column).

SLO and SBO files are accessed by each job step in the append mode. As the SLO and SBO files pass from one job step to the next, listed and/or binary output is accumulated in a single extendible SLO or SBO file. Extendible files eliminate the need for spooled output file linkage.

An SLO or SBO file can be made permanent by supplying the appropriate parameter(s) on the $JOB statement (SLOF= or SLOD=).

1.11 SYC and Terminal I/O

If there is not a current directive file, the SYC file is identical to the UT file except for maximum transfer reads. See next section.

TSM provides optional preprocessing and postprocessing for user I/O requests on the TSM terminal. This processing is inhibited by certain control flags in word two of the user's FCB:

Bit	Description
0	No-wait I/O
2	Data formatting inhibited

If these control flags are not set, I/O operations on the terminal are restricted as described in the following sections.

1.11.1 Reads

On a read (M.READ), the maximum input is limited. See Figure 1-2. I/O is automatically buffered to enable task swapping during I/O wait. The TSM scanner is initialized at I/O completion. Error returns are not honored, but error status is returned in the FCB. Carriage returns are appended, but the actual input byte count is returned in the FCB. The entire input buffer is blank-filled prior to input to insure proper parsing by the scanner. The special control characters described in Table 1-1 are honored. Reads directed to a TSM terminal in batch mode, such as AS lfc TO DEV=TYnnnn, are redirected to read from SYC.

1.11.2 Writes

The maximum output record on a write (M.WRIT) is limited to the width of the terminal specified at SYSGEN unless overridden by the TSM $LINESIZE directive. A line counter is maintained to detect the bottom-of-screen, and bottom-of-screen logic is in effect. See Section 1.9. Output is only buffered if required by the controller. Error returns are not honored, but error status is returned in the FCB. The carriage control characters in position 1 are also in effect as described in the MPX-32 Reference Manual, Volume I, Chapter 5.

I/O Mode	LFC	Maximum Input
Interactive	UT	Line size (or for FORTRAN I/O, the buffer size, if smaller)
	SYC	80 bytes or line size (if smaller)*
Command file	UT	Line size (or for FORTRAN I/O, the buffer size, if smaller)
	SYC	80 bytes or line size (whichever is smaller)*
Batch	UT	Line size (or for FORTRAN I/O, an RT90 abort)**
	SYC	80 bytes or line size (whichever is smaller)*

* Parameter substitution is limited to 80 bytes regardless of previous line size specifications.

** Actual result depends on the access mode specified at open. In batch mode, UT is SLO and is opened as append mode by default.

Figure 1-2. Input Limitations for I/O Through the SYC and UT

1.11.3 Close and Open

When accessing a terminal, a task does not have to use an M.CLSE or M.OPENR system service. IOCS handles the open and TSM handles the close for UT automatically.

1.11.4 Rewind

The M.RWND service returns the TSM scanner to the first field in the TSM line buffer.

1.12 TSM Options

Options control various aspects of the user's operating environment and set flags that may be tested by a task in execution. Options are specified on a $OPTION directive. See the $OPTION directive description for a list of TSM and batch options.

1.13 Developing an Interactive Task

There are a number of functions that TSM automatically handles for a task that is activated in the on-line environment. It handles sequencing for all interactive tasks, using time-distribution priorities 55-64 and time-sharing algorithms to provide maximum response to each terminal on an equal time slice basis. It returns the TSM prompt to the terminal when a task activated in the on-line environment aborts or exits. It handles break and wakeup interrupts to let the terminal user communicate with a task. It supplies override assignments for SYC that allow a task cataloged for reads from SYC to

read from the terminal without special programming or cataloging. The user interaction in these cases is described in previous sections.

The following interactive system services are available:

- M.TSCAN Accesses the TSM scanner
- M.TBRKON Used by a task to access TSM break handling capability

Regular calls to IOCS (read, write, etc.) are handled by TSM for a task with an assignment to UT.

Conversion and time format interactive services are also available and are described in the system services section of the MPX-32 Reference Manual, Volume I, Chapter 6.

1.13.1 TSM Scanner (M.TSCAN/M_TSCAN)

When a terminal user enters a $RUN directive with the name of a task, TSM loads the entire input line in a line buffer in memory pool, scans the task name, activates the task, and leaves the pointer in its scanner at the next field.

The M.TSCAN service is called by the nonbase mode assembler task. Likewise, M_TSCAN is called by a base mode task to have TSM pass additional user-supplied fields.

For subsequent interface to the terminal, the task also uses M.TSCAN. Each read from a logical file code assigned to UT puts the line typed at the terminal into the OS line buffer and initializes the scanner to point to the first field.

Functional Description

A record is terminated by a carriage return. The parameters (fields) to be scanned are all in the user's line buffer and are reinitialized during each terminal read. Each subsequent call to M.TSCAN or M_TSCAN returns another argument (field) from the buffer.

Notes:

TSM maintains the address of the line buffer in the user's TSA.

The current scan position is updated automatically each time this service is used.

A field accessed by this service is left-justified, blank-filled, and stored in registers six and seven.

Register five contains the character count of the last field found by the scanner. When the character count in register five is zero and the delimiter in register four is a carriage return, there are no more fields in the line.

The M.RWND service resets the cursor at the first field in the current input record. The input line is scanned by M.TSCAN and M_TSCAN without any additional IOCS calls.

M.TSCAN and M_TSCAN ignore all blanks encountered before the first parameter or delimiter. If a delimiter is encountered before the first parameter, all blanks are still ignored until the first parameter is encountered.

M.TSCAN and M_TSCAN recognize a pathname of 1 to 16 characters in length and enclosed within single quotes. When the single quote character is used as a delimiter, the following characters are valid in pathnames: A through Z, 0 to 9, dot, underscore, blanks, commas, equal signs, percent signs, semicolons, exclamation points, and left and right parentheses. The single quote character can be in a pathname by using two consecutive single quotes. The length of the pathname is the number of characters between the delimiting single quotes. Consecutive single quotes are counted as a single character.

Character strings that are not complete pathnames are limited to 1 to 16 characters in length and are not enclosed within single quotes. The following characters are delimiters for this character string and should not be used within the character string: blanks, commas, semicolons, equal signs, carriage return, and left and right parentheses.

Entry Conditions	Nonbase Mode			Base Mode	
Calling Sequence:	M.TSCAN		(or)	M_TSCAN	
	SVC	1,X'5B'	(or)	SVC	1,X'50'
	(or)			(or)	
	M.CALL	H.TSM,2		M_CALL	H.TSM,2

Exit Conditions

Return Sequence:	M.RTRN	4,5,6,7 or
	M.RTRN	With CC1 set if a line buffer is not found. R4 contains a carriage return and R5 contains zero.
Registers:	R4	Delimiting character
	R5	Number of significant characters before delimiter
	R6,7	First eight characters of the character string, left-justified. The entire character string is in words 0 through 3 of the terminal line buffer.
		For nonbase mode tasks, the terminal line buffer address can be obtained by accessing the T.LINBUF variable in the TSA. Base mode tasks must locally define the value of T.LINBUF in order to access that field of the TSA.

1.13.2 TSM Break Processor (M.TBRKON/M_TBRKON)

M.TBRKON and M_TBRKON process a pause or break from the terminal or calling task. The following prompt is displayed on the terminal:

```
** BREAK ** ON:taskname AT:location CPU TIME = n SEC.
CONTINUE, ABORT, DEBUG, OR HOLD?
```

Enter C to resume execution of the task at the instruction following the call. Enter A to abort the task. Enter D to load the debugger as an overlay and transfer control to the debugger. Enter H to place the task on hold.

M.TBRKON and M_TBRKON are also the default receivers for any on-line tasks and are called as a result of a hardware or software break. If a user Transfer Control Word (TCW) is loaded in register 2, it is printed along with the break message. Refer to the MPX-32 Reference Manual, Volume I, Chapter 5 for additional information on the TCW.

Entry Conditions	Nonbase Mode	Base Mode
Calling Sequence:	M.TBRKON tcw (or) (or) ZR R2 (or) LW R2, tcw (or) SVC 1,X'5C' (or) (or) LW R2, tcw (or) ZR R2 M.CALL H.TSM,6	M_TBRKON [TCW=] tcw (or) LW R2,tcw (or) ZR R2 SVC1, X'5C' (or) LW R2, tcw (or) ZR R2 M_CALL H.TSM,6

Exit Conditions

Return Sequence:	M.RTRN	
Registers:	None	
Abort Cases:	TS01	If user enters A (abort) to the break prompt described above
	RX34	If called within a user break receiver, aborts when the user break receiver's M.BRKXIT or M_BRKXIT is processed

1.14 Activating Tasks from Directive Files

Tasks are activated from directive files. Any parameters passed with the $RUN directive are accessed by an interactive task through the M.TSCAN or M_TSCAN service. Records following the $RUN directive line within the directive file are accessible to tasks; however, any directive preceded by a dollar sign ($) is interpreted as a directive to the command processor and a pseudo end-of-file for the currently executing task.

If the task exits, the command processor continues reading from the directive file, interpreting the next $directive as its own.

If the task aborts, all subsequent records on the SYC are ignored until a directive preceded by a dollar sign is encountered.

1.14.1 Chaining Directive Files

$SELECT directives can be embedded in directive files to access other directive files. Without conditional processing, the chaining capability is directly from one directive file to the next; for example, the first $SELECT encountered accesses a new directive file. However, with conditional processing, the user can select alternate directive files to gain more flexibility in directive file processing.

Using task names identical to directive file names should be avoided. The task names take precedence over the directive file unless the $SELECT directive is used.

1.14.2 Directive File Error Processing

Errors encountered by TSM in directive file processing include:

- Supplying invalid parameters for a TSM directive or the unsuccessful execution of a directive. Corrects the line that caused the error and continues processing from the directive file. Processing from the directive file is terminated by a $CLEAR directive.

- Interactive errors encountered by a task or processor activated through a directive file. If an error is encountered by a task or processor that has been activated from a directive file, the error and recovery are the same as in normal interactive processing. A break is issued at any time and a continue, abort, or debug path is followed. When a task is aborted, control returns to TSM. TSM continues processing directives from the directive file at the point following the abort. $IFT and $IFF directives are used for conditional processing on abort.

1.15 Conditional Processing and Parameter Passing

The $IFP, $IFA, $GOTO, $IFT, $IFF, $DEFNAME, $SET, and %name directives provide the ability to:

- Substitute parameters. Up to eight predefined variables are passed to a file when it is accessed, providing the capability to adapt one directive file to various run-time uses. These parameters can be reassigned within the macro directive file.

- Provide parameter defaults. A default value is obtained in run-time parameter passing by omitting a parameter or by using an extra comma or other valid nonblank command processor delimiter to indicate a missing parameter.

- Use conditional execution capabilities. Based on a value actually supplied for a particular condition, the directive file branches to a particular set of directives or selects a different directive file altogether.

1.16 Argument Replacement by Macro Directive Files

An inherent feature of the SYC file is the argument replacement facility. A directive file that requires argument replacement is called a macro. Up to eight dummy arguments or parameters are defined when a macro file is created. When the macro file is invoked (by TSM's $BATCH for batch jobs, or $SELECT for interactive processing), up to eight parameters can be specified. These parameters, or user-established defaults, are replaced in the text of the macro file as it is read from the SYC. Dummy arguments within the text are identified by a percent sign (%) preceding the argument name. The correlation between the called parameters and the dummy arguments is established by the $DEFM directive. If argument replacement is required, $DEFM must be the first record within the directive file. See the $DEFM directive description.

1.17 Concatenating a Value to a User-supplied Parameter

A directive file that uses parameter substitution uses two optional forms of a parameter name:

```
value %par
   or
%par;value
```

This allows the user who develops the directive file to append a constant string before or after the variable value (%par) supplied by the user of the directive file.

1.18 Spooled Input Control by $SELECTx

MPX-32 provides an input spooling capability to preprocess files before they are submitted to the batch stream. For one or more input devices or disc resources, the input spoolers J.SSIN1 and J.SSIN2 copy data into a single SYC file. For one or more binary devices or discs, J.SSIN1 and J.SSIN2 also copy data into a single SGO file. The SYC and SGO files are temporary, unnamed spool type files that are deleted after job processing is complete. Data flow of a batch job is shown in Figure 1-3.

Batch data can be spooled to SYC files from devices and files specified by the TSM BATCH directive, the OPCOM BATCH directive, the EDIT BATCH directive, the Submit Job from Disc File (M.CDJS) system service, or the Batch Job Entry (M.BATCI ') system service. These devices and files are designated as primary system input sources. While a job is being spooled to its SYC/SGO file, data from alternate sources may be merged with data from the primary source. Alternate system input sources are designated by $SELECTx directives that can be included anywhere in the job following the $JOB directive. When a $SELECTx directive is encountered during input spooling, the $SELECTx directive is not written to the SYC file, but is replaced by data from the device or file specified in the directive. However, $SELECTx directives that contain errors are written to the SYC file.

Each $SELECTx directive that specifies a valid device or file establishes a new alternate level. A maximum of three alternate levels is provided. The conditions which an alternate level is reset to a previous level (alternate or primary) are summarized in Table 1-2.

A device or file can be specified on a $SELECTx directive that is the primary source or an alternate system input source at a previous level. If so, a new alternate level is established, but reading resumes from the device or file at its current position. This occurs for devices only if the device is identically specified, like the specification MT1000 is not identical to MT10.

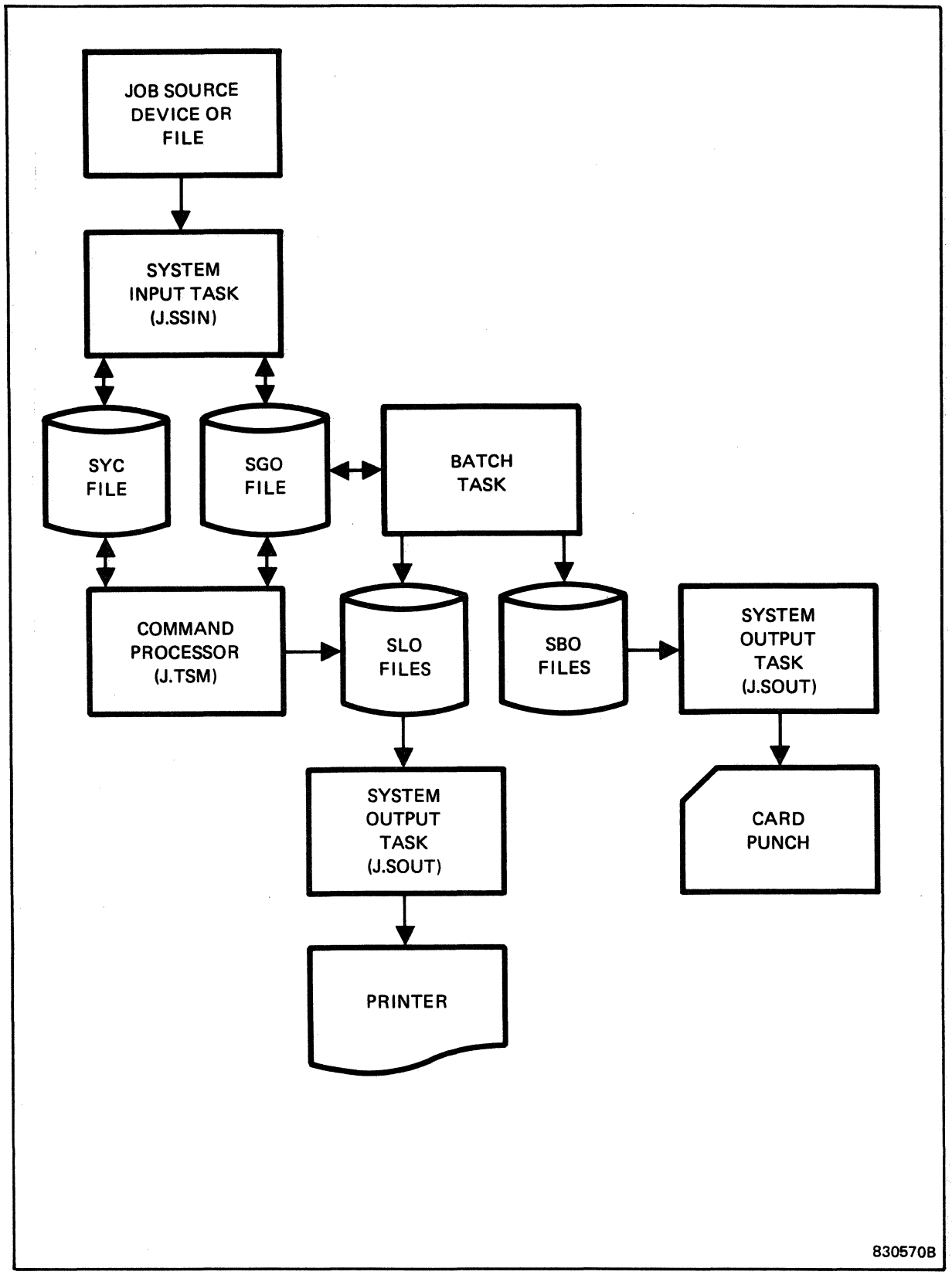


Figure 1-3. Data Flow for a Job

Table 1-2
Terminating Conditions for Spooled Input Processing

Condition	Processing
I/O Error	On alternate source: performs end-of-job processing (ENDJOB). Resets to primary source. If primary source is a device, continues reading from device. If primary source is a file, terminates reading from the file.
	On device or file primary source: ENDJOB and terminates reading from device or file.
End-of-Medium	On alternate source: ENDJOB and resets to primary source. If primary source is a device, continues reading from device. If primary source is a file, terminates reading from the file.
	On device or file primary source: ENDJOB and terminates reading from device or file.
End-of-File	On alternate or primary card device source: writes pseudo end-of-file (EOF record) to SYC and continues reading from device.
	On alternate noncard device source: reverts to previous level if specified number of files have been read.
	On primary noncard device source: continues reading from device.
	On file primary source: ENDJOB and terminates reading from file.
$EOJ	On alternate or primary source: ENDJOB and continues reading from source.
$$	On alternate source: ENDJOB and resets to primary source. If primary source is a device, continues reading from device. If primary source is a file, terminates reading from the file.
	On device primary source: ENDJOB and terminates reading from device if continuous batch mode is not set. If continuous batch mode is set, continues reading from device.
	On file primary source: ENDJOB and terminates reading from the file.
$$$	On alternate source: identical to $$.
	On device or file primary source: terminates reading from device or file.

1.19 TSM Directives

TSM directives are entered with or without a preceding dollar sign in the interactive mode. The dollar sign must be specified to process directive files in the batch mode.

The general syntax of TSM directives is:

```
$keyword [arguments...]
```

The keyword can be abbreviated as indicated by the underlining in the directive descriptions that follow. Valid delimiters between arguments are blanks, commas, equal signs, semicolons, or parentheses. Where the order of arguments is implicit, the absence of an argument is specified by inserting an extra comma. For example:

```
$A3 LFC=MT,REEL,,UNBLOCKED
```

1.20 $ACCOUNT Directive

The $ACCOUNT directive displays the entries in the job accounting file to logical file code LO which is assigned to SLO. Keyword parameters can contain leading or trailing wild card characters in the form of question marks.

If no parameters are specified, the entire contents of the job accounting file are displayed.

Syntax:

$ACCOUNT[,OWNE=name] [,PROJ=proj] [,DATE=date] [,ORIG= {TSM.nnnn | BATCH}]

OWNE= specifies the owner name associated with the files to be displayed

PROJ= specifies the project group name associated with the files to be displayed

DATE= specifies the numeric date associated with the files to be displayed, entered in the format month/day/year

ORIG= specifies the mode of operation in which the files to be displayed were used

Usage:

```
$ACCOUNT,OWNE=USER1,DATE=06/16/80,ORIG=TSM.20??
```

Displays statistics on all jobs with owner name USER1 run on June 16, 1980 on any command processor device 20xx.

```
$ACCOUNT,PROJ=082950
```

Displays statistics on all jobs run under project number 082950.

1.21 $ACTIVATE Directive

The $ACTIVATE directive initiates the execution of a specified task at its cataloged priority. JCL directives are processed immediately after the task is activated.

The task is activated at its base priority with its pseudonym set to the job's sequence number. The task's owner name is set to the owner name from the $JOB directive. Tasks activated by the $ACTIVATE directive cannot perform I/O to SYC or SGO files (for SYC or SGO, use $EXECUTE).

$ASSIGN and $OPTION directives are used prior to $ACTIVATE to assign files or devices and set option bits for a real-time task.

This directive can be used when a task is in a hold state.

Syntax:

$ACTIVATE pathname

pathname is the pathname of a load module or executable image. If the task is not available for multicopying and another copy is already in execution, the message UNIQUE TASK IS ALREADY ACTIVE is displayed and the next statement is processed.

Usage:

```
TSM>$ACTI JOBFILE
YOUR TASK # IS taskno
```

1.22 $ALLOCATE Directive

The $ALLOCATE directive increases the amount of memory allocated for execution of a nonbase task.

If the size of the operating system plus the size of the task plus the size of the allocation is more than 128KW, the task cannot be loaded and an abort condition occurs.

The $ALLOCATE directive is only used on nonbase mode tasks. If the number of bytes specified is less than the task's cataloged memory requirements, or if the task is executed in the base mode, the directive is ignored.

Syntax:

$ALLOCATE bytes

bytes is the total hexadecimal number of bytes to be allocated for a task, excluding the Task Service Area (TSA)

Usage:

```
START    EQU    $
           .
           .
           .
         SVC    1,X'55'
           .
           .
           .
BUFFERX  DATAW  1W
         END    START
```

Using the $ALLOCATE directive allows for increasing the size of BUFFERX and having a label attached to it.

1.23 $ASSIGN Directive

The $ASSIGN directive supplies default assignments for logical file codes used by the task being run.

This statement applies to the task subsequently executed by $DEBUG, $EXECUTE, $ACTIVATE, or a TSM activation without a preceding directive verb.

Syntax:

```
                                  { SBO                    }
                                  { SLO                    }
                                  { SYC                    }
                                  { SGO                    }
                                  { @ANSITAPE(lvid)file    }
                                  { pathname               }
                                  { RID=resid              }
$ASSIGN lfc TO                    { TEMP[= (volname)]      } [FORMAT= format] [SIZE=blocks]
                                  { DEV=devmnc             }
                                  { LFC=lfc                }

[SHARED= bool] [GENERATION=gennum] [GENVERSION=genvum] [BSIZE=bsize]
[RECLENGTH=recsize][ACCESS=( [READ][WRITE][MODIFY][UPDATE][APPEND] ) ]
[BLOCKED= bool]

[EXPIRE = { date  }]  [PRINT]  [DENSITY= { N    }]  [PROTECT = { 0    }]
          { +days }   [PUNCH]            { P    }               { A...Z }
                                         { G    }
                                         { 800  }
                                         { 1600 }
                                         { 6250 }

[MULTIVOL=number] [ID=id] [BBUF=buffers]
```

SBO	treat resource as System Binary Output
SLO	treat resource as System Listed Output
SYC	treat resource as a System Control file
SGO	treat resource as a System General Object file
@ANSITAPE	treat resource as an ANSI labeled tape
lvid	is the one- to six-character logical volume identifier previously mounted by the ANSI labeled tape AMOUNT utility
file	is a one- to seventeen-character file identifier
pathname	is the pathname to be associated with the resource
resid	is a unique resource identifier (including the volume name, creation date, creation time, resource descriptor block, resource type, and code) returned by the system when a resource is created
volname	is the volume name on which temporary space is to be allocated. If not specified, the default is any volume.

devmnc — is the device mnemonic of a configured peripheral device. See Appendix A.

lfc — is a one- to three-character logical file code used in the task. For an ANSI labeled tape, only one LFC can be assigned to an lvid. Before further assignments can be made, the M.DASN service must be used.

format — is the ANSI labeled tape record format. If not specified, the default for write access is D. For read access, the format is read from the tape. The formats are:

Format	Description
F	Fixed length
D	Variable length
S	Spanned

blocks — specifies the initial size, not greater than 65,535 blocks, of a file in logical blocks. If not specified, the default is 16 blocks. If EOM is encountered, the file extends automatically. This option is only valid when used with the TEMP parameter.

SHARED — if yes (Y) is specified, the resource is explicitly shared. If no (N) is specified, the resource is exclusive. If not specified, the default is implicitly shared. This option is only valid when used with the pathname, RID, TEMP, and DEV parameters.

gennum — is the one- to four-decimal digit ANSI labeled tape file generation number. On input (read access), this number must match the generation number of the ANSI tape file that is being assigned. On output (write, update, or append access), this value becomes the generation number of the new ANSI tape file. If not specified, the default is one on output; no check on input.

genvum — is the one- or two-decimal digit ANSI labeled tape file generation version number. On input (read access), this value must match that of the ANSI tape file. On output (write, update or append access), this value becomes the generation version number of the new ANSI tape file. If not specified, the default is zero on output; no check on input.

bsize — is read from the ANSI labeled tape on read access. For other types of access, the value specifies the byte size of each data block including the padding on an ANSI labeled tape. A maximum bsize of 2048 provides sufficient space for ANSI tape-switch label information after the physical end-of-tape marker. If not specified, the default is 2048 bytes.

recsize — is read from the ANSI labeled tape header on read access. For other types of access, this value specifies the record size for fixed length records or the maximum record size for spanned and variable length record formats. The maximum size for recsize is bsize. If not specified, the default is 80.

ACCESS — specifies the type of access for resource. This must be a subset of access allowed at resource creation. If not specified, the default is the access specified at resource creation. This option is only valid when used with the @ANSITAPE, pathname, RID, TEMP, and DEV parameters.

For ANSI tapes, only read, write, update and append can be specified. The ANSI default is read.

ACCESS for ANSI labeled tapes is as follows:

Value	Description
R	Read existing file
W	Create file at first unexpired file on tape
A	Create file at end of tape
U	Overwrite existing file with a new file of the same name

BLOCKED — if yes (Y) is specified, the resource is explicitly blocked. If no (N) is specified, the resource is explicitly unblocked. If not specified, the default is blocked. This option is only valid when used with the @ANSITAPE, pathname, RID, TEMP, and DEV parameters.

EXPIRE — specifies the termination date of an ANSI labeled tape file. If the file has a termination date that is later than the file that physically precedes it, the termination date is identical to the termination date of the preceding file. If a file has a termination date that is earlier than the file that physically precedes it, the files will expire on the earlier termination date. If not specified, the default is +30 days from creation.

date — specifies the date after which an ANSI labeled tape file can be overwritten. The date is given in ASCII format--YYDDD where YY is the year and DDD is the day number within the year (January 1 is 001). If the date is 00000, or a date prior to the current date, the file has been terminated and is no longer accessible.

+days — specifies the number of days after the creation date that an ANSI tape file can be overwritten. This number must be preceded with a plus (+) when entered. If not specified, default is +30 days.

WARNING: If the number of days is not preceded by a plus (+), the number entered can be read as the date.

PRINT — indicates the file is to be printed after deassignment. This option is only valid when used with the pathname, RID, and TEMP parameters.

PUNCH — indicates the file is to be punched after deassignment. This option is only valid when used with the pathname, RID, and TEMP parameters.

DENSITY — specifies density of high speed XIO tape. If not specified, the default is 6250 BPI. This option is only valid when used with the DEV parameter.

PROTECT — specifies protection for new ANSI labeled tape files. Zero specifies owner only access. A...Z are reserved by the ANSI specification for installation-specific protection. MPX-32 treats A...Z as owner-only protection. If the correct protection value is not specified when using an ANSI labeled tape, an I/O error occurs. If a user signs on as 'system', any protection value or ownername written by J.LABEL can be overridden. If not specified, the default is no protection.

MULTIVOL — is a volume number for a multivolume tape. If not specified, the default is zero (not multivolume). This option is only valid when used with the DEV parameter.

ID — is an identifier for an unformatted medium. If not specified, the default is SCRA (scratch). This option is only valid when used with the DEV parameter.

buffers — is the number of 192W blocking buffers if using a large blocking buffer. If not specified, the default is one.

Notes:

1. To continue parameters over more than one input line, a hyphen (-) must terminate the current input line. A blank space is required before the hyphen as shown in the following example:

```
$ASSIGN ABC TO DEV=M9 DENSITY=800 -
BLOCKED=Y
```

2. An individual parameter cannot be split between input lines.

1.24 $ASSIGN1 Directive

The $ASSIGN1 directive assigns permanent files for logical file codes used by the task being run.

This directive is provided for compatability with earlier releases of MPX-32. It is recommended that the $ASSIGN directive be used.

Syntax:

$ASSIGN1 lfc=filename [,[password] [,U]] [lfc=...]

lfc — is a one- to three-character logical file code used in the task

filename — is a one- to eight-character name of a disc file (in the current working directory or the system directory) to assign to the LFC

Any one of the optional parameters following the file name can be entered as shown in the syntax statement. Each option must be separated by a comma. If an option is omitted, a comma must be inserted in its position; for example, filename,,U

password — is ignored

U — specifies the file is optionally unblocked. If not specified, the default is blocked.

If multiple LFC assignments are made with one $ASSIGN1 directive, at least one blank must separate each LFC assignment.

Usage:

```
$ASSIGN1 LIB=LIBRARY,,U  DIR=DIRECTORY,,U
```

is equivalent to:

```
$ASSIGN LIB TO LIBRARY BLOC=N SHAR=Y
$ASSIGN DIR TO DIRECTORY BLOC=N SHAR=Y
```

```
$ASSIGN1  OT=OUTFILE  IN=INFILE,MYPASS
```

is equivalent to:

```
$ASSIGN OT TO OUTFILE
$ASSIGN IN TO INFILE
```

1.25 $ASSIGN2 Directive

The $ASSIGN2 directive supplies system file assignments to logical file codes. An LFC assignment to a system file results in IOCS creating one of the following types of files for use by the task:

System Binary Output (SBO): A temporary file created and used by IOCS for buffering output to the device defined at SYSGEN or by the OPCOM SYSASSIGN directive as POD (Punched Output Device). Output from the user task directed to the LFC associated with SBO is buffered and routed by IOCS to the POD.

System Listed Output (SLO): A temporary file created and used by IOCS for buffering output to the device defined at SYSGEN or by the OPCOM SYSASSIGN directive as LOD (Listed Output Device). Output from the user task directed to the LFC associated with SLO is buffered and routed by IOCS to the LOD.

System Control File (SYC): The command processor automatically assigns SYC to UT if a directive file does not exist. Tasks should use $ASSIGN4 to equate any other logical file codes to UT.

System General Object (SGO): A temporary file used to accumulate object code.

The $ASSIGN2 directive is provided for compatability with earlier releases of MPX-32. It is recommended that the $ASSIGN directive be used.

Syntax:

$ASSIGN2 lfc= { SBO,cards / SLO,printlines / SYC / SGO } [lfc=...]

lfc — is a one- to three-character logical file code used in the task

SBO — is the System Binary Output file

cards — is the number of cards expected as output from an object deck. This specification determines the size of the SBO temporary file.

SLO — is the System Listed Output file

printlines — is the number of print lines required for listed output. This specification determines the size of the SLO temporary file.

SYC — is the System Control file

SGO is the System General Object file. SYSGEN size is overridden by the $JOB directive.

If multiple LFC assignments are made with one $ASSIGN2 directive, at least one blank must separate each LFC assignment.

Usage:

```
$ASSIGN2 CAR=SBO,1000
```

is equivalent to:

```
$ASSIGN CAR TO SBO
```

```
A2 OUT=SLO,50 OT2=SGO
```

is equivalent to:

```
$ASSIGN OUT TO SLO
$ASSIGN OT2 TO SGO
```

1.26 $ASSIGN3 Directive

The $ASSIGN3 directive supplies device assignments for logical file codes used by the task being run.

This directive is provided for compatability with earlier releases of MPX-32. It is recommended that the $ASSIGN directive be used.

Syntax:

$ASSIGN3 lfc=devmnc, [blocks / reel [,vol]] [,U] [lfc=...]

lfc — is a one- to three-character logical file code used in the task

devmnc — is the device mnemonic of a configured peripheral device. See Appendix A.

blocks — is the number of disc blocks (192 words) to be allocated for this file

reel — specifies a one- to four-character identifier for the reel. If not specified, the default is SCRA (scratch).

vol — is the volume number for a multivolume tape. If not specified, the default is 0 (not multivolume).

U — specifies the tape or disc is unblocked. If not specified, the default is blocked.

If a parameter is omitted, a comma must be inserted in its position. If multiple LFC assignments are made with one $ASSIGN directive, at least one blank must separate each LFC assignment.

Usage:

Tape: $A3 IN=M91000,SRCE,,U OT=PT

is equivalent to:
```
$ASSIGN IN TO DEV=M91000 ID=SRCE BLOC=N
$ASSIGN OT TO DEV=PT
```

Disc: $A3 IN=DC,20

is equivalent to:
```
$ASSIGN IN TO TEMP SIZE=20
```

1.27 $ASSIGN4 Directive

The $ASSIGN4 directive associates one or more logical file codes used by the task being run with an existing LFC assignment. This assignment remains in effect for the associated file or device even if the original assignment is deallocated.

A logical file code assigned to the logical file code UT implies an assignment to the user's terminal.

This directive is provided for compatability with earlier releases of MPX-32. It is recommended that the $ASSIGN directive be used.

Syntax:

$ASSIGN4 lfc=lfc [lfc=lfc]

lfc=lfc is a pair of logical file codes, where the first LFC is the new assignment and the second is the LFC already associated with a file or device in any previous $ASSIGN directive (including $ASSIGN4). Any number of LFC to LFC assignments can be specified. In the interactive mode, the user's terminal is preassigned to UT.

Usage:

```
$A1 IN=PERMDISC
$A4 IN2=IN
```

is equivalent to:

```
$ASSIGN IN TO PERMDISC
$ASSIGN IN2 TO LFC=IN
```

1.28 $BATCH Directive

The $BATCH directive submits a directive file to run in the batch stream. The file format must be blocked. The command processor searches for the specified file under the current working volume and directory in effect when the $BATCH directive is issued. See Section 1.80 for details on batchstream/memory pool interaction.

If sequential execution is required for jobs run by the $BATCH directive, the S parameter must be specified on the $JOB directive.

When using the $BATCH directive, the $JOB directive must be the first card in the directive file, unless the $DEFM directive is specified. If $DEFM is specified, then $JOB must be the second card in the directive file. If multiple $JOB directives are present, they must be preceded by the initial $DEFM card or the $EOJ from the previous job.

$BATCH is similar to the OPCOM BATCH directive except that argument passing is not supported by OPCOM BATCH.

This directive can be used when a task is in a hold state and is valid in the interactive mode only.

Note: Proper parameter substitution is only supported for the first job if the directive file has multiple jobs.

Syntax:

$BATCH pathname [par1] [par2] ...

pathname is the pathname of a permanent file containing TSM directives

par1 is a parameter to pass for the directive file. Up to eight parameters can be passed.

Response:

The specified file is verified and enqueued. When selected by the command processor, the job number is displayed as follows:

jobnumber ownername $JOB jobname

1.29 $CHANGE Directive

The $CHANGE directive establishes a new default project group name or working directory for all subsequent resource specifications. This directive can be used when a task is in a hold state.

This directive remains in effect until overridden by another $CHANGE directive or until logging off the system.

The project group is used for accounting purposes and to establish access restrictions for resources. When a project group is changed, any nonzero CPU and IPU execution time for the previous session is recorded in the M.ACCNT file under the old project group.

The working directory is used to specify the pathname to be prefixed for file name specification. If a nonpublic volume is specified, a TSM $MOUNT directive must be issued before the $CHANGE DIRECTORY directive.

Although both PROJECT and DIRECTORY are optional, one of them must be specified with $CHANGE. Both can be specified in one directive.

Syntax:

$CHANGE [PROJECT=projname [,key]] [DIRECTORY=pathname]

projname is the name of the new default project name

key is the one- to eight-character key associated with the project name. If not specified, the default is no key.

pathname specifies the pathname of the new default directory. If not specified, the default is the current working directory. Due to the method J.TSM uses to parse pathnames, parentheses in pathname should be avoided, especially if the directory pathname is followed by the project keyword.

Usage:

```
$CHANGE DIRE=@NEWVOL1^NEWDIR
```

Changes the current working directory to NEWDIR on volume NEWVOL1.

```
$CHAN PROJ=NEWPROJ
```

Changes the default project group to NEWPROJ.

```
$CHANGE PROJ=061643 DIRE=ATLAS
```

Changes the default project group to 061643 and current working directory to ATLAS on the current working volume.

1.30 $CLEAR Directive

The $CLEAR directive terminates output spools and clears all previous $SHADOW, $ASSIGN, $OPTION, $ALLOCATE and $EXTDMPX directives. $CLEAR also terminates processing from a directive file before end-of-file and returns control to the terminal user.

Syntax:

$CLEAR

1.31 $CONTINUE Directive

The $CONTINUE directive resumes execution of a task that was placed in a hold state in response to a CONTINUE, ABORT, DEBUG, or HOLD? prompt. The task resumes execution until completion or until another break is issued.

This directive is valid in the interactive mode only.

Syntax:

$CONTINUE

1.32 $CREATE Directive

The $CREATE directive creates a permanent file. This directive can be used when a task is in a hold state. Directive file error processing or batch job termination occurs if the create operation fails.

Syntax:

$CREATE pathname

pathname specifies the pathname of the file being created. The established system default access rights apply to the file.

To avoid an error message if the file being created already exists, the following format is used:

```
$IFT PATH=@ATLAS04(NEWDIR)INDEX SKIP
$CREATE @ATLAS04(NEWDIR)INDEX
$DEFNAME SKIP
```

Usage:

$CREA INDEX

Creates file INDEX on the current working volume and directory.

$CREA @ATLAS04(SOURCE)INDEX

Creates file INDEX on volume ATLAS04 in directory SOURCE.

1.33 $DEBUG Directive

The $DEBUG directive debugs a task (load module or executable image). The appropriate interactive debugger (MPXDB or SYMDB for nonbase mode or DEBUGX32 for base mode) is attached to the task when the task is activated.

A nonbase task containing a shared CSECT is loaded as a multicopied task with the debugger attached so that setting break points in the CSECT does not impact other users of the CSECT.

Syntax:

$DEBUG pathname

pathname is the pathname of a load module or executable image. This name is the same as the pathname of the file containing the load module or executable image.

1.34 $DEFM Directive

The $DEFM directive supplies parameters that are processed by the command processor when a directive file is executed. A parameter can be a substitution string for part of a value or any value normally supplied as a variable for a directive. Up to eight parameters are defined for a directive file.

A single parameter cannot contain embedded commas, blanks, or other command processor delimiters and can be a maximum of 16 characters in length. The parameters typed on the directive line when the file is executed must match the order of the parameters supplied on the $DEFM directive. If a missing value is followed by other values, the missing value must be indicated by an extra comma or other nonblank command processor delimiter.

The $DEFM directive is valid only from interactive and batch directive files. If used in a batch directive file, a $JOB statement must follow it.

Syntax:

$DEFM [par1] [,par2] ...

par1 defines a parameter. If the parameter is more than 16 characters, the extra characters are truncated.

Response:

At run time, parameters supplied with the name of the directive file are matched positionally against the parameters defined above. Each time substitution is indicated by a percent sign, the command processor matches the parameter against the input parameter and places the proper value in the terminal input buffer.

Documentation of any directive file containing $DEFM parameters should include definitions of the parameters to enter when the directive file is accessed and the exact order for entering them.

Usage:

The following directive file is used to make assignments and select standard options for assembly. SI is the parameter indicating a base file name. If the user leaves out the first parameter, SI is the name of the input file for the assembly. The macro selects one as the default option by the second parameter of the $DEFM directive.

Directive File (ASM):

```
$DEFM SI,1
$ASSIGN SI TO %SI
$ASSIGN BO TO OB%SI
$OPTION %1 3 4
ASSEMBLE
$ENDM
```

User enters: TSM> $SELECT ASM SRCE

Expansion of Directive File as Executed:

```
$ASSIGN SI TO SRCE
$ASSIGN BO TO OBSRCE
$OPTION 1 3 4
ASSEMBLE
```

The file names generated are unique and consistently related by concatenation in the two $ASSIGN directives. The $SELECT directive forces the identification of the file named ASM to be a directive file and not a load module.

1.35 $DEFNAME and %name Directives

The $DEFNAME and %name directives establish names to branch to for conditional processing.

Following $DEFNAME or %name, the user supplies the sequence of directives to process when a conditional branch occurs. A branch is the result of a directive such as $IFF, $IFT, $IFP, etc., that references the name.

This directive is valid only from interactive and batch directive files.

Syntax:

```
%
$DEFNAME name
```

name — specifies a one- to eight-character name corresponding to a name referenced in a previous directive. The name must contain at least one alphanumeric character.

Usage:

```
      .
      .
      .
$DEFN CKDIR
$IFP %VOL CKVOL
$DEFN CKVOL
$IFP %VOL CKBLD
$DEFN CKBLD
      .
      .
      .
```

1.36 $DELETE Directive

The $DELETE directive deletes a permanent file. This directive can be used when a task is in a hold state. Directive file error processing or batch job termination occurs if the delete operation fails.

Syntax:

$DELETE pathname

pathname specifies the pathname of the file to be deleted

Usage:

$DELE INDEX

Deletes file INDEX on the current working volume and directory.

$DELE @ATLAS04(SOURCE)INDEX

Deletes file INDEX on volume ATLAS04 in directory SOURCE.

1.37 $DISMOUNT Directive

The $DISMOUNT directive informs the operator to remove a volume or an unformatted medium from a device. The requestor can receive positive or negative acknowledgement of the request. The use count for the volume is decremented if the operation is successful.

If an error occurs and the dismount is not successful, one of the following messages is displayed:

*VOLUME NOT ASSIGNED

*UNABLE TO DISMOUNT VOLUME

Syntax:

$DISMOUNT volname

volname is the name of the volume to be dismounted

Usage:

$DISM TB00

Volume TB00 is dismounted if there are no other users of the volume.

1.38 $ENDM Directive

The $ENDM directive is optional and terminates a directive file. This directive is valid only in the interactive mode.

Syntax:

$ENDM

1.39 $EOJ Directive

The $EOJ directive designates the end-of-job and the termination of the output spools. If $JOB is used, $EOJ is also used. If $JOB is not used and $EOJ is used, an error message is displayed.

In the interactive mode, $EOJ does not clear assignments. Assignments are cleared by a task exit sequence or the $CLEAR directive.

Syntax:

$EOJ

1.40 $ERR Directive

The $ERR directive displays the description of valid abort codes. This directive can be used when a task is in a hold state.

If an abort code that is not defined in M.ERR is requested, the following message is displayed:

<UNRECOGNIZABLE ERROR CODE>

Syntax:

$ERR code

code is a four-character abort code

Usage:

```
TSM>$ERR AS02
PHYSICAL END-OF-FILE ENCOUNTERED ON WRITE TO THE BINARY OUTPUT
(BO) FILE
TSM>
```

1.41 $EXECUTE Directive

The $EXECUTE directive activates a task in the interactive or batch modes.

Syntax:

$EXECUTE taskname

taskname is the name of a load module or executable image file. The file is assumed to be on the system volume and directory.

Usage:

$EXEC EDIT

Activates the Text Editor utility.

$EXEC TEST1

Activates the load module or executable image file TEST1, located on the system volume in the system directory.

1.42 $EXIT Directive

The $EXIT directive logs off the system and is valid in the interactive mode only.

Syntax:

$EXIT

1.43 $EXTDMPX Directive

The $EXTDMPX directive dynamically overrides the SYSGEN and CATALOGER assignments for the logical starting address of extended MPX-32. This directive is ignored if the task is shared.

Syntax:

$EXTDMPX { logical map block number | MINADDR | MAXADDR }

logical map number — is the decimal logical map block number where extended MPX-32 is logically mapped for this task

MINADDR — specifies that extended MPX-32 will be mapped into the task's logical address space at the end of the TSA

MAXADDR — specifies that extended MPX-32 will be mapped into the task's highest available logical address space.

Usage:

```
$EXTDMPX      65
$EXTD MINADDR
$EXTDMPX MAXA
```

Note: There is no default assignment for $EXTDMPX. If $EXTDMPX is not specified, the SYSGEN or CATALOGER assignment is valid. If $EXTDMPX is specified with no parameters, an "ILLEGAL BLANK FIELD" message is displayed on the user's terminal.

1.44 $GOTO Directive

The $GOTO directive skips subsequent directives until the specified name is found in the directive file. The name is defined by a $DEFNAME or %name directive. This directive is valid only from interactive and batch directive files.

Syntax:

```
$GOTO name
```

name is a one- to eight-character string that indicates a point to go to in the file

Usage:

```
$GOTO START
      .
      .
      .
$DEFNAME START
      .
      .
      .
```

1.45 $IFA and $IFP Directives

The $IFP/$IFA directives branch to a name based on whether a particular parameter is present ($IFP) or absent ($IFA) at run time. $IFP and $IFA can be used, for example, to set up a default course when a parameter is absent or to vary processing depending upon the parameters supplied at run time when a directive file is selected.

These directives are valid only from interactive and batch directive files.

Syntax:

{$IFP / $IFA} %par name

%par is one of the parameters defined by the associated $DEFM directive

name is a one- to eight-character name that marks a point to go to in the directive file to continue processing directives. The name can be either a directive or a string that marks a branch forward through the directive file.

Usage:

In this partial example, OP is the parameter for selecting an Assembler option. It is the fourth parameter entered when the directive file is accessed.

```
$DEFM SI,ASSEMBLE,NEW,OP
          .
          .
          .
$IFA %OP ASSM
$OPTION %OP
$GOTO NOP
%ASSM
$OPTION 1
%NOP
$OPTION 3 4
```

User enters: TSM > EXAMPLE SRCE,ASSEMBLE,CREATE,2

Expanded Directives for $OPTION are:

```
          .
          .
          .
$OPTION 2
$OPTION 3 4
          .
          .
          .
```

The $IFA directive line is translated as "if the OP parameter is absent, select option 1 plus options 3 and 4. If not absent, use the specified option as indicated on the next line plus options 3 and 4."

1.46 $IFF Directive

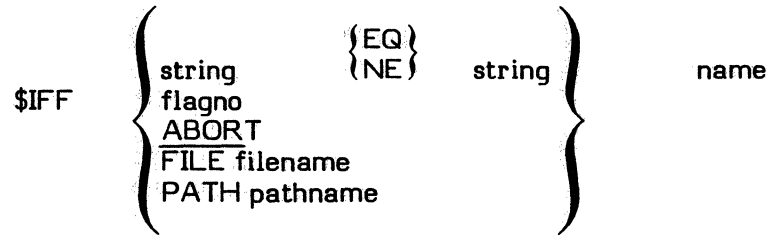
The $IFF directive branches to a name in a directive file when a condition is false. Several types of conditions can be tested:

- strings - the first character string is not equal to the second character string
- flags - the specified flag or flags are false. Flags are set to true or false by the $SETF (true) or $RESETF (false) directives.
- abort - the preceding task does not abort
- files - the specified file does not exist

If a condition is false, a name to branch to is provided in the directive file. If a condition is not false, the command processor continues processing with the next directive in the directive file.

This directive is valid only from interactive and batch directive files.

Syntax:



string — is any string with a maximum of 16 characters. The string may be the result of argument substitution or concatenation. If the string contains more than 16 characters, characters to the right are truncated. The value can also be an alphanumeric string. If the value of the string comparison is false, the command processor branches. If it is true, the next directive in the directive file is processed. Do not use a keyword (ABOR, ABORT, FILE, PATH) as a string.

flagno — is a numeric flag number from 0 to 31. One flag number can be specified. If the specified flag is false, the command processor branches. If the flag is true, the next directive in the directive file is processed.

ABORT — if the task immediately preceding $IFF did not abort, the command processor branches. If the task aborted, the next directive in the directive file is processed.

FILE filename — specifies the one- to eight-character name of a user or system file. If the file does not exist, the command processor branches. If the file exists, the next directive in the directive file is processed.

PATH pathname — specifies the pathname of a file. If the file does not exist, the command processor branches. If the file exists, the next directive in the directive file is processed.

name — is a one- to eight-character string defined by a $DEFNAME or %name directive that marks a point in a directive file. If the label referenced for a branch does not exist, the command processor continues to the end of the directive file and returns to the terminal user.

Usage:

```
$IFF %A EQ B YOURS
```

Branches when B is not entered as the value of parameter %A.

```
$IFF %A NE B MINE
```

Branches when B is entered as the value of parameter %A.

```
$IFF FILE %C SKIP
```

Branches if the file name denoted by %C does not exist in the current working directory or system directory.

1.47 $IFT Directive

The $IFT directive branches to a name in a directive file when a condition is true. Several types of conditions can be tested:

- strings - the first character string is equal to the second character string
- flags - the specified flag or flags are true. Flags are set to true or false by the $SETF (true) or $RESETF (false) directives.
- abort - the preceding task aborts
- files - the specified file exists

If a condition is true, a name to branch to is provided in the directive file. If a condition is not true, the command processor continues processing with the next directive in the directive file.

This directive is valid only from interactive and batch directive files.

Syntax:

```
$IFT  { string    {NE} string }   name
      {           {EQ}        }
      { flagno                }
      { ABORT                 }
      { FILE filename         }
      { PATH pathname         }
```

string	is any string with a maximum of 16 characters. The string may be the result of argument substitution or concatenation. If the string contains more than 16 characters, characters to the right are truncated. The value can also be an alphanumeric string. If the value of the string comparison is true, the command processor branches. If it is false, the next directive in the directive file is processed. Do not use a keyword (ABOR, ABORT, FILE, PATH) as a string.
flagno	is a numeric flag number from 0 to 31. One flag number can be specified. If the specified flag is true, the command processor branches. If the flag is false, the next directive in the directive file is processed.
ABORT	if the task immediately preceding $IFT aborts, the command processor branches. If the task did not abort, the next directive in the directive file is processed.
FILE filename	specifies the one to eight character name of a user or system file. If the file exists, the command processor branches. If the file does not exist, the next directive in the directive file is processed.
PATH pathname	specifies the pathname of a file. If the file exists, the command processor branches. If the file does not exist, the next directive in the directive file is processed.

name — is a one- to eight-character string defined by a $DEFNAME or %label directive that marks a point in a directive file. If the label referenced for a branch does not exist, the command processor continues to the end of the directive file and returns to the terminal user.

Usage:

The following example illustrates conditional processing for parameters. The NEW parameter is the point of concentration. At run time, the string CREATE can be entered to create a new library and directory file before running the Subroutine Library Editor (LIBED) utility. If the string is absent or other than CREATE, the existing library will be updated only.

Note: CREATE and DELETE tasks shown in the example are demonstration tasks only. (See the end of this chapter for an example which uses DELETE as an interactive task.)

Directive File (EXAMPLE):

```
$DEFM SI,ASSEMBLER,NEW,OP
          .
          .
          .
$IFT %NEW NE CREATE OLD
DELETE DIR
DELETE LIB
CREATE LIB
CREATE DIR
$OPTION 1
%OLD
$AS LLO TO LFC=UT
$AS DIR TO DIR BLOC=N
$AS LIB TO LIB BLOC=N
$AS LGO TO OB%SI
LIBED
```

User enters: TSM > EXAMPLE SRCE,ASSEMBLE,CREATE

Expanded Directives are:

```
          .
          .
          .
DELETE DIR
DELETE LIB
CREATE LIB
CREATE DIR
$OPTION 1
$AS LLO TO LFC=UT
          .
          .
          .
$AS LGO TO OBSRCE
LIBED
```

The $IFT directive is translated as "if the NEW parameter (third parameter) is any valid string other than CREATE, branch to OLD". If the run time value string supplied is CREATE, delete the existing library and directory files and create new file spaces, and select option 1 before making LIBED assignments. The name %OLD moves the run time user past the CREATE and DELETE tasks to update existing library files if the NEW parameter supplied is not equal to the character string CREATE.

1.48 $JOB Directive

The $JOB directive identifies a job to the system. It is required as the first statement of a job unless preceded by a $DEFM directive. $JOB is used with $EOJ to delimit a job. $JOB is optional in the interactive mode, but should be used when SGO is required or when output spooling options are specified.

Syntax:

$JOB jobname [ownername,key] [SLOF=file] [SBOF=file] [SLOD=devmnc]

[SBOD=devmnc] [S] [SGO=size] [SLO=size] [SBO=size]

jobname is a one- to eight-character job identifier. If more than eight characters are specified, only the first eight characters are used.

ownername is an owner name. If not specified, the default is the owner name used to logon. If the job is run in the batch mode and the owner name is key-protected, the default is SYSTEM.

key is the key associated with the owner name in the M.KEY file (if any).

file is the name of a permanent disc file to contain the job's SLO or SBO. If the specified file does not exist, a file is dynamically created.

SLOF/SBOF SLOD/SBOD These fields specify the final destinations of SLO and SBO files generated by the job. If these fields are omitted, final destinations for SLO and SBO files are automaticaly selected from eligible devices specified by the SPOOL parameter of SYSGEN DEVICE directives. These fields are interpreted as follows:

Field	Description
SLOD	Any SLO files generated by the job are output to the specified device.
SLOF	Any SLO files generated by the job are accumulated on the specified permanent disc file. If the specified file does not exist, it is automatically created.
SBOD	Any SBO files generated by the job are output to the specified device.
SBOF	Any SBO files generated by the job are accumulated on the specified permanent disc file. If the specified file does not exist, it is automatically created.

devmnc is the six-character device mnemonic of the final destination device for SLO or SBO files. The device may be a paper tape punch, line printer, or magnetic tape device. The entry consists of a two-character device code followed by a four-character hexadecimal device address, such as LP7EF8. (See Appendix A.) The device address consists of a two-character device channel number followed by a two-character device subaddress. If the subaddress is omitted, zero is assumed. If the entire device address is omitted, a channel number and subaddress of zero are assumed. Reel identifiers should not be specified for magnetic tape devices.

S — specifies sequential execution is required for jobs run by $BATCH or $RUN directives. A sequential job is not run until all previously entered sequential jobs are completed. This parameter is ignored in the interactive mode or if the job was activated by a TSM $SUBMIT directive.

size — specifies the initial number of 192-word blocks of disc space to allocate for the job's SGO/SLO/SBO file. If an SLO, SGO or SBO size is not specified, the default is 32 blocks. If the job is run in batch mode, SGO defaults to 32 blocks even if a size is specified. The actual number of blocks allocated may be greater than the number specified depending on the allocation unit of the disc. If EOM is encountered, the file is extended automatically.

Notes:

When the job is activated, the job sequence number, owner name, and job name are listed on the operator's console.

When a $JOB directive has been used, an $EOJ directive must be used before another $JOB directive can be specified. Two $JOB directives without an intervening $EOJ directive causes an error message and end-of-job (EOJ) processing.

When the using the $BATCH or the $SUBMIT directive, the $JOB directive must be the first card read by J.SSIN or J.TSM, unless the $DEFM directive is specified. If $DEFM is specified, then $JOB must be the second card in the directive file.

Fields following the owner name field can be entered in any order.

When a user logs on, the owner name is verified in the M.KEY file, thereby establishing the operating environment, such as key, if applicable, access restrictions, project group, working directory, etc. The following exceptions apply to the $JOB directive:

- An owner name and key, if applicable, do not have to be specified. The submitter's owner name is used by default.
- For SLOF/SBOF, the current working directory is assumed in the interactive mode. Only a one- to eight-character file name can be specified. For batch jobs, the job owner's directory defined in the M.KEY file is used. If an M.KEY file does not exist, the system directory is assumed by default.

At EOJ processing, the command processor executes a run request to the system output executive J.SOEX. J.SOEX directs data from the SLO or SBO files to their destination peripheral device based on the job statement options SLOF/SBOF and SLOD/SBOD as follows:

- No SLOF/SBOF or SLOD/SBOD option specified on the job statement:

  The SLO/SBO file is queued for output on the first available automatically selectable SLO or SBO output device. After output, the SLO/SBO file is deleted.

- SLOF/SBOF specified but not SLOD/SBOD:

  The named permanent file is created under the current working directory. All the SLO/SBO data collected during the execution of the job is written to the permanent SLOF/SBOF file.

- SLOD/SBOD specified but not SLOF/SBOF:

  At end-of-job processing, the collected SLO/SBO temporary file is output to the device specified by the SLOD/SBOD option and then deleted.

- Both SLOF/SBOF and SLOD/SBOD specified:

  A permanent file containing the SLO/SBO is created. At end-of-job, SLO/SBO is also output to the device specified by the SLOD/SBOD option.

Along with specifying a device, the SLOD option has two special case functions:

- If the SLOF option is not specified and SLOD is specified as SLOD=NU, listed output will not be produced. This prevents a hardcopy listing when it is not required.

- If SLOD or SBOD are specified to a magnetic tape device, the magnetic tape is created as multivolume with a default volume name of SLO if it is an SLOD operation, or a default volume name of SBO if it is an SBOD operation.

Usage:

```
$JOB COPYTAPE USER1 SLOF=TESTLIST SLOD=MT1000
```

1.49 $LINESIZE Directive

The $LINESIZE directive dynamically modifies the screen width defined for a terminal at SYSGEN. It is effective for the duration of the job or interactive session. This directive can be used when a task is in a hold state and is valid in interactive mode only.

Syntax:

$LINESIZE maxchars

maxchars specifies the maximum character position to be displayed (written to) on the terminal. Valid range is 40 through 236, inclusively. If not specified, defaults to the number specified with the SYSGEN DEVICE directive. See Section 1.11.1 for linesize effects on reads.

Usage:

```
$LINE 80

$LINE 125
```

1.50 $LIST Directive

The $LIST directive displays the contents of a file. In the interactive mode, output is generated on the terminal. In the batch mode, output is generated on the spooled output file.

Syntax:

$LIST pathname

pathname is the pathname of the file to be displayed

Usage:

$LIST INDEX

$LIST @ATLAS04(USER2)INDEX

1.51 $MOUNT Directive

The $MOUNT directive displays a message on the operator's console requesting the operator to mount a volume on a device. A mount remains in effect for the duration of the job or interactive session. An implicit DISMOUNT is issued at the end-of-job or end of the interactive session.

If the $MOUNT directive is used to mount a volume as public, the volume is not considered dismountable and remains mounted as long as the system is running. It cannot be dismounted with a DISMOUNT directive.

Syntax:

$MOUNT volname ON devmnc [SYSID=id] [OPTIONS= [PUBLIC] [NOMSG]]

volname is the name of the volume to be mounted

devmnc is the device mnemonic of a configured peripheral device. See Appendix A.

SYSID specifies a three-character identifier required for port identification on multiport volumes only. Must be MP0, MP1...MPF. For compatibility, DP0 or DP1 can be entered; DP0 and DP1 are equivalent to MP0 and MP1.

OPTIONS if PUBLIC is specified, the volume is to be mounted for public use and cannot be dismounted until a reboot is performed (valid only if task has the System Administrator attribute). If PUBLIC is not specified, the default is nonpublic. If NOMSG is specified, a mount message is not displayed on the operator's console and it is assumed the disc drive is ready.

When a batch job is initiated or a user logs on, the default volume name is established from the M.KEY file. If the default volume is not mounted, the user must explicitly mount the volume and change the current working directory.

If the mount request is for a multiprocessor volume, mount messages are not inhibited, and the volume was not previously dismounted, the operator is prompted for volume clean-up for the multiprocessor volume. Reply Y (yes) or N (no). Volume clean-up deletes temporary files and resets multiprocessor access information and all resource descriptor locks. Therefore, volume clean-up should not be performed if the volume is currently mounted on another system.

If mount messages are inhibited, no volume clean-up is performed.

Response:

If the operator denies a $MOUNT request, the command processor takes the following action:

- Batch jobs with implicit or explicit mounts that fail are terminated.
- On-line users are prompted for alternate action. If the default volume cannot be mounted, the system volume is selected as the current default.

If file overlap is detected, the following messages are displayed on the system console and the volume is not mounted:

```
FILE OVERLAP HAS OCCURRED IN RDnum
RD TYPE num
FILENAME IS name
SECTORS num THROUGH num
```

num is a hexadecimal number

name is the one- to sixteen-character file name

To mount the disc so that data can be recovered, set control switch seven.

Usage:

```
$MOUNT ANYVOL ON DM0802
```

Requests the operator to mount volume ANYVOL on device DM0802. If the volume is not already mounted, a mount message is displayed and the volume is mounted as nonpublic.

If an error occurs and the mount is not successful, the reason for the failure is displayed. See the OPCOM MOUNT directive for a list of error messages.

1.52 $NOTE Directive

The $NOTE directive sends a message to the user's terminal. This job is not affected by the TSM NOCOMMAND directive.

For batch jobs, if the owner name associated with the job is not logged on to a terminal, the message is sent to the operator's console. The batch sequence number and owner name are prefixed to the message.

If $NOTE is processed by a batch job and the terminal where the job originated is busy, some messages may not be displayed on the terminal because a buffer was overwritten. A buffer is overwritten when a new message is generated before the previous message was displayed (the TSM message facility has only two buffers for each terminal). The SLO for the batch job contains all the $NOTE messages. If the batch job parent terminal is put into $WAIT after the $SUBMIT or $BATCH directive is processed, the number of displayed messages can be maximized.

In the interactive mode, the verb NOTE is omitted from the actual message.

Syntax:

$NOTE message

message is the message to be displayed. The message is a maximum of 72 characters.

1.53 $OBJECT Directive

The $OBJECT directive serves as a precursor for program object records and is valid from batch directive files only.

The program object deck that follows the directive is stored on the SGO file. More than one deck can be included. The last deck is terminated by the next JCL statement. A $SELECTx directive does not terminate the deck.

Syntax:

$OBJECT

1.54 $OPTION Directive

The $OPTION directive sets options that control various aspects of the user operating environment for tasks running on-line or in batch mode. Options also set flags that can be tested by an executing task.

Options are specified by name or number. Options 1 to 20 are task-dependent; options 21 to 32 are system-defined and available to all tasks. Refer to the MPX-32 Utilities Reference Manual for task-dependent options for the MPX-32 utilities. System-defined options are described below.

Syntax:

```
$OPTION [ n
          DUMP
          CPUONLY
          IPUBIAS
          COMMAND
          NOCOMMAND
          ERROR
          NOERROR
          TEXT
          PROMPT
          LOWER
          QUIET
          UNQUIET
          ABORT
          NOABORT
          RETAIN
          CLEAR
          WRAP
          NOWRAP
          U/C
          L/C ]
```

n is a number from 1 to 32 specifying a particular option available for an MPX-32 utility or user task

PROMPT (option 21) issues an automatic prompt before a read from the terminal. The prompt is preceded by a carriage return/line feed and consists of the first three characters of the task name or utility being run. This option should be used only if the task does not write the prompt itself. This option is not valid in batch mode.

LOWER (option 22) inhibits automatic conversion of lower case characters to upper case. File names must be entered in upper case. This option is not valid in batch mode.

TEXT (option 23) echoes text to the terminal (interactive mode) or SLO file (batch mode) as it is read from the SYC file. This is a one-shot option which applies only to the next executed task.

DUMP (option 24) dumps the task's area of memory to the SLO file if an abort occurs. The SLO file is printed on the LOD device.

CPUONLY (option 25) specifies that the task execute only on the CPU. If not specified, the default is to execute the task on the first available processor (CPU or IPU).

IPUBIAS	(option 26) specifies that IPU-compatible tasks execute on the IPU
COMMAND	echoes JCL directives to the terminal (interactive mode) or SLO file (batch mode) as they are read from the SYC file. This option is the default if not overridden by the NOCOMMAND option.
NOCOMMAND	(option 27) inhibits echoing JCL directives as they are read from the SYC file. This option remains in effect until reset by the COMMAND option.
ERROR	displays a description of an abort code at the terminal (interactive mode) or on the SLO file (batch mode) when an abort occurs. This option is the default if not overridden by the NOERROR option.
NOERROR	(option 28) inhibits the display of abort code descriptions if an abort occurs
QUIET	inhibits messages from being displayed asynchronously on a full duplex terminal. This option has no effect when specified from the console. Default.
UNQUIET	causes messages to be displayed asynchronously on a full duplex terminal. Default for the console only.
ABORT	causes abort messages to be displayed to the terminal, console, or SLO device in the interactive, batch or real-time environments. Default.
NOABORT	(option 29) inhibits the display of abort messages during a job or interactive session. Any tasks executed or activated while this option is in effect have OPTION 29 set. OPTION 29 can be specified instead of option NOABORT to inhibit abort message displays for the next executed or activated task only.
RETAIN	(option 30) forces any cataloged options to be ORed with any user-supplied options during task activation.
CLEAR	(option 31) forces all cataloged options from 1 to 20 to be cleared during task activation.
WRAP	enables terminal line wrap. Not valid in the batch mode. Default.
NOWRAP	disables terminal line wrap. Not valid in the batch mode.
U/C	converts all directive line input to upper case. Default.
L/C	allows characters to be read as entered on a directive line.

Options ERROR, NOERROR, COMMAND, NOCOMMAND, QUIET, UNQUIET, WRAP, NOWRAP, U/C, and L/C do not have a valid numeric equivalent as they are local to TSM only and do not get propagated to the task option word at task activation.

If no parameters are specified, all previous options are cleared.

Usage:

```
$OPTION COMMAND
$OPTION 17 20
```

1.55 $PAGESIZE Directive

The $PAGESIZE directive specifies the number of consecutive output records (lines) to write at the terminal without an intervening read, or enter CR FOR MORE message. It dynamically modifies the page size specified at SYSGEN. It remains effective for the duration of the job or interactive session. This directive can be used when a task is in a hold state and is valid in interactive mode only.

A page size of zero can be specified to output lines without intervening CR messages. Do not specify zero if a task does not have a break receiver because output cannot be stopped with the break key.

Syntax:

$PAGESIZE [maxlines]

maxlines specifies the maximum number of lines (0 to 254) to display (write) before another read (or CR) from the terminal. If not specified, the default is the number specified with the SYSGEN DEVICE directive.

Usage:

$PAGE 0

$PAGE 125

1.56 $PRINT Directive

The $PRINT directive submits a file to the output spooler. The file must be blocked. This directive can be used when a task is in a hold state.

Syntax:

$PRINT pathname [DEVICE=devmnc] [COPIES=number] [FORMAT= {Y / N}]

pathname specifies the pathname of the file to be printed

devmnc is a device mnemonic (see Appendix A). If not specified, the output defaults to the autoselectable device defined with the SYSGEN DEVICE directive.

number is a decimal number specifying the number of copies to be printed. If not specified, the default is one.

FORMAT if Y (yes) is specified, the character in column one is interpreted as a carriage control character. If N (no) is specified, the character in column one is interpreted as data. If nothing is specified, the character in column one is interpreted as a carriage control character for SLOF files and as data for all other operations.

Usage:

```
$PRINT INDEX COPI=2
```

Prints two copies of file INDEX from the current working volume and directory.

```
$PRINT @ATLAS04(USER)INDEX  FORM=N
```

Prints one copy of file INDEX from volume ATLAS04 in directory USER. Characters in column one are interpreted as data.

1.57 $REMOVE Directive

The $REMOVE directive terminates any further processing of the specified job. The job does not have to be active. If the job is active, any task executing within the job is aborted. Output that is already spooled to an SLO or SBO file is processed normally. This directive can be used when a task is in a hold state and is valid in interactive mode only.

Syntax:

$REMOVE jobno

jobno is the job's sequence number (1-9999)

Response:

LF (line feed)

Usage:

$REMOVE 700

Removes job number 700 from the system.

1.58 $RENAME Directive

THE $RENAME directive changes the file name of a permanent file.

Syntax:

$RENAME pathname1 pathname2

pathname1 is the pathname of the file to be renamed

pathname2 is the pathname of the new file name

Response:

The following error message is generated if the file to be renamed does not exist, or if the new file name is currently in use:

UNABLE TO RENAME FILE

If a volume name is specified, it must be the same for both pathnames. To change the volume name, use the Volume Manager (VOLMGR) utility.

Usage:

$RENA TEST1 LAB1

Renames the file named TEST1 on the current working volume and directory to LAB1.

$RENA @TB00(USER1)TEST1 @TB00(USER2)LAB1

Renames the file named TEST1 on volume TB00 in directory USER1 to a file named LAB1 on volume TB00 in directory USER2.

1.59 $RESETF Directive

The $RESETF directive sets a false (=0) condition for up to 32 distinct flags. The flags can then be tested by $IFT and/or $IFF directives within a directive file. $RESETF should precede any $IFF or $IFT directives that check the condition of the specified flags. The $RESET directive remains in effect for the duration of the job or interactive session. In an interactive session, flags are reset only at logon time. They are not reset when a $JOB or $EOJ command is specified.

This directive can be used when a task is in a hold state and is valid from interactive and batch directive files.

Syntax:

$RESETF flagno [flagno] ...

flagno specifies a flag number in the range 0 to 31. Any number of flags may be reset with a single $RESETF directive. Flags remain reset to false within a directive stream until they are set to true with a subsequent $SETF directive.

Usage:

```
$RESETF 3 7 9
```

1.60 $RUN Directive

The $RUN directive activates a task in the interactive or batch environment.

Syntax:

$RUN pathname

pathname is the pathname of a load module or executable image file to be activated

1.61 $SELECT Directive

The $SELECT directive reads directives from a file rather than the terminal. The file must be in blocked and uncompressed format. The file must be in the directory in effect when the $SELECT directive is issued.

A $SELECT directive can be used to select another directive file from the current directive file. Processing continues at the beginning of the newly selected file. If there are any directives following $SELECT on the first file, they are skipped. Any number of directive files can be chained by $SELECT directives.

This directive is valid from terminals and in interactive mode only.

Syntax:

$SELECT pathname [par1] [par2] ...

pathname is the pathname of a permanent file containing directives

par1 is a parameter to pass for the directive file. Up to eight parameters can be passed.

The directive verb $SELECT is optional for initial directive file selection only. All directive file chaining (if used) beyond the initial directive file must use the $SELECT directive. $SELECT can be used to avoid any ambiguity about whether a task or a directive file is to be executed.

Response:

In the interactive environment, directives that are read from the file are echoed to the terminal and executed immediately. When a $RUN or equivalent directive activates a task from the directive file, the command processor activates the task. The task is then run interactively, typically with input from the user at the terminal or from the directive file. When the task exits, the command processor reads the next directive from the directive file, if any. Directive file processing terminates at the end-of-file, when a $CLEAR directive is issued, or when an $ENDM directive is encountered.

Errors:

If an invalid directive is detected in the directive file, the command processor prompts the user for interactive input. The user can correct the error or enter any valid directive.

Entering a carriage return when the command processor encounters an error on the directive file skips the invalid directive and executes the next directive in the directive file.

If the $SELECT directive is not specified when chaining to another directive file from the current directive file, the message UNRECOGNIZED COMMAND is displayed. Correct the error or clear the macro.

1.62 $SELECTD Directive

The $SELECTD directive spools batch records from a peripheral device to the SYC file. This directive is valid in batch directive files only.

Syntax:

$SELECTD devmnc reel [mode] [density] [parity] [NORE]

[UNBLOCKED] [fs fr]

devmnc — is the six-character device mnemonic and address of a card reader, paper tape reader, or magnetic tape device. The entry consists of a two-character device mnemonic (see Appendix A) followed by a four-character hexadecimal device address, such as CR7800. The device address consists of a two-character device channel number followed by a two-character device subaddress. If the subaddress is omitted, zero is assumed. If the entire device address is omitted, a channel number and subaddress of zero are assumed. If the device is the primary or an alternate system input source at a previous level, the following fields are not interpreted.

The following fields are applicable only for magnetic tape devices:

reel — is a four-character tape reel identifier

The following three fields are applicable only for 7-track magnetic tape devices:

mode — specifies the 7-track magnetic tape format:

I = interchange (BCD) (default)
P = packed (binary)

density — specifies the 7-track magnetic tape density:

H = 800 BPI (default)
L = 556 BPI

parity — specifies the 7-track magnetic tape parity

E = even parity (default)
O = odd parity

NORE — inhibits magnetic tape rewind. If this parameter is omitted, the magnetic tape is rewound before and after being read.

UNBLOCKED — specifies that the magnetic tape is unblocked, i.e., contains one logical record per physical record. If this parameter is omitted, blocked is assumed.

fs — is the number of files to be skipped prior to reading the magnetic tape. If this parameter is omitted, no files are skipped. If this parameter is specified, the fr parameter must also be specified.

fr — is the number of files to be read from the magnetic tape. If this parameter is omitted, one file is read. EOF marks read from the tape are not written to the SYC file. If this parameter is specified, the fs parameter must also be specified.

1.63 $SELECTF Directive

The $SELECTF directive spools batch records from a permanent disc file to the SYC file. This directive is valid in batch directive files only.

Syntax:

$SELECTF pathname [UNBLOCKED] [fs fr] [password]

pathname — is the pathname of a permanent file. If a directory name is not specified in the pathname, the default is the directory associated with the owner name on the most recent $JOB statement. If neither has been specified, the default is the directory associated with the owner name at the time the $SELECTF directive is processed. If the owner name currently in effect is key-protected, the default is the system directory. If the file is the primary or alternate system input source at a previous level, the following fields are not interpreted.

UNBLOCKED — specifies the disc file was written in the unblocked mode. If this parameter is omitted, the default is blocked.

The user program is responsible for defining and testing end conditions since there is no hardware EOF on a disc file. The operating system does not have an EOF to test or detect.

fs — is the number of files to be skipped prior to batch input data records. If this parameter is omitted, no files are skipped. If the UNBLOCKED option is requested, fs must not be specified or must be equal to zero. If the fs parameter is specified, the fr parameter must also be specified.

fr — is the number of files to be read. If this parameter is omitted, one file is read. EOF marks are not copied to the SYC file. If the UNBLOCKED option is requested, the fr parameter must not be specified or must be equal to one. If the fr parameter is specified, the fs parameter must also be specified.

password — is ignored

1.64 $SELECTLD Directive

The $SELECTLD directive spools batch records in library format from a peripheral device to the SYC file. Data in library format is created by the Source Update (UPDATE) utility.

This directive is valid in batch directive files only.

Syntax:

$SELECTLD devmnc reel [mode] [density] [parity]

[NORE] [UNBLOCKED] header

devmnc is the six-character device mnemonic and address of a card reader, paper tape reader, or magnetic tape device. The entry consists of a two character device mnemonic (see Appendix A) followed by a four-character hexadecimal device address, such as CR7800. The device address consists of a two-character device channel number followed by a two-character device subaddress. If the subaddress is omitted, zero is assumed. If the entire device address is omitted, a channel number and subaddress of zero are assumed. If the device is the primary or an alternate system input source at a previous level, the following fields are not interpreted.

The following fields (through UNBLOCKED) are applicable only for magnetic tape devices:

reel is a four-character tape reel identifier. Required for magnetic tape.

The following three fields are applicable only for 7-track magnetic tape devices:

mode specifies the 7-track magnetic tape format:

I = interchange (BCD) (default)
P = packed (binary)

density specifies the 7-track magnetic tape density:

H = 800 BPI (default)
L = 556 BPI

parity specifies the 7-track magnetic tape parity:

E = even parity (default)
O = odd parity

NORE inhibits magnetic tape rewind. If this parameter is omitted, the magnetic tape is rewound before and after being read.

UNBLOCKED specifies the magnetic tape is unblocked. For example, contains one logical record per physical record. If not specified, the default is blocked.

header is the one- to eight-character name which appears on the header record where the device is to be positioned prior to reading. The device is positioned to this header record and one file is copied to the SYC file.

1.65 $SELECTLF Directive

The $SELECTLF directive spools batch records in library format from a permanent disc file to the SYC file. Library-formatted disc files are created by the Source Update (UPDATE) utility.

This directive is valid in batch directive files only.

Syntax:

$SELECTLF pathname [UNBLOCKED] header [password]

pathname — is the pathname of a permanent file. If a directory name is not specified in the pathname, the default is the directory associated with the owner name on the most recent $JOB statement. If neither has been specified, the default is the directory associated with the owner name at the time the $SELECTLF directive is processed. If the owner name currently in effect is key-protected, the default is the system directory. If the file is the primary or an alternate system input source at a previous level, the following fields are not interpreted.

UNBLOCKED — specifies the disc file was written in the unblocked mode. If this parameter is omitted, the default is blocked.

header — is the one- to eight-character name which appears on the header record where the file is to be positioned prior to reading. The file is positioned to this header record and one file is copied to the SYC file.

password — is ignored

1.66 $SELECTS Directive

The $SELECTS directive resets all alternate system input source levels established by previous $SELECTx directives. Reading of batch stream data reverts to the primary system input source. This directive has no effect when read from the primary system input source.

Syntax:

$SELECTS

1.67 $SET Directive

The $SET directive associates a new value with a dummy argument within a macro. This directive remains in effect for the duration of the job or interactive session and is valid from interactive and batch directive files. In an interactive session, flags are reset only at logon time. They are not reset when a $JOB or $EOJ directive is specified.

Syntax:

```
$SET  %par [value]
```

par — is one of the parameters defined by the associated $DEFM directive. An error message is generated if the parameter does not match a $DEFM parameter.

value — is a 1- to 16-character string that is assigned to the dummy parameter. This value overrides the default value from the $DEFM and any value passed by the $SELECT directive. If the value is not specified, the value reverts to the default value in the $DEFM. In the interactive environment, a missing value issues a read to the terminal to obtain a user-specified value. This option should be used with the $NOTE directive because a prompt is not issued. Special TSM characters should not be used in the character string because M.TSCAN attempts to parse them and an incomplete substitution is performed.

Usage:

```
$DEFM PRE,DIR
$IFP %PRE CKDIR
$SET %PRE J
$DEFN CKDIR
$IFP %DIR CKVOL
$SET %DIR SOURCE
      .
      .
      .
```

1.68 $SETF Directive

The $SETF directive specifies a true (=1) condition for up to 32 distinct flags. The flags can then be tested by $IFT and/or $IFF directives within a directive file. $SETF should precede any $IFF or $IFT directives that check the condition of the specified flags.

This directive is effective for the duration of the job or interactive session.

$SETF can be used when a task is in a hold state and is valid from interactive and batch directive files.

Syntax:

```
$SETF flagno [flagno] ....
```

flagno — specifies a flag number in the range 0 to 31. Any number of flags can be set to true within a single $SETF directive. Flags remain set within a directive job stream until they are reset to false with a subsequent $RESETF directive.

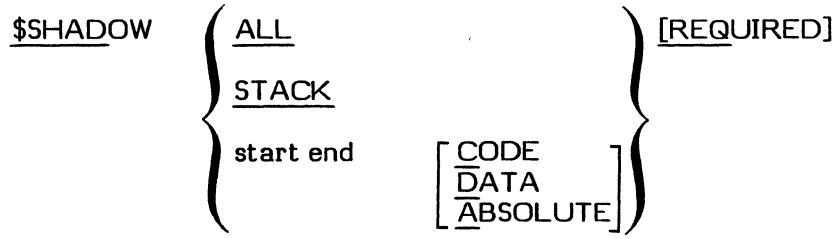
Usage:

```
$SETF 3 7 9
```

1.69 $SHADOW Directive

The $SHADOW directive specifies the portions of a task's logical address space to be located in shadow memory. The portions to be shadowed are specified using relative or absolute addresses. Multiple $SHADOW directives have a cumulative effect and allow noncontiguous portions of the task to be shadowed. The $SHADOW directive does not allocate memory to a task. The $SHADOW directive controls the class of the memory allocated to a task during task activation, during the first inclusion of a shared image, and during the usage of dynamic memory allocation services.

Syntax:



start — is the hexadecimal logical starting address of the logical address space to be shadowed

end — is the hexadecimal logical ending address of the logical address to be shadowed

CODE — specifies that the start and end addresses are relative to the task's code section origin as opposed to the default assumption that the addresses are relative to the task's start.

DATA — specifies that the start and end addresses are relative to the task's data section origin as opposed to the default assumption that the addresses are relative to the task's start.

ABSOLUTE — specifies that the start and end addresses are absolute; the addresses are relative to logical address zero.

ALL — specifies that the entire task is to be shadowed (except the TSA)

STACK — specifies that the task's stack is to be shadowed. This option applies only to base mode tasks and is ignored for nonbase mode tasks.

REQUIRED — specifies that shadow memory is required by the logical address space and the task can wait until shadow memory is available. If REQUIRED is not used, and if shadow memory is not available, then E- or S-class memory is allocated to the logical address space.

Notes:

MPX-32 does not support shadowing of the Task Service Area (TSA).

The $CLEAR directive clears $SHADOW directives.

1.70 $SHOW Directive

The $SHOW directive displays elapsed CPU execution time, lists jobs waiting or active in the system, or lists the names of all logged on terminal users. This directive is used when a task is in a hold state.

Syntax:

$SHOW [CPUTIME | JOBS | USERS]

CPUTIME displays the total CPU execution time of all interactive tasks activated during the current session for the owner issuing the directive

JOBS displays all jobs waiting or active in the system

USERS displays terminal addresses, owner names, task numbers, project names, volume names, and directory names currently in effect for all logged on terminal users

If no parameters are specified, the terminal address (or job number if in batch mode), owner name, task number, project name, volume, and directory for the owner name issuing the directive are displayed.

Response:

In response to $SHOW CPUT, the following message is displayed:

CPU EXECUTION TIME = xx HOURS- xx MINUTES- xx.xx SECONDS

xx is a decimal number indicating units of CPU execution time

In response to $SHOW JOBS, the display format is:

jobno ownername jobname priority taskno taskname

jobno is the job sequence number in the range 1 to 9999

ownername is the owner name for the job as specified on the $JOB statement

jobname is the job name as specified on the $JOB statement if the job was executed by $BATCH. If the job was executed by $SUBMIT, the first eight characters of the pathname are displayed; after the job begins actual processing, its job name is displayed.

priority is the job's software priority in the range 1 to 64

taskno is the task number of the last task in the job that was activated or is queued

taskname is the name of the task

In response to $SHOW USER, the display format is:

address/job ownername taskname project volume directory

address/job — is the terminal address of active terminals. An asterisk is displayed before the mnemonic of the terminal or batch context issuing the directive. If preceded by "JB", the job number of an active batch context is displayed.

ownername — is the owner name used to logon

taskname — is the name of the load module in use by the owner name

project — is the project name in effect for the owner name for file access

volume — is the volume name in effect for the owner name for file access

directory — is the directory name in effect for the owner name for file access

Usage:

```
TSM>$SHOW CPUT
CPU EXECUTION TIME = 00 HOURS-  07 MINUTES- 03.92 SECONDS

TSM>$SHOW JOBS
0004  FORTRAN        JOBTEST     62     0E00000C     ASSEMBLE
0005  GUEST          JOBFILE     62     QUEUED
TSM>$SHOW JOBS
0005  GUEST          JOBNAME     62     0F00000D     CATALOG

TSM>$SHOW USER
ADDRESS OWNERNAME TASKNAME   PROJECT VOLUME       DIRECTORY
======= ========= ========   ======= ===========  ==============
*TY7EFC GUEST     (WAIT)     EXAMPLE DM0800       M27X
 TY7EA0 SYSTEM    (OPCOM)    SYSTEM  USER1        SYSTEM
 JB0004 FORTRAN   (ASSEMBLE) SYSTEM  USER1        FORTRAN

TSM>$SHOW
ADDRESS OWNERNAME TASKNAME   PROJECT VOLUME       DIRECTORY
======= ========= ========   ======= ===========  ==============
*TY7EFC GUEST     (TSM)      EXAMPLE DM0800       M27X
```

In batch mode:

```
$SHOW USERS
ADDRESS OWNERNAME TASKNAME   PROJECT VOLUME       DIRECTORY
======= ========= ========   ======= ===========  ==============
 TY7EFC GUEST     (WAIT)     EXAMPLE DM0800       M27X
 TY7EA0 SYSTEM    OPCOM      SYSTEM  DM0800       SYSTEM
*JB0007 SYSTEM    (TSM)      SYSTEM  DM0800       SYSTEM
```

1.71 $SIGNAL Directive

The $SIGNAL directive sends a message to another logged on owner or to all terminals. This directive is used when a task is in a hold state.

Batch users can signal interactive users but cannot receive messages.

Syntax:

$SIGNAL [ownername]

ownername is the owner name of another logged on user. If the owner is not logged on, the message is denied. If an owner name is not specified, the message is sent to all logged on terminals. The message is a maximum of 80 characters and is terminated by a carriage return.

Response:

In response to $SIGNAL, the following message is displayed:

ENTER MESSAGE

When invoked from a directive file, the prompt is not issued and the line following the directive is interpreted as the message.

1.72 $SPACE Directive

The $SPACE directive expands a task's logical address space. This directive is ignored on the 32/27 and 32/87 computers. This directive is also ignored if the task is a shared task.

Syntax:

$SPACE size

size is the expanded logical address space in megabytes

The default logical address space for nonbase mode tasks is 2MB minus the operating system's size. The default logical address space for base mode tasks is either 2MB minus the operating system's size, or the value specified by the LINKER/X32, whichever is greater. If the value specified in the size parameter is greater than a task's default logical address space, that value is used as the logical address space size. If the specified value is smaller than the default, the default is the logical address space size.

1.73 $SUBMIT Directive

The $SUBMIT directive submits a directive file to run in the batch stream. The file must be in blocked and uncompressed format. The command processor searches for the specified file under the current working volume and directory in effect when the $SUBMIT directive is issued. This directive can be used when a task is in a hold state and is valid from interactive and batch directive files. See Section 1.80 for details on batchstream/memory pool interaction.

This directive is similar to the TSM $BATCH directive except for the following restrictions:

- No input spooling or preprocessing is performed. Therefore, the job is not preserved if the system is reIPLed.
- The $SELECTx and $OBJECT directives are not supported within the file.
- The sequential option is assumed if the SEQUENTIAL keyword is specified in the M.KEY file.
- The owner name and the S keyword on the $JOB statement are ignored.
- Only one job is supported per $SUBMIT directive; for example, one $JOB and one $EOJ pair.

As a result of these restrictions, the $SUBMIT directive has less overhead than the $BATCH directive.

When using the $SUBMIT directive, the $JOB directive must be the first card in the directive file, unless the $DEFM directive is specified. If $DEFM is specified, then the $JOB must be the second card in the directive file.

Syntax:

$SUBMIT pathname [par1] [par2]...

pathname is the pathname of a permanent file containing TSM directives

par1 is a parameter to pass for the directive file. Up to eight parameters are passed.

Response:

The named file is enqueued. When selected by the command processor, the job number is displayed:

```
jobno ownername $JOB jobname
```

Usage:

```
TSM>$SUBM JOBFILE
YOUR JOB NUMBER IS 2952
```

1.74 $SYSOUT Directive

The $SYSOUT directive specifies how SLO assignments are interpreted in the interactive mode. This directive remains in effect until the user logs off the system or issues another $SYSOUT directive. It is valid in the interactive mode only.

Syntax:

$SYSOUT={UT / SLO}

UT directs all SLO output to the user's terminal. This specification applies to both static and dynamic assignments, and overrides an SLOF or SLOD option specified on a $JOB statement.

SLO specifies assignments to SLO are spooled. If a $JOB statement is in effect, the SLO is spooled to a single extendible file until an $EOJ statement is encountered.

1.75 $URGENT Directive

The $URGENT directive changes the priority of a batch job. The job must be queued or active. This directive is used when a task is in a hold state and is valid in interactive mode only.

This directive overrides any active jobs designated as sequential on the $JOB statement (must proceed to $EOJ before beginning another job). It is used to boost a task in a job to a priority higher than 64.

Syntax:

$URGENT jobno,priority

jobno is the job's sequence number in the range 1 to 9999

priority is the software priority at which the job is to run in the range 55 to 64

Response:

LF (line feed)

Usage:

$URGENT 700,60

Changes the priority of job number 700 to priority 60.

1.76 $USERNAME Directive

The $USERNAME directive changes the current working directory to gain access to another directory. This directive does not change the current working volume.

Syntax:

$USERNAME [dirname]

dirname is the one- to eight-character name of an existing directory on the current working volume. If $USERNAME is specified without supplying a directory name, all files created are system files and only system files on the current working volume are accessed.

Usage:

$USER SOURCE

is equivalent to: $CHANGE DIRE=^SOURCE

1.77 $WAIT Directive

The $WAIT directive puts a terminal into a special wait state so that it receives messages. For example, $WAIT is used to receive messages pertaining to batch jobs that are submitted prior to continuing terminal operation or logging off. Then, messages output by job control are displayed as the job is processed. The user continues to wait for other messages or exits the wait state by pressing the wake-up character to return to normal interactive operation.

If $WAIT is not used, an inactive terminal is in a read condition and a read is not interrupted by messages unless the terminal times out. Carriage returns are issued to get out of a read condition and display a message.

There is no overhead associated with a terminal in the wait state; it is the same as being logged off.

This directive can be used when a task is in a hold state and is valid in interactive mode only.

Terminals in a wait or logged off state are not subject to screen control; information is displayed as if PAGESIZE=0.

Syntax:

$WAIT

Usage:

```
        EDT>COL
        1. $JOB X OWNER1,G SLOF=SLOF1
        3. $EXECUTE FILEMGR
        4. LOGU
        5. $EOJ
        6. $$
1       EDT>BATCH
        EDT>EXIT
        TSM>$WAIT
2       0002 OWNER1   $JOB X
        0002 OWNER1   $EOJ  X
3       <wakeup>
        TSM>
```

1 The user submits the work file displayed above.

2 The job number, owner name, and job name are displayed upon initiation. When the job is complete (successfully or with an abort), the batch end-of-job message is displayed. The form of the message is:

jobno owner {$JOB | $EOJ} jobname

This job is number 0002, it belongs to OWNER1, and its job name is X.

3 At this point, the terminal is still in the ANYW state. The user continues waiting to receive messages from other jobs that have completed or uses the wake-up character to enable TSM input.

1.78 $WHO Directive

The $WHO directive displays the current terminal addresses, ownernames, task names, project names, volume names and directory names currently in effect for all logged on users.

Syntax:

```
$WHO
```

Usage:

```
TSM>$WHO
ADDRESS OWNERNAME TASKNAME  PROJECT VOLUME      DIRECTORY
======= ========= ========  ======= ==========  ==========
*TY7E00 GUEST     (WAIT)    EXAMPLE DM0800      M27X
 TY2001 SYSTEM    OPCOM)    SYSTEM  USER1       SYSTEM
```

1.79 $$

The $$ directive terminates a batch input stream. It follows the $EOJ directive on the last job in the batch stream. When the $$ directive is encountered, the system input task terminates processing of batch directives. This directive is ignored if read from a primary system input source device and if continuous batch mode has been requested by Operator Communications (OPCOM).

This directive is valid in batch directive files only.

Syntax:

```
$$
```

1.80 $$$

The $$$ directive terminates a batch input stream when the OPCOM continuous batch mode is in effect. If continuous batch mode has not been requested, this directive is treated as a $$ directive.

This directive is valid in batch directive files only.

Syntax:

```
$$$
```

1.81 Examples

The following sections contain examples of task processing in the batch and interactive modes, conditional processing, and setting up directive files.

1.81.1 Sample Interactive Task

```
$JOB EXAMPLE USER
$OPTION 2 5
$ASSEMBLE
          PROGRAM     ECHO
          M.EQUS
          LIST        NOMA,NORE,NODA
*
**        THIS PROGRAM WILL ECHO DATA INPUT FROM THE TERMINAL
*
ECHO      LI          R1,IBUFL          LOAD INPUT BUFFER LENGTH
          TRN         R1,R1             NEGATE IT TO CONTROL LOOP
          LI          R4,G' '           LOAD A BLANK
CLEAR     STB         R4,IBUF+IBUFL,X1  BLANK THE CURRENT BYTE
          BIB         R1,CLEAR          FOR THE WHOLE BUFFER
          M.READ      IFCB              READ INTO BUFFER
          TBM         7,IFCB+3W         EOM ?
          BS          ENDIT             YES,QUIT
          TBM         6,IFCB+3W         EOF ?
          BS          ENDIT             YES, QUIT
          LI          R4,G'X'           LOAD AN X
          CAMB        R4,IBUF           DID THE USER ENTER ONE ?
          BEQ         ENDIT             YES, QUIT
          M.WRIT      OFCB              OUTPUT THE BUFFER
          TBM         7,OFCB+3W         EOM?
          BS          ENDIT             YES,QUIT
          TBM         6,OFCB+3W         EOF ?
          BS          ENDIT             YES, QUIT
          BU          ECHO              GO GET A NEW LINE
ENDIT     M.EXIT                        EXIT
          BOUND       1W
IFCB      DATAW       G'IN '            INPUT FCB
          GEN         12/IBUFL,20/B(IBUF)
          REZ         6W
OFCB      DATAW       G'OUT'            OUTPUT FCB
          GEN         12/OBUFL,20/B(OBUF)
          REZ         6W
OBUF      DATAB       C' '              OUTPUT BUFFER
IBUF      REZ         80B               INPUT BUFFER
OBUFL     EQU         $-OBUF            OUTPUT BUFFER LENGTH
IBUFL     EQU         $-IBUF            INPUT BUFFER LENGTH
          END         ECHO
$CATALOG
OPTION PROMPT LOWER
AS IN TO LFC=UT
AS OUT TO LFC=UT
BUILD ECHO
$EOJ
$$
```

1.81.2 Terminal Session

The following example demonstrates the interaction between the user and the system during an interactive terminal session.

```
?                                   Wake-up character defined in LOGONFLE is a ?

MPX-32 RELEASE 3.3 .01 TBDSYS99 TERMINAL SERVICES MANAGER
(C) COPYRIGHT 1983, GOULD INC., CSD, ALL RIGHTS RESERVED
ENTER YOUR OWNERNAME:SMITH
ENTER KEY:
################

TSM>ASSIGN 5 TO LFC=UT              LFC 5 is this terminal.
TSM>AS 6 TO LFC=UT                  LFC 6 is also this terminal.
TSM>OPTION PROMPT                   Generate prompt for input.
TSM>EXECUTE ANYTASK                 Run the task.

HELLO, I AM ANYTASK,                Task outputs task-generated message.
WHAT CAN I DO FOR YOU

ANY>LIST                            Prompt is issued before read.

(task lists 24 lines)

ENTER CR FOR MORE<CR>               End-of-screen is reached; user presses CR.

(program resumes listing)

<break>                             Task is in infinite loop; user issues break.

** BREAK ** ON:ANYTASK  AT:2003BC44  CPU TIME = 1.02 SEC.
CONTINUE, ABORT, DEBUG, OR HOLD? A
ANYTASK  #02000001 ABORTED.  PSW:2003BC44 BIAS:34000 REASON: TS01
USER REQUESTED REMOVAL FROM A BREAK REQUEST

TSM>CLEAR

TSM>OPCOM

?? ACTIVATE REALTIME

?? EXIT

TSM>EXIT

CPU EXECUTION TIME  =  00 HOURS-  33 MINUTES-  00.40 SEC
TOTAL CONNECT TIME  =  01 HOURS-  50 MINUTES-  16.32 SEC
RING IN FOR SERVICE
```

1.81.3 Batch Job Examples

Example 1

The following example shows a FORTRAN compilation:

```
$JOB EXAMP1 USERA
$OPTION 2                    Inhibits punched object output.
$OPTION 3                    Inhibits printing of storage dictionary.
$EXECUTE FORTRAN
(Source)
$EOJ
```

Example 2

This example illustrates program assembly from magnetic tape:

```
$JOB EXAMP2 USERB SLOF=LOFILE   All listed output from job is directed to
                                user file LOFILE.
$AS SI TO DEV=M9 ID=SRCE        Overrides cataloged assignment. (Assem-
                                bler reads source from SYC file.)
$EXECUTE ASSEMBLE
$EOJ
```

Example 3

This example catalogs and executes a load module:

```
$JOB EXAMP3 USERC               Establishes owner name USERC for the
                                job and for files allocated during job.
$CHANGE DIRE=ABC                Overrides USERC to ABC for file access.
$OPTION 5                       Outputs assembled program to SGO file.
$EXECUTE ASSEMBLE
(Source)
$EXECUTE CATALOG
CATALOG FILX U 64               Catalogs a load module named FILX.
$ASSIGN AB TO FILAB
$EXECUTE FILX                   Executes the cataloged program.
$EOJ
```

Example 4

In the following example, the $SELECTx statement obtains a source program from a magnetic tape:

```
$JOB EXAMP4 USERD
$EXECUTE FORTRAN
$SELECTD MT10 SRCE 1 1          Source program obtained from second file
                                of a blocked magnetic tape.
$EOJ
```

1.81.4 Conditional Batch Processing

Example 1

The $IFT directive specifies that if file ABC exists, suspend processing through the $DEFNAME statement and do not execute FILEMGR.

```
$JOB EXAMP5 USERF
$IFT FILE ABC FILEPR
$EXECUTE FILEMGR
CREATE ABC,DC,100
$DEFNAME FILEPR
$EOJ
```

Example 2

The $IFT directive specifies that if the UPDATE run is aborted, do not execute ASSEMBLE.

```
$JOB EXAMP6 USERG
$ASSIGN SI1 TO SSS
$ASSIGN SO TO CCI
$OPTION 1 2
$EXECUTE UPDATE
(Source update directives)
$IFT ABORT NOASSEM
$ASSIGN SI TO CC1
$EXECUTE ASSEMBLE
$DEFNAME NOASSEM
$EOJ
```

Example 3

In the following example, SI1 is assigned to CC1 and SO is assigned to CC2. Replacing the $SETF 1 with $RESETF 1 causes SI1 to be assigned to CC2 and SO to CC1.

```
$JOB EXAMP7 USERH
$SETF 1
$IFF 1 NOTCC1
$ASSIGN SI1 TO CC1
$ASSIGN SO TO CC2
$DEFNAME NOTCC1
$IFT 1 NOTCC2
$ASSIGN SI1 TO CC2
$ASSIGN SO TO CC1
$DEFNAME NOTCC2
$EXECUTE UPDATE
(Source update directives)
$EOJ
```

1.81.5 Sample Directive Files

Example 1

The following example assembles source code with binary output assigned to SGO, assigns SYC as the input file for assembling, catalogs the load module file LMSRCE, then executes LMSRCE as an interactive task. The name of the directive file shown below is OWNER2.

```
$JOB PRE OWNER2
$OPTION 2 3 4 5
$AS LO TO LFC=UT
ASSEMBLE
START          M.EXIT
               END START
$EXECUTE CATALOG
CATALOG LMSRCE
$LMSRCE
$EOJ
```

OWNER2 is then executed as:

```
TSM>OWNER2
```

Example 2

This directive file uses parameter substitution for assembling, cataloging, and executing a task. Cataloging and execution are treated as conditional parameters in the directive file. The name of the directive file shown below is GEN.

```
1    $DEFM SI,T,OPT
2    $IFA %OPT 1
     $OPTION %OPT
     %1
     $OPTION 3 4
3    $AS SI TO %SI
     $AS BO TO %SI;BO
     $AS LO TO LFC=UT
4    $ASSEMBLE
5    $IFT %T EQ C EXIT
     $AS SGO TO %SI;BO
     $AS SLO TO LFC=UT
     $EXECUTE CATALOG
     CATALOG LM%SI U 60 NOM
6    $IFT %T NE XA EXIT
     LM%SI
     $EXIT
```

GEN is then executed as:

```
TSM>GEN SFIL,XA
```

Expanded Directives are:

```
$OPTION 3 4
$AS SI TO SFIL
$AS BO TO SFILBO
$AS LO TO LFC=UT
$ASSEMBLE
$AS SGO TO SFILBO
$AS SLO TO LFC=UT
$EXECUTE CATALOG
CATALOG LMSFIL U 60 NOM
$LMSFIL
$EXIT
```

1 Parameters are SI (base file name), T (C = do not catalog or execute; XA = catalog and execute, blanks or anything else = catalog but do not execute). OPT is a numeric option in addition to options 3 and 4 which are selected automatically by the directive file.

2 If the OPT parameter is absent, select options 3 and 4 and continue. If not absent, enable specified option plus options 3 and 4.

3 Assign a file named in the first parameter to SI or use SI if the first parameter is not specified. Assign the same file name suffixed by BO for binary output. The file name specified must be the name of a previously created file.

Note: A semicolon is used to concatenate the parameter with a string. Assign listed output to the terminal.

4 Execute the assembler.

5 If a C is entered as the second parameter, exit. If C is not entered as the second parameter, make the SGO and SLO assignments and catalog.

6 If anything other than XA is entered as the second parameter, exit. If XA is entered as the second parameter, execute the task that has been cataloged on LMSFIL.

Example 3

Parts of this example have been used in directive descriptions. The entire sample directive file is:

```
      $DEFM SI,NEW,OP
1     DELETE OB%SI
      CREATE OB%SI
      $AS LO TO LFC=UT
      $AS BO TO OB%SI
      $AS SI TO %SI
2     $IFA %OP ASSM
      $OPTION %OP
      %ASSM
      $OPTION 3 4
      $ASSEMBLE
```

```
3    $IFT %NEW NE CREATE OLD
     DELETE DIRFILE
     DELETE LIBFILE
     CREATE DIRFILE
     CREATE LIBFILE
     $OPTION 1
     %OLD
     $AS DIR TO DIRFILE BLOC=N
     $AS LIB TO LIBFILE BLOC=N
     $AS LGO TO OB%SI
4    $LIBED
     $EOJ
```

1 The directive file unconditionally deletes and recreates the file to be used for the object output. The assigns are made for the assembly.

2 In addition to setting options 3 and 4 for the assembly, user-specified options are also passed as parameters.

3 The files for LIBED are deleted and recreated and the option set to rebuild the subroutine library if NEW is requested.

4 The subroutine library is created or updated determined by NEW.

Example 4

This directive file is an example of patch application using the task debugger in a transparent manner.

```
$SYSOUT UT                  !Directs listed output to terminal
$AS CMD TO LFC=UT           !VOLMGR takes its directives from the terminal
$AS TAP TO DEV=M9 BLOC=N    !Magnetic tape is not blocked
$AS #OT TO DEV=NU           !No debugger output requested
$AS #IN TO SYC              !Debugger input is the directive file
$DEBUG VOLMGR               !Start up VOLMGR under debugger control
BA $SDT,$DSS+10708
CM $SDT+328=0F88453DF
DETACH
```

An exclamation point (!) in a line beyond the first field is used as a comment delimiter (the remainder of the line is ignored).

If DEV=NU is not specified, debugger output defaults to the terminal.

Debugger directive lines cannot contain comments.

When this directive file is used, a base is established, memory is changed, and the debugger is detached. This process is transparent to the user.

Example 5

This example shows how to set up a directive file for use with the assembler. With the $DEFM directive, one to eight parameters are supplied, and each parameter is one to sixteen characters in length.

```
$DEFM SOURCE OBJECT M.MPXMAC MPXPRE
$NOTE ENTER NEW FILENAMES OR <CR> FOR DEFAULTS
$NOTE SOURCE INPUT (DEFAULT = SOURCE) =>
$SET %SOURCE
$NOTE BINARY OBJECT (DEFAULT = OBJECT) =>
$SET %OBJECT
$NOTE MACRO LIBRARY (DEFAULT = M.MPXMAC) =>
$SET %MPXMAC
$NOTE SOURCE PRE FILE (DEFAULT = MPXPRE) =>
$SET %MPXPRE
$ASSIGN SI TO %SOURCE
$ASSIGN BO TO %OBJECT
$ASSIGN PRE TO %MPXPRE
$ASSIGN MAC TO %MPXMAC BLOCKED=N
$ASSEMBLE
```

This directive file requires user input when used. As the => prompts are displayed, a valid file name or carriage return must be entered. If a carriage return is entered, the default is the file name specified in the corresponding field on the $DEFM directive line. The four $ASSIGN statements automatically reflect the response of the user's input.

1.82 Batch Stream Memory Pool Interaction

By monitoring the availability of the memory pool, the batch stream memory pool interaction facility protects the system from memory pool deadlocks that are caused by excessive $SUBMIT or $BATCH commands. When the memory pool is 75% utilized, the following messages are displayed:

> J.SSIN:**WARNING..<<BATCH QUEUE FULL>>** (This message is displayed on the system console)
>
> *CONTEXT WAITING FOR MEMORY POOL, ENTER WAKEUP TO RETURN TO TSM. (If possible, this message is displayed on the user's terminal; otherwise, it is displayed on the system console.)

These messages warn that efforts are being made to restrict the batchstream from using all of the memory pool. Jobs that are submitted by the $BATCH and $SUBMIT directives suspend the command file where the request originated. The command file is resumed by J.TSM when more than 25% of the memory pool is available. When the required memory pool is available, the following message is displayed on the user's terminal:

> *MEMORY POOL NOW AVAILABLE, CONTINUING COMMAND FILE PROCESSING.

Due to system dynamics such as other tasks, queued jobs, or messages using and freeing the memory pool, this message can be displayed several times before the command is actually processed. It is possible that this message will print before it can be read. See the $NOTE directive.

If the context is not waiting for memory pool, a TSM prompt is displayed on the user's terminal. Enter CR to continue or $CLEAR to terminate.

If the command file is waiting for memory pool, it can be interrupted by entering the wakeup character. The following message is displayed on the user's terminal:

> *COMMAND FILE WAS WAITING FOR MEMORY POOL. VALID COMMANDS ARE:
> CLEAR, CONTINUE, LINESIZE, PAGESIZE, ERR, SHOW, SETF, RESETF, SIGNAL, CHANGE, CREATE, DELETE, PRINT, ACTIVATE, URGENT, REMOVE AND WAIT.
> TSM>

If an invalid command is entered or an invalid operation is attempted, the command file continues processing, and the following message is displayed on the user's terminal:

> *INVALID COMMAND, RETURNING TO COMMAND FILE PROCESSING.

If a command is executed, the last command read from the command file is saved in the logical address space of J.TSM. This allows the linebuffer to process any single-shot commands. When the command is completed, J.TSM returns the saved command to the user's linebuffer. If J.TSM is unable to save the command, an H.MEMM denial is issued, and the following message is displayed on the user's terminal:

> *WARNING, UNABLE TO SAVE CURRENT LINEBUFFER, CURRENT COMMAND FROM COMMAND FILE WILL BE LOST UNLESS <CR> IS ENTERED RATHER THAN A VALID COMMAND.

This allows the user to decide between terminating the command file by entering the valid command $CLEAR or losing the command that interrupted the wait or a $BATCH or $SUBMIT command. If other commands are entered, the original command can be lost or overwritten by the new command in the linebuffer.

Notes:

The batch stream memory pool interaction facility does not keep tasks that can submit batch jobs independently from doing so (except to a limited extent by shutting down J.SSIN until the memory pool is free).

The J.SSIN shutdown is limited in its ability to protect the memory pool due because run requests are sent while it is waiting to use the memory pool.

CHAPTER 2

OPERATOR COMMUNICATIONS (OPCOM)

2.1 Introduction

The Operator Communications service (OPCOM) provides a set of directives for control of system operations from the system console or any TSM terminal.

OPCOM directives are used to:

- Activate and control system and user tasks
- Submit and control batch jobs
- Display system status information
- Control peripherals associated with a batch job, user task, or system task
- Display and access physical memory
- Connect and disconnect tasks to/from interrupts
- Set or delete timers to activate or resume tasks or request interrupts
- Disable and re-enable hardware interrupt levels

2.2 Directive Summary

OPCOM directives are summarized below and described in detail in the following pages. Valid abbreviations are indicated by underlining.

Directive	Function
ABORT	Enters the specified task's abort receiver, if any. If none, deletes the task.
ACTIVATE	Activates a task.
BATCH	Reads batch jobs from a device or file.
BREAK	Enters pseudointerrupt receiver for a specified task.
CONNECT	Connects a task to an indirectly connected interrupt level defined at SYSGEN.
CONTINUE	Releases a task or device from a hold state.

DEBUG	Accesses the System Debugger.
DELETETIMER	Deletes a timer attached to a task.
DEPRINT	Deletes the current SLO file from the system output queue.
DEPUNCH	Same as DEPRINT but for SBO files.
DISABLE	Disables an interrupt at a specified priority level.
DISCONNECT	Disconnects a task from an indirectly connected interrupt level defined at SYSGEN.
DISMOUNT	Allows the operator to dismount a volume from a device.
DUMP	Dumps a specified word location to the SLO file or device.
ENABLE	Enables an interrupt at a specified priority level.
ENTER	Updates the system date and time.
ESTABLISH	Activates and suspends a task.
EXCLUDE	Excludes a resident shared image.
EXIT	Exits OPCOM and returns to the TSM prompt.
HOLD	Inhibits a task from getting CPU control, stops spooled output to a printer, punch, or magnetic tape, or stops spooled input from a card reader or magnetic tape until an operator issues CONTINUE.
INCLUDE	Includes a resident shared image.
KILL	Deletes a task from the system.
LIST	Lists entries in system dispatch queue, output print or punch queue, account file, system patch file, or traps which occurred in the IPU.
MODE	Selects continuous batch mode and inhibits or enables banner page on SLO, mount messages, or operator intervention.
MODIFY	Changes a physical memory word in the operating system address space.
MOUNT	Allows the operator to mount a volume on a device.
OFFLINE	Makes a device unavailable for allocation.
ONLINE	Makes a device available for allocation.
PURGEAC	Deletes the current contents of the accounting file M.ACCNT.
REDIRECT	Redirects SLO or SBO output to a specified device.
REPRINT	Reprints the current SLO file.

REPUNCH	Same as REPRINT but for SBO files.
REQUEST	Generates an RI (request interrupt) instruction for a specified interrupt priority level.
RESUME	Resumes a task.
SEARCH	Searches physical memory for a specified value.
SEND	Sends a message to a task that has established a message receiver.
SETTIMER	Sets a timer for resumption of a task, activation of an established task, or execution of an RI instruction at an interrupt level.
SNAP	Dumps the physical word locations in physical memory to the system console.
STATUS	Lists the amount of memory in current use, channel, device, and I/O queue status information, device type and status information, task attributes and current status, or status of mounted volumes.
SYSASSIGN	Establishes the availability of a device for selection as an SLO and/or SBO destination or temporarily overrides the SYSGEN-selected SID device. Specifies the use of the device for batch and real-time output.
TIME	Prints the time and date on the terminal.
TURNON	Activates a task at a specified time.
UNLOCK	Sends a run request to the dual-processor recovery task.
WAIT	Waits for messages when the terminal would otherwise be inactive. Entering the wakeup character resumes interactive processing.

2.3 Activating OPCOM

To activate OPCOM from a terminal, enter OPCOM or EXECUTE OPCOM at the TSM prompt:

```
TSM> EXECUTE OPCOM
??
```

The OPCOM prompt, a double question mark, indicates that OPCOM is ready to accept directive input. The double question mark prompt is issued automatically by OPCOM after an OPCOM directive has been processed.

Alternatively, an OPCOM directive can be issued at the TSM prompt with immediate return to TSM rather than remaining in OPCOM by preceding the OPCOM directive with an exclamation point:

```
TSM> !directive
```

For example:

```
TSM> !LIST
. . . OPCOM response . . .
TSM>
```

2.4 Restricting OPCOM Directives

Directive usage can be restricted based on owner names and keywords specified in the M.KEY file.

If a directive is restricted through a keyword, the directive is not available to that owner name and generates the message:

```
INVALID COMMAND VERB
```

2.5 System Task Restrictions

System tasks J.SWAPR (the Swapper), J.TSM (Terminal Services Manager), and OPCOM (running on the terminal used to access OPCOM) cannot be activated or controlled by OPCOM directives. The status of these tasks is obtained by using the LIST or STATUS directives.

2.6 System Console

The system console is the terminal where mount, dismount, and I/O error messages are written. It can run OPCOM or any other interactive task. However, the terminal must be shared with the operating system, resulting in system messages being interspersed with OPCOM directives. The system console should not be left idle with an input prompt outstanding. The WAIT directive should be used to free the terminal for incoming system messages. If a time out occurs while OPCOM is waiting for input, an exit occurs.

See the MPX-32 Reference Manual, Volume III, Chapter 10 for further information on system messages.

2.7 Task Names, Task Numbers, and Owner Names

OPCOM directives used to control tasks (ABORT, BREAK, CONNECT, CONTINUE, DEPRINT, DEPUNCH, DISCONNECT, HOLD, KILL, SEND, and RESUME) require unique identification of the task, either by task name, owner name, or task number.

A task number is an eight-digit hexadecimal number assigned to a task by MPX-32 when the task is activated. The task number is unique and identifies a particular copy or sharer of a task.

A task name is the name supplied when a task is cataloged or linked. More than one task with the same name can be active in the MPX-32 system at a time if it is cataloged as multicopied or shared.

The owner name for a task activated from a terminal is the name specified at logon. It cannot be changed except by logging on again. Normally, the logon owner name is associated with any task the owner activates on the system except a task activated by the BATCH directive.

If a load module or executable image can be multicopied or shared, the task name/owner name may not uniquely identify a particular copy of a task because one owner could activate several tasks of the same name.

OPCOM thus restricts any user (terminal or system console) from entering a task name for any task that can be multicopied or shared, and accepts a control directive only if the task number is used. The task number can be obtained by using either the STATUS or LIST directive. Either directive will list all tasks of the specified name with task numbers and other information that allow the user to determine which task number to use.

Control of tasks is further restricted by the MPX-32 KEY utility. If access to tasks with other owner names is restricted, the user who issues an [EXECUTE] OPCOM directive from his terminal can control only those tasks which he has activated. If a user issues a control directive for a task that he does not own, the directive is not accepted by OPCOM.

2.8 Batch Jobs, Job Numbers, and Owner Names

OPCOM directives used to control batch jobs (REDIRECT, REPRINT, REPUNCH, DEPRINT, or DEPUNCH) use job numbers to identify the job. If several jobs are submitted on a job file or device medium, a separate number is supplied for each occurrence of a $JOB job control statement. The job number is a decimal value in a range from 1-9999, and uniquely identifies the job in the system.

If a job is submitted from a terminal and the owner name is changed on the $JOB card, the $JOB owner name applies to the job only. The logon owner name is maintained for the terminal user. To issue OPCOM control directives pertaining to the task(s) activated by the job (see previous section), the restricted user must logoff the terminal and logon again with the owner name used on the $JOB card.

2.9 OPCOM Directives

An OPCOM directive is separated from its parameters by one or more blanks or any valid delimiter (comma, left or right parentheses, equal sign). A comma must be used only where shown in the syntax statement.

Each complete input directive is terminated by pressing the carriage return (CR). OPCOM checks the directive and its parameters. If correct, it completes the operation indicated and acknowledges the directive after processing is complete by returning to the OPCOM or TSM prompt.

An OPCOM directive that is processed to completion issues a CR/LF (carriage return/line feed) and the OPCOM or TSM prompt to acknowledge completion. If a directive is valid but cannot be executed at this time, it issues the message:

```
REQUEST NOT EXECUTED
```

OPCOM always acknowledges a directive with a message or a CR/LF. It then reissues the OPCOM or TSM prompt at the terminal.

Any typing error on the directive line can be corrected by entering a CNTRL H or using the backspace key on the terminal. A directive line can be erased completely by pressing the rub or delete key.

If a directive verb is incorrect, OPCOM displays the message:

INVALID COMMAND VERB

It then returns the OPCOM or the TSM prompt so the directive can be reissued. If a directive parameter is incorrect, OPCOM displays an error message that identifies the invalid parameter. Reissue the directive as described above.

To abort a directive, press the break key or equivalent. If the directive produces no terminal output, break has no effect. If the directive produces output, the directive is processed up to the point of first output, a double asterisk (**) is displayed to indicate suppression, and the OPCOM or TSM prompt is returned.

2.10 ABORT Directive

The ABORT directive aborts a task. Only tasks in the system dispatch queue can be aborted.

Syntax:

ABORT { T,taskname / taskno }

T,taskname is the task name. If task name is used to abort a task, the task must be a unique copy.

taskno is the eight-digit task number assigned at activation

Response:

If successful: LF (line feed)

If the task is in a wait state (e.g., for I/O or run requests), the entry is deferred until it is safe to abort.

If the task does not have an abort receiver, files and devices are closed. IOCS purges blocking buffers and generates automatic EOFs as appropriate in an attempt to preserve data integrity. The DQE for the task is then deleted.

The KILL directive can be used to terminate outstanding I/O and run requests associated with the task with no deferred processing.

Any abort condition detected during abort processing (e.g., an ABORT directive issued when a task is already in an abort condition) kills outstanding I/O and deletes the task.

Usage:

```
ABORT T,PGMTEST
```

Aborts the task PGMTEST if it is a single copy load module with no outstanding allocation requirements or outstanding I/O.

```
ABORT 02000001
```

Aborts task number 02000001 if it has no outstanding allocation requirements or outstanding I/O.

2.11 ACTIVATE Directive

The ACTIVATE directive activates a task. System modules J.SWAPR and OPCOM cannot be activated by this directive. ACTIVATE initiates tasks independent of the interactive or batch environment at their base priorities. The other alternative is to use the ESTABLISH directive.

If ACTIVATE is used and the task is structured internally to suspend itself (with M.SUSP), it is suspended at the end of the activation sequence. It can resume on a timer (see the SETTIMER directive), connect to an indirectly connected interrupt level (see the CONNECT directive), or resume by the RESUME directive. The ESTABLISH directive suspends a task without structuring it internally to suspend.

Syntax:

ACTIVATE filename

filename is the one- to eight-character name of the permanent load module or executable image file name. It must be a system file. The name of the task and the name of the file are always identical.

Response:

LF (line feed)

Execution begins when the task is the highest priority task in the system.

If activation is not successful, like if an assigned device is not configured in the system, an abort code and message are displayed on the system console.

Usage:

```
ACTIVATE PGMTEST
```

Activates the permanent file PGMTEST.

2.12 BATCH Directive

The BATCH directive is used to read batch jobs from the current System Input Device (SID), a specified device, or a permanent file. If the file was created by the Text Editor utility, it must have been saved using the STORE directive.

Syntax:

BATCH [D,devmnc [,density,parity] / pathname]

If no parameters are specified, the job file is read from the System Input Device.

D,devmnc is a device mnemonic specifying the device containing the job file. Can contain an unblocked specification for files or magnetic tape, as well as other descriptors. (See Appendix A.)

density is H (high density) or L (low density) if the device is a 7-track magnetic tape. Otherwise, omit this field.

parity is E (even parity) or O (odd parity) if the device is a 7-track magnetic tape. Otherwise, omit this field.

pathname is the pathname associated with the file

Response:

LF (line feed)

The job file and any records specified in a $SELECT directive are copied to an SYC file. When the SYC file is complete, the job is entered into the batch stream (dynamic job stream queue) at the current batch priority.

If continuous batch mode has not been specified with the MODE directive, reading stops at the $$ statement. If continuous batch mode is in effect, reading continues until a $$$ statement is encountered.

Errors are displayed on the system Listed Output Device (LOD), or if LOD is not available, on any device related to LOD for automatic selection.

Usage:

```
BATCH
```

Reads batch jobs from the current System Input Device.

```
BATCH D,CR7A00
```

Reads batch jobs from the card reader on channel 7A, subaddress 00.

```
BATCH @SYSTEM(SYSTEM)PGMTEST
```

Reads batch jobs from the file PGMTEST on the system volume and directory.

2.13 BREAK Directive

The BREAK directive interrupts a task and enters the task's pseudointerrupt receiver.

Syntax:

BREAK { T,taskname / taskno }

T,taskname is the task name. The task must be a unique copy.

taskno is the task number assigned at activation

Response:

LF (line feed)

If the specified task is not in the system or has not established a pseudointerrupt receiver address using the M.BRK system service, the task continues and a message is displayed on the console or terminal. Alternatives are to ABORT or KILL.

Usage:

```
BREAK T,PGMTEST
```

If the task PGMTEST is a unique copy with a break receiver, the break receiver is entered.

```
BREAK 02000001
```

If task number 02000001 has a break receiver, the break receiver is entered.

2.14 CONNECT Directive

The CONNECT directive indirectly connects a task to the specified interrupt level so that when the interrupt occurs, the task is resumed. The interrupt level must be described as indirect during SYSGEN. Before using CONNECT, use the ACTIVATE or ESTABLISH directive to activate the task.

Syntax:

CONNECT { T,taskname,intlevel | taskno,intlevel }

T,taskname — is the task name. If the task name is used to connect a task, the task must be a unique copy.

intlevel — is the two-character hexadecimal interrupt priority level

taskno — is the eight-digit task number assigned at activation

Response:

LF (line feed)

If the connection is successful, the task is resumed at its current priority when the interrupt occurs.

If a task is already connected to the specified interrupt level or if the task is not in the system, the directive is ignored. A task abort or delete automatically disconnects a task from the interrupt.

The STATUS directive can indicate that a task is indirectly connected to an interrupt. However, the user is responsible for keeping track of what task is indirectly connected to a particular interrupt level.

Usage:

```
CONNECT T,PGMTEST,2F
```

Connects the task PGMTEST to interrupt level '2F' if PGMTEST is a unique copy task. Level '2F' must be defined at SYSGEN for indirect connection.

```
CONNECT 02000001,2F
```

Connects task number 02000001 to interrupt level '2F'.

2.15 CONTINUE Directive

The CONTINUE directive continues the system output task, system input task, or a specified user or system task that was held by the HOLD directive.

Syntax:

CONTINUE { PRINT [,devmnc] | PUNCH [,devmnc] | READ [,devmnc] | T,taskname | taskno }

PRINT — if no device mnemonic is specified, continues the system task controlling SLO output to the system LOD

PUNCH — if no device mnemonic is specified, continues the system task controlling SBO output to the system POD

READ — if no device mnemonic is specified, continues the system task controlling input from the SID

devmnc — is a device mnemonic. Continues the system output task controlling SLO (print) or SBO (punch) output to the specified device. Continues the system input task controlling SID (read) input from the specified device.

T,taskname — is the task name. If the task name is used to continue a task, the task must be a unique copy.

taskno — is the eight-digit task number assigned at activation

Response:

LF (line feed)

The HOLD bit is turned off in the DQE for the task and the task continues at the address following the hold. J.SSIN1, J.SSIN2, and J.SOUT system tasks control the device I/O described above.

Usage:

```
CONTINUE PRINT
```

Continues output to the current system LOD.

```
CONT READ,CR7801
```

Continues input from the card reader on channel 78, subaddress 01.

```
CONT T,PGMTEST
```

Continues the task PGMTEST if it is a unique copy task.

```
CONT 02000001
```

Continues task number 02000001.

2.16 DELETETIMER Directive

The DELETETIMER directive deletes the timer so that its specified function is no longer performed on time out. If the timer is not in the system, a message is sent to the operator. Timers are set up by the SETTIMER directive.

Syntax:

DELETETIMER timer

timer is the two-character ASCII name of the timer to be deleted

Response:

LF (line feed)

2.17 DEPRINT Directive

The DEPRINT directive deletes the SLO file currently being output to a particular device, deletes an SLO file generated for a particular task, or deletes all SLO files for a job.

Syntax:

```
DEPRINT [ J,jobno
          T,{taskname}
            {jobname}
          taskno
          D,devmnc ]
```

J,jobno — is the job sequence number assigned when the job was queued. All SLO files for the job are deleted.

T,taskname — is the task name. All SLO files for the task are deleted.

T,jobname — is the job name as specified in the $JOB statement. All SLO files for the job are deleted.

taskno — is the eight-digit task number assigned at activation. All SLO files for the task are deleted.

D,devmnc — is a device mnemonic used to delete the SLO file currently being output on a device other than the SYSGEN-defined LOD

If no parameters are specified, the SLO file currently being output to the system LOD is deleted.

Response:

LF (line feed)

The system output task producing the SLO file will delete the file from the system output queue. If a specified SLO file is being printed, printing stops.

Usage:

```
DEPRINT J,700
```

Deletes all SLO files for job number 700.

```
DEPR T,PGMTEST
```

Deletes all SLO files for the task PGMTEST.

```
DEPR 02000001
```

Deletes all SLO files for task number 02000001.

```
DEPR
```

Deletes the SLO file currently being printed on the system Listed Output Device.

```
DEPRINT D,LP7A00
```

Deletes the SLO file currently being output to the printer on channel 7A, subaddress 00.

2.18 DEPUNCH Directive

The DEPUNCH directive deletes the SBO file currently being output to a particular device, deletes an SBO file generated for a particular task, or deletes all SBO files generated for a job.

Syntax:

DEPUNCH [J,jobno | T, { taskname / jobname } | taskno | D,devmnc]

J,jobno — is the job sequence number assigned when the job was queued. All SBO files for the job are deleted.

T,taskname — is the task name. All SBO files for the task are deleted.

T,jobname — is the job name as specified in the $JOB statement. All SBO files for the job are deleted.

taskno — is the eight-digit task number assigned at activation. All SBO files for the task are deleted.

D,devmnc — is a device mnemonic used to delete the SBO file currently being output on a device other than the SYSGEN-defined POD

If no parameters are specified, the SBO file currently being output to the system POD is deleted.

Response:

LF (line feed)

If a specified SBO file is being output, output stops.

Usage:

```
DEPUNCH J,700
```

Deletes all SBO files for job number 700.

```
DEPU T,PGMTEST
```

Deletes all SBO files for the task PGMTEST.

```
DEPU 02000001
```

Deletes all SBO files for task number 02000001.

```
DEPU
```

Deletes the SBO file currently being punched on the current system Punch Output Device.

2.19 DISABLE Directive

The DISABLE directive executes a Disable Channel Interrupt (DCI) instruction, which prohibits the channel from requesting an interrupt. The channel is determined by the Controller Definition Table (CDT) entry associated with the specified priority interrupt level. To allow the channel to request interrupts, an Enable Channel Interrupt (ECI) instruction must be executed. See the ENABLE directive.

Syntax:

DISABLE intlevel

intlevel is the two-character hexadecimal interrupt priority, in the range 00 to 7F

Response:

LF (line feed)

If the request is unsuccessful, the following message is displayed:

REQUEST NOT EXECUTED

2.20 DISCONNECT Directive

The DISCONNECT directive disconnects a task from its indirectly connected interrupt level. See the CONNECT directive for a description of an indirectly connected task.

Syntax:

DISCONNECT { T,taskname / taskno }

T,taskname is the task name. If the task name is used to disconnect a task, the task must be a unique copy.

taskno is the eight-digit task number assigned at activation

Response:

LF (line feed)

Usage:

```
DISCONNECT T,PGMTEST
```

Disconnects task PGMTEST from its indirectly connected interrupt level if the task is a unique copy task.

```
DISCONNECT 02000001
```

Disconnects task number 02000001 from its indirectly connected interrupt level.

2.21 DISMOUNT Directive

The DISMOUNT directive dismounts OPCOM-mounted volumes requested by the OPCOM MOUNT directive. DISMOUNT calls the Resource Management Module (H.REMM) to update the Mounted Volume Table (MVT) entry. If the MVT indicates there are other users of the volume when the OPCOM DISMOUNT is requested, a dismount message is not issued, but subsequent mount requests are denied for that volume. A dismount message is issued when the last task attached to the volume detaches itself either through a task exit or a dismount request. At this time, the volume is dismountable.

A public volume cannot be dismounted through OPCOM.

Syntax:

DISMOUNT volname

volname is the name of the volume to be dismounted

Response:

If an error does not occur and no other tasks are attached to the specified volume, the following message is displayed and the volume can be dismounted:

DISMOUNT VOLUME ON devmnc

If an error occurs and the dismount is not successful; for example, if another task is attached to the specified volume, one of the following messages is displayed:

DISMOUNT PENDING DUE TO OUTSTANDING ASSIGNMENTS

VOLUME PUBLIC OR NOT OPCOM-MOUNTED

2.22 DUMP Directive

The DUMP directive reports the word locations specified by the starting and ending physical addresses. OPCOM dynamically allocates an SLO file for output. Output is listed in side-by-side ASCII-coded hexadecimal with ASCII format.

DUMP can also be used for an automatic dump if an abort occurs for a task running independent of the batch or interactive environment.

Syntax:

$$\text{DUMP} \left\{ \begin{matrix} \text{start,end} \\ \text{ON} \\ \text{OFF} \end{matrix} \right\}$$

start — is the starting hexadecimal physical word address

end — is the ending hexadecimal physical word address

ON — indicates that a dump is required if an independent task aborts

OFF — indicates that a dump is not required if an independent task aborts. The indication may have been previously set by a DUMP ON directive.

Response:

LF (line feed)

If an SLO file cannot be dynamically allocated when requested, the following message is displayed:

```
DUMP NOT PERFORMED - TRY AGAIN LATER
```

Usage:

```
DUMP 3000,3FFF
```

Dumps the contents of physical memory between logical address 3000 and logical address 3FFF to the SLO file.

2.23 ENABLE Directive

The ENABLE directive executes an Enable Channel Interrupt (ECI) instruction, which allows the channel to request interrupts from the CPU. The channel is determined by the Controller Definition Table (CDT) entry associated with the specified priority interrupt level.

Syntax:

ENABLE intlevel

intlevel is the two-character hexadecimal interrupt priority level

Response:

LF (line feed)

If the request is unsuccessful, the following message is displayed:

REQUEST NOT EXECUTED

2.24 ENTER Directive

The ENTER directive updates the system's date and time. This directive is issued only by the System Administrator.

Syntax:

ENTER { mo/dd/yy hh:mm:ss | dd-mo-yy hh:mm:ss | ddmonyy hh:mm:ss } [, [D] [, TZ=num]]

mo is the two-digit decimal month

mon is the three-character ASCII month abbreviation

dd is the two-digit decimal day

yy is the two-digit decimal year

hh is the two-digit decimal hour

mm is the two-digit decimal minute

ss is the two-digit decimal second

D indicates daylight savings time is in effect

num is the number of hours to bias the internal binary time. This can be a negative or positive number.

Response:

LF (line feed)

Usage:

```
ENTER 09/07/58 12:30:59
```

Sets the system date to September 7, 1958 and the system time to 12:30:59 p.m.

```
ENTER 09/07/58 12:30:59,D
```

Indicates daylight savings time is in effect.

```
ENTER 07-09-58 12:30:59, , TZ=3
```

Specifies a positive three-hour time bias. The double comma indicates daylight savings time is not in effect.

```
ENTER 07SEP58 12:30:59,D,TZ=-1
```

Specifies daylight savings time and a negative one-hour time bias.

2.25 ESTABLISH Directive

The ESTABLISH directive activates a task and suspends it at the end of the activation sequence. Tasks that are established remain inactive until they are activated by a timer (see the SETTIMER directive), connected to an indirectly connected interrupt level (see the CONNECT directive), or resumed (see the RESUME directive). When activated, they are brought into execution at their base priority (cataloged or linked).

This directive enables a user task to activate and suspend for resumption by a timer, interrupt, or RESUME directive without building the suspension into the task itself. ESTABLISH also allows the task to resume with all devices and memory allocation complete.

If a task activated with ESTABLISH is defined as RESIDENT when it is cataloged or linked, it is not swappable; otherwise, it can be swapped.

System modules J.SWAPR and OPCOM cannot be established through this service.

Syntax:

ESTABLISH loadmod

loadmod is the task name. It must be a system file.

Response:

LF (line feed)

The task is activated, then suspended.

Usage:

```
ESTABLISH PGMTEST
```

Establishes the permanent load module or executable image file PGMTEST with the logon owner name.

2.26 EXCLUDE Directive

The EXCLUDE directive removes a resident shared image from memory. The shared image is removed only after the use count decrements to zero.

Syntax:

EXCLUDE path

path is the pathname of the shared image

Usage:

EXCLUDE SYSTEM(DIR1)SHARE1

Removes the resident shared image SHARE1 on the system volume in directory DIR1 from memory.

EXCL SHARE1

Removes the resident shared image SHARE1 from the user's current working volume and directory from memory.

2.27 EXIT Directive

The EXIT directive terminates OPCOM and returns control to TSM.

Syntax:

EXIT

2.28 HOLD Directive

The HOLD directive holds a system output task, system input task, or a specified user or system task.

Syntax:

HOLD { PRINT [,devmnc] | PUNCH [,devmnc] | READ [,devmnc] | T,taskname | taskno }

PRINT — if no device mnemonic is specified, holds the system task controlling SLO output to the system LOD

PUNCH — if no device mnemonic is specified, holds the system task controlling SBO output to the system POD

READ — if no device mnemonic is specified, holds the system task reading input from the SID

devmnc — is a device mnemonic. Holds the system task controlling SLO (print) or SBO (punch) output to the specified device. Holds the system task controlling SID (read) input from the specified device.

T,taskname — is the task name. If task name is used to hold a task, the task must be a unique copy.

taskno — is the eight-digit task number assigned at activation

Response:

LF (line feed)

A hold bit is turned on in the DQE for the task. Its current status is retained so that it continues where it left off.

Usage:

```
HOLD PRINT
```

Holds output to the current system Listed Output Device.

```
HOLD READ,CR7801
```

Holds input from the card reader device on channel 78, subaddress 01.

```
HOLD T,PGMTEST
```

Holds the task PGMTEST if it is a unique copy.

```
HOLD 02000001
```

Holds task number 02000001.

2.29 INCLUDE Directive

The INCLUDE directive loads a resident shared image into memory. The shared image remains resident until excluded by the OPCOM EXCLUDE directive.

Syntax:

INCLUDE path

path is the pathname of the shared image

Usage:

INCLUDE @ SYSTEM(TEST)SHARE1

Loads the shared image SHARE1 on the system volume in directory TEST into memory.

INCL SHARE1

Loads the shared image SHARE1 from the user's current working volume and directory into memory.

2.30 KILL Directive

The KILL directive deletes a task from the system dispatch queue and terminates all outstanding I/O and run requests. It should only be used when the ABORT directive fails to remove a task or when a task is queued for a resource. File integrity may be affected because operations do not complete normally. To preserve system integrity, the KILL directive is processed as an abort for the amount of time specified in the SYSGEN KTIMO directive. If this does not remove the task, it is killed.

Use of the KILL directive may impact the integrity of blocked files because blocking buffers are not purged and end-of-file (EOF) marks are not written.

Syntax:

KILL { T,taskname / taskno }

taskname is the name of a single-copy task

taskno is the task number assigned at activation

Response:

LF (line feed)

Processing is not deferred for outstanding I/O or run requests. All outstanding I/O is terminated. The DQE for the task is deleted.

Usage:

```
KILL T,PGMTEST
```

Kills the task PGMTEST if it is a unique copy.

```
KILL 02000001
```

Kills task number 02000001.

2.31 LIST Directive

The LIST directive displays the entries in the system dispatch queue, system output print and punch queues, accounting file, system patch file, or traps which occurred in the IPU.

Syntax:

```
LIST  [ EXECUTION
        PRINT
        PUNCH
        ACCOUNT [,][OWNE=name][,][PROJ=proj][,][DATE=date]
                   [,] [ORIG={TSM.nnnn}]
                       {BATCH  }
        PATCHES
        IPU
        [taskname], [ownername] [,pseudonym]                  ]
```

EXECUTION lists all entries in the system dispatch queue

PRINT lists entries in the SLO file output queue

PUNCH lists entries in the SBO file output queue

ACCOUNT copies the contents of the job accounting file to an SLO file

PATCHES copies the contents of the system patch file to an SLO file

IPU lists the last twenty events which caused a trap in the IPU

taskname is the task name. If not specified, all tasks with the specified owner and/or pseudonym are listed.

ownername specifies the owner name for a particular task or all tasks belonging to the owner. If task name has not been specified, the comma must still be used:

```
LIST ,ownername
```

If the owner name is not specified, all tasks with the specified task name and/or pseudonym are listed.

pseudonym specifies the pseudonym for a task. A pseudonym is established by some system tasks (for example, TSM uses TSM.terminal number, and job control uses devmnc) and can be established by user tasks. The pseudonym allows identification of a particular copy of a task without the task number. For example, a TSM pseudonym allows you to identify the TSM copy for a particular terminal and in so doing, see what task and owner are currently active on the specified terminal.

If task name and owner name are not specified, two commas must precede the pseudonym:

```
LIST ,,pseudonym
```

If no parameters are specified, all entries in the system dispatch queue are listed.

Response:

To LIST with a specified task name, owner name, and/or pseudonym, OPCOM displays the status of the task(s). Any one or a combination of these parameters is used to select tasks. The parameters must be entered in the order shown in the syntax statement, supplying a comma for any missing parameter.

To LIST with no parameters, all entries in the system dispatch queue are displayed. The format of the display is the same as for LIST EXECUTION as described below.

To LIST EXECUTION or tasks selected by task identifiers for all tasks on system or each task selected the following is displayed:

taskno taskname ownername pseudonym priority state swap time

taskno	is the task number assigned at activation
taskname	is the task name
ownername	is the owner who activated the task
pseudonym	is the pseudonym name of the task
priority	is the current software priority level in the range 1 to 64
state	is a four-character identifier corresponding to the state of the task. See the STATUS directive in this section (description of response to STATUS T) for details.
swap	is the swap status of the task: IN the task is in memory OUT the task is outswapped
time	is the task execution time in milliseconds

Usage:

```
LIST PGMTEST
```

Lists all entries in the dispatch queue with the task name PGMTEST.

```
LIST PGMTEST,USER1
```

Lists all entries in the dispatch queue with the task name PGMTEST and owner name USER1.

```
LIST ,USER2
```

Lists all entries in the dispatch queue with the owner name USER2.

To LIST PRINT or LIST PUNCH:

jobno jobname ownername { pseudonym / QUEUED } priority

(for SLO or SBO files generated by batch jobs)

taskno taskname ownername { pseudonym / QUEUED } priority

(for each SLO or SBO file generated by a task running independent of the batch or interactive environment)

jobno	is the job's sequence number
jobname	is the job name as specified on the $JOB statement
ownername	is the owner who activated the task
pseudonym	is the pseudonym of the J.SOUT task that is currently processing the SLO or SBO files, such as, devmnc
QUEUED	is displayed if the SLO or SBO file is queued for output to a device
priority	is the current software priority level in the range 1 to 64
taskno	is the task number of the task that created the SLO or SBO file
taskname	is the task name

Note: When two asterisks are displayed, the directive has been aborted. Reissue the directive.

To LIST ACCOUNT:

With no parameters specified, outputs the current contents of the accounting file to an SLO file.

In response to LIST ACCOUNT specified with one or more keywords, only statistics of requested data is output. Keyword parameters can contain leading or trailing wild card characters in the form of question marks.

owner project date logon elapsed origin CPU time IPU time

owner	is the owner name
project	is the project name and number
date	is the numeric date
logon	is the time of day the owner logged on

elapsed	is the length of time the owner was logged on
origin	is the mode of operation which was used
CPU time	is the amount of CPU time used
IPU time	is the amount of IPU time used

The ORIG keyword for LIST ACCOUNT can be used in one of six ways:

ORIG=TSM	outputs statistics of all jobs run under TSM
ORIG=TSM.20??	output statistics of all jobs run under TSM on devices 20xx
ORIG=TSM.2009	outputs statistics of all jobs run under TSM on device 2009
ORIG=BATCH	outputs statistics of all batch jobs
ORIG=JOB.20??	outputs statistics of all batch jobs with job numbers 20xx
ORIG=JOB.2033	outputs statistics of batch job number 2003

Usage:

```
LIST ACCOUNT,OWNE=USER1, DATE=04/16/81,ORIG=TSM.20??
```

Outputs statistics on all jobs with owner name USER1 run on April 16, 1981 on any TSM device 20xx.

```
LIST ACCOUNT,OWNE=USER2, PROJ=120543
```

Outputs statistics on all jobs with owner name USER2 and project number 120543.

To LIST PATCHES:

Outputs the contents of the system patch file to an SLO file for printing.

To LIST IPU:

The last twenty events that caused a trap in the IPU are displayed. If less than twenty traps occurred, only those that occurred are displayed.

```
taskname PSD trap
```

taskname	is the task name
PSD	is the Program Status Doubleword of the task specifying the location where the trap occurred
trap	specifies the reason for the trap, such as SVC call, privilege violation, etc.

Usage:

LIST IPU

```
NUGGET    80040000    80000048    PRIVILEGE VIOLATION
AGENT     80036000    80000028    ARITHMETIC EXCEPTION
FASTED    80050000    80000068    START IPU
BRASS     80078000    80000080    SVC CALL
   :
```

2.32 MODE Directive

The MODE directive defines the following special system operations:

- Continuous batch - Batch stream input from SID is processed until the $$$ job control statement is encountered. All $$ job control statements are ignored.
- Inhibit banner page - Suppresses the banner page produced by system output tasks when processing SLO files.
- Inhibit mount message - Suppresses the mount message produced by J.MOUNT. Not valid for use with multivolume magnetic tape operations (for example, when mount message is displayed).
- Inhibit operator intervention - Suppresses all prompts normally displayed on the system console.

Syntax:

MODE { SCBT | RCBT | SIBP | RIBP | SIMM | RIMM | SNOP | RNOP }

SCBT	sets continuous batch mode
RCBT	resets continuous batch mode
SIBP	sets inhibit banner page
RIBP	resets inhibit banner page
SIMM	sets inhibit mount messages for disc and nonmultivolume magnetic tape operations
RIMM	resets inhibit mount messages
SNOP	sets inhibit operator intervention
RNOP	resets inhibit operator intervention

Response:

LF (line feed)

2.33 MODIFY Directive

The MODIFY directive resets a memory word at a physical address to the specified value. A mask can modify selective bit positions in the word. If no mask is specified, a mask of binary zeroes is used; for example, all bits are to be reset as indicated by the specified value.

The MPX-32 task debuggers (MPXDB, SYMDB, and DEBUGX32) can access and modify locations in a task's logical address space using logical addressing.

If a mask is used, a logical AND operation is performed between the specified memory word and the mask word, followed by a logical OR operation performed between the result of the logical AND and the specified value. This result is stored in the specified memory word.

Syntax:

MODIFY address,value [,mask]

address is the hexadecimal physical word address

value is the hexadecimal value of the word

mask is the hexadecimal word mask. If not specified, a mask of binary zeroes is used.

Response:

LF (line feed)

Usage:

```
MODIFY 3000,52535253
```

Stores the hexadecimal value '52535253' at physical location 3000.

```
MODI 12000,52535253,00000000
```

Stores the hexadecimal value '52535253' at physical location 12000.

```
MODIFY 3004,52530000, 0000FFFF
```

Changes the upper halfword of physical location 3004 to '5253'. The lower halfword is unchanged.

```
MODI 12004,00530055,FF00FFFF
```

Changes byte one of the word at physical location 12004 to '53'. Byte three of the word at physical location 12004 is logically ORed with '55' and the result is stored at that location. Bytes zero and two of the word are unchanged.

2.34 MOUNT Directive

The MOUNT directive mounts a volume on a device. These volumes are considered OPCOM-mounted volumes and remain mounted until an OPCOM DISMOUNT directive is issued. These volumes are allocated for implicit sharing.

The mounting of a volume establishes a task as a user of that volume. If the volume is not physically mounted when the MOUNT is requested, the nonresident task J.MOUNT is called. J.MOUNT issues a mount message at the system console and waits for the operator to respond that a volume has been physically mounted on the requested disc drive. This mount message may be inhibited if the mount option NOMSG is used.

Control returns to OPCOM after the operator has responded to the J.MOUNT prompt. If the mount was not successful, the reason for the failure is displayed.

If the OPCOM MOUNT directive mounts a volume as public, the volume is not considered dismountable and remains mounted as long as the system is running. It cannot be dismounted with a DISMOUNT directive.

Syntax:

MOUNT volname ON devmnc [SYSID=id] [OPTIONS= [PUBLIC] [NOMSG]]

volname — is the one- to sixteen-character left-justified, blank-filled name of the volume to be mounted

Specifying the wild card character (*) instead of the volume name allows the volume to be mounted on a specified drive regardless of the actual volume name.

devmnc — specifies the device on which to mount the volume. See Appendix A.

OPTIONS — if PUBLIC is specified, the volume is to be mounted for public use and cannot be dismounted (valid only if task has System Administrator attribute). If not specified, the default is nonpublic. If NOMSG is specified, a mount message is not displayed on the operator's console and it is assumed the volume has already been mounted. If NOMSG is not specified, a mount message will be displayed.

SYSID — specifies a three-character identifier required for port identification on multiport volumes only. Must be MP0, MP1...MPF. For compatibility, DP0 and DP1 can be entered. DP0 and DP1 are equivalent to MP0 and MP1.

Response:

J.MOUNT issues the following messages:

```
MOUNT DISC VOLUME ANEWVOL ON DRIVE DM0800
TASK OPCOM REPLY R,H,A OR DEVICE:
```

To indicate that the volume and drive specified in the MOUNT message are ready and proceed with the task, enter R (resume). To hold the task, enter H (hold). To abort the task, enter A (abort). To change the specified device, enter a new device mnemonic (such as DM0802).

If the volume is mounted successfully, the use count for the volume is incremented (if nonpublic) and the following message is displayed on the operator's console:

VOLUME MOUNT SUCCESSFUL

If an error occurs and the mount is not successful, the reason for the failure is displayed. The error messages are as follows:

INVALID VOLUME NAME

VOLUME MOUNT IS PENDING

VAT SPACE UNAVAILABLE

UNRECOVERABLE I/O ERROR TO VOLUME

MOUNT DEVICE NOT IN SYSTEM

INVALID MOUNT DEVICE SPECIFIED

J.MOUNT RUN REQUEST FAILED

VOLUME ALREADY MOUNTED

MOUNT DEVICE UNAVAILABLE

MVT SPACE UNAVAILABLE

INVALID PORT ID SPECIFIED, ENTER 'DP0', 'DP1', OR 'MP0'...'MPF'

UNABLE TO ACTIVATE MULTIPROCESSOR RECOVERY PROGRAM

WARNING - VOLUME SHOWS PORT DESIGNATOR MP(DP)n ALREADY ALLOCATED... REPLY C TO CONTINUE OR A TO ABORT

This message is displayed if the volume was not previously dismounted from the designated port and if volume clean-up was not performed. Before continuing, the operator should verify that the volume is not mounted under the same port ID on another processor.

If file overlap is detected, the following messages are displayed on the system console and the volume is not mounted:

FILE OVERLAP HAS OCCURRED IN RDnum
RD TYPE num
FILENAME IS name
SECTORS num THROUGH num

num is a hexadecimal number
name is the one- to sixteen-character file name

To mount the disc, set control switch seven.

Usage:

MOUNT ANEWVOL ON DM0800

2.35 OFFLINE Directive

The OFFLINE directive inhibits a specified device from all further allocation or inhibits using the IPU for task execution. The OFFLINE directive is invalid for shadowed Intelligent Peripheral System (IPS) devices; use the J.SHDW task. The device or IPU are put back on-line with the ONLINE directive.

Syntax:

OFFLINE { devmnc / IPU } [,DEAL]

devmnc is the device mnemonic of the device to be taken off-line (See Appendix A)

IPU inhibits all task execution on the IPU

DEAL indicates memory for the memory disc is deallocated after the device is marked off-line. DEAL is ignored if devmnc does not specify a memory disc.

Response:

For devmnc:

LF (line feed)

Any task that has assigned a specific device that is off-line will abort. If the device is a system device (LOD, POD, or SID), it will be bypassed for autoselection. If another device is available for autoselection, it will be used automatically so that tasks producing SLO and SBO output can proceed. If no other device is SYSGENed for autoselection, the SYSASSIGN directive can be used to assign autoselect devices.

The REDIRECT directive can also be used to divert output from a batch job to the system LOD or POD if needed.

For IPU:

LF (line feed)

While the IPU is off-line, tasks can only be executed on the CPU. Tasks currently executing on the IPU and tasks in the IPU request queue are dequeued and linked to the CPU ready-to-run queue.

Usage:

```
OFFLINE LP7A
```

Takes the line printer on channel 7A, subaddress 00 off-line.

```
OFFLINE IPU
```

Disables task execution on the IPU configured in the system.

```
OFFLINE DM0002 DEAL
```

Takes the memory disc on channel 00, subaddress 02 off-line and deallocates its memory.

2.36 ONLINE Directive

The ONLINE directive makes a specified device available for allocation or makes the IPU available for task execution.

Syntax:

ONLINE {devmnc / IPU}

devmnc is the device mnemonic of the device to be put on-line. See Appendix A.

IPU specifies that the IPU can be used for task execution

Response:

LF (line feed)

Usage:

```
ONLINE LP7A
```

Places the line printer on channel 7A, subaddress 00 on-line.

```
ONLINE IPU
```

Enables the IPU configured in the system to be used for task execution and resumes IPU task scheduling.

2.37 PURGEAC Directive

The PURGEAC directive deletes the current contents of the M.ACCNT accounting file.

Syntax:

PURGEAC

Response:

LF (line feed)

2.38 REDIRECT Directive

The REDIRECT directive redirects SLO or SBO output from a job or task to a device other than the SYSGEN-defined default device. If the task or job's SLO or SBO file is not queued for output or is in the output process, the directive is ignored.

Syntax:

REDIRECT { J,jobno / T,{taskname / jobname} / taskno / D,devmnc } [,devmnc]

J,jobno — is the job's sequence number

T,taskname — is the task name

T,jobname — is the job name as specified in the $JOB statement

taskno — is the eight-digit task number assigned at activation

D,devmnc — is the device where output is currently being processed

devmnc — is the device where output is to be redirected. If not specified, output is directed to the SYSGEN-defined LOD or POD.

Response:

LF (line feed)

Usage:

```
REDIRECT J,700,LP7EF8
```

Redirects the SLO files for job number 700 to the line printer on channel 7E, subaddress F8.

```
REDIRECT J,710
```

Redirects the SBO files for job number 710 to the System Punched Output device.

```
REDI D,LP7EF9
```

Redirects the SLO file currently being output to the line printer on channel 7E, subaddress F9 to the System Listed Output device.

2.39 REPRINT Directive

The REPRINT directive reprints the SLO file currently being output on a particular device, reprints all SLO files for a particular task, or reprints all SLO files for a particular job. The task or job must be queued for output or in the output process else the directive is ignored. Printing always begins at the beginning of the SLO file unless overridden by the PAGE parameter.

Syntax:

```
REPRINT [ J,jobno                          ] [ ,PAGE,start [,end] ]
        [ T,{taskname}                     ] [ ,times             ]
        [   {jobname}                      ]
        [ taskno                           ]
        [ D,devmnc                         ]
```

J,jobno — is the job number used to reprint all SLO files for a particular job

T,taskname — is the task name

T,jobname — is the job name as specified on the $JOB statement

taskno — is the eight-digit task number assigned at activation

D,devmnc — is the device mnemonic used to reprint SLO files being output on a device other than the SYSGEN-defined LOD. A channel and subaddress must be specified. See Appendix A for MPX-32 device addressing.

PAGE — specifies a page number on the current listing where reprinting should start and optionally end. If an ending page number is not specified, the default is end-of-file. If no page numbers are specified, printing starts at the beginning of the SLO file.

times — specifies the number of times to reprint the entire contents of the current file. If not specified, the default is one.

If no parameters are specified, the current SLO file being output to the SYSGEN-defined LOD is reprinted. SLO files being output on other devices can be specified by providing the device mnemonic as described above.

Response:

LF (line feed)

Usage:

If the LOD on channel 7E, subaddress 00 malfunctions in the middle of printing, the following directive would resume printing at page five of the SLO file that was interrupted and continue to the end-of file:

```
REPRINT D,LP7EF8,PAGE,5
```

If two additional copies of the SLO file currently being output to the LOD is needed, the following directive could be specified:

```
REPRINT,2
```

If the LOD malfunctions while printing job number 54, the following directive would reprint the SLO files for job 54 from the beginning of the job:

```
REPRINT J,54
```

If the line printer has less than 59 lines per page or the banner page is inhibited, the PAGE= parameter must be specified in the SYSGEN DEVICE directive for the SLO device.

2.40 REPUNCH Directive

The REPUNCH directive repunches the SBO file currently being output on a particular device or all SBO files for a particular task or job. The task or job must be queued for processing or in the output process or the directive is ignored. Punching always begins at the beginning of the SBO file.

Syntax:

```
REPUNCH [ J,jobno
          T, { taskname }
             { jobname  }
          taskno
          D,devmnc ]
```

J,jobno — is the job number used to repunch all SBO files for a particular job

T,taskname — is the task name

T,jobname — is the job name as specified in the $JOB statement

taskno — is the eight-digit task number assigned at activation

D,devmnc — is the device mnemonic used to repunch SBO files being output on a device other than the SYSGEN-defined POD. A channel and subaddress must be specified. See Appendix A for MPX-32 device addressing.

If no parameters are specified, the current SBO file being output to the SYSGEN-defined POD is repunched. SBO files being output on other devices can be specified by providing the device mnemonic as described above.

Response:

LF (line feed)

Usage:

If the POD malfunctions in the middle of a job, the following directive would repunch the SBO file that was interrupted:

```
REPUNCH
```

If the POD malfunctions while processing job number 54, the following directive would repunch the SBO files for job 54 from the beginning of the job:

```
REPUNCH J,54
```

2.41 REQUEST Directive

The REQUEST directive executes a Request Interrupt (RI) instruction for the specified interrupt priority level. Refer to the ENABLE and SETTIMER directive descriptions for additional information on controlling interrupts.

Syntax:

REQUEST intlevel

intlevel is the two-character hexadecimal interrupt priority level

Response:

LF (line feed)

2.42 RESUME Directive

The RESUME directive resumes execution of a task that has been suspended. A task can also be resumed by a timer or an interrupt. See the SETTIMER and CONNECT directives.

If the task is not suspended, the directive is ignored.

Syntax:

RESUME { T,taskname / taskno }

T,taskname is the task name. If the task name is used to resume a task, the task must be a unique copy.

taskno is the task number assigned at activation

Response:

LF (line feed)

Usage:

```
RESUME T,PGMTEST
```

Resumes the task PGMTEST if it is a unique copy.

```
RESUME 02000001
```

Resumes task number 02000001.

2.43 SEARCH Directive

The SEARCH directive searches memory within the specified physical addresses for a value under control of a mask. A logical AND operation is performed between the memory word and the task word. The result is compared to the value and, if equal, the memory address and contents are displayed.

The task debuggers (MPXDB, SYMDB, and DEBUGX32) can search memory locations in a task's logical address space using logical addressing.

Syntax:

SEARCH start,end,value [,mask]

start is the starting hexadecimal physical word address

end is the ending hexadecimal physical word address

value is the hexadecimal value used in comparison

mask is the hexadecimal word mask. If not specified, a mask of binary ones is used.

Response:

The following information is displayed for each successful comparison:

address content

address is the hexadecimal physical word address

content is the hexadecimal word content

Usage:

SEARCH 3000,3FFF,52535253

Displays all words between physical locations 3000 and 3FFF with the value '52535253' and their locations.

SEARCH 12000,12FFF,52535253,FFFFFFFF

Displays all words between physical locations 12000 and 12FFF with the value '52535253' and their locations.

SEARCH 12000,12FFF,00530000,00FF0000

Displays all words between physical locations 12000 and 12FFF with the value '53' in byte one and their locations.

2.44 SEND Directive

The SEND directive sends a message to a task which has established a message receiver. The message can be a maximum of 72 characters.

Syntax:

SEND { T,taskname / taskno } [,message]

T,taskname — is the task name. If the task name sends a message to a task, the receiving task must be a unique copy.

taskno — is the task number assigned at activation

message — is the message to be sent to the specified task. If the message is entered on the same line as the directive, no response is given. If more space is needed for a message, omit it from the directive line and press the carriage return. The following prompt will be displayed:

```
"ENTER MESSAGE"
??
```

Type the message in response to the ?? prompt, followed by a carriage return.

Usage:

```
??SEND T,PGMTEST
"ENTER MESSAGE"
??THIS IS A TEST
```

If the task PGMTEST is a unique copy and has a message receiver, the message THIS IS A TEST is sent to its receiver buffer address.

```
SEND 02000001,THIS IS A TEST
```

The message above is short enough to be included on the directive line. If task number 02000001 has a message receiver, the message THIS IS A TEST is sent to its receiver buffer address.

The operator at the system console enters:

```
??LIST OPCOM
03000004 OPCOM SYSTEM  62 TD T
04000009 OPCOM SMITH   62 TD T

??SEND 04000009
"ENTER MESSAGE"
??EXIT
```

The response at the terminal logged on under owner name SMITH is:

```
??TIME
01/29/78 11:59:35
**EXIT
```

The operator at the system console can send OPCOM directives to a terminal user's interactive OPCOM task. The directive is queued and processed after the user's current directive is complete as shown above.

Owners who are not restricted in their ability to access tasks with different owner names can also use this capability.

2.45 SETTIMER Directive

The SETTIMER directive causes one of three types of events at specified time intervals: activate a task, resume a task, or execute a Request Interrupt (RI) instruction for a hardware interrupt level.

The time intervals are specified in terms of time units. The duration of a time unit is defined at SYSGEN by the NTIM and MTIM directives.

Syntax:

SETTIMER timer,t1,t2, { ACP,filename / RST,taskno / RQI,intlevel }

timer	is the two-character ASCII name of the timer being created
t1	is the decimal number of time units until the first time out of this timer
t2	is the decimal number of time units used to reset this timer at each time out. If t2 is zero, the timer will time out only once.
ACP	indicates the time-out event is the activation of the specified task
filename	is the name of the system file containing the task
RST	indicates the time-out event is the resumption of the specified task
taskno	is the eight-digit task number assigned at activation
RQI	indicates the time-out event is the execution of a Request Interrupt for the specified interrupt level
intlevel	is the two-character hexadecimal interrupt priority level

Response:

LF (line feed)

Usage:

```
SETTIMER AB,2,0,ACP,PGMTEST
```

Activates the file PGMTEST when its associated timer (AB) expires. The time out is set to occur only once.

```
SETTIMER AB,2,3,RST,02000001
```

Resumes task number 02000001 when its associated timer (AB) expires. The timer AB is reset to three upon each following time event.

```
SETTIMER AB,2,0,RQI,2F
```

Issues a Request Interrupt for interrupt level 2F when its associated timer (AB) expires. The time out is set to occur only once.

2.46 SNAP Directive

The SNAP directive dumps word locations specified by starting and ending physical addresses to the requesting terminal.

The task debuggers (MPXDB, SYMDB, and DEBUGX32) can snap memory locations in a task's logical address space using logical addresses.

Syntax:

```
SNAP start,end
```

start is the starting hexadecimal physical word address

end is the ending hexadecimal physical word address

Response:

The following information is displayed for each successful comparison:

```
address  content
```

address is the starting hexadecimal physical word address of the current line

content is the hexadecimal word content

Usage:

```
SNAP 3000,3FFF
```

Displays the contents between physical locations 3000 and 3FFF.

2.47 STATUS Directive

The STATUS M directive displays a current memory utilization map.

The STATUS CHAN directive returns information relating to a particular I/O device channel, including the number of controllers, the number of devices attached, the number of I/O entries currently queued on the channel, the status of the channel, the status of each controller connected to the channel, and the status of each device assigned to the controller. The read, write, read error, and write error fields are optionally zeroed.

The STATUS CONT directive returns information relating to a particular device controller, including the number of devices and I/O entries currently queued on the controller, the status of the controller, and the status of each device assigned to the controller.

The STATUS D directive returns information such as device type and device status for the specified device.

The STATUS T (or STATUS taskno) directive returns the current status of a task and its significant attributes.

The STATUS IPU directive returns information about the IPU.

The STATUS VOLU directive returns information relating to a particular mounted volume or all mounted volumes, depending upon the option specified. For each active entry in the Mounted Volume Table (MVT), information such as the volume name, current number of users, maximum and current number of logical blocks available for resource space allocation, and other miscellaneous status are displayed.

Syntax:

STATUS { M [,class] | CHAN,address | CONT,devmnc | D,devmnc | T,taskname | taskno | IPU | VOLU [,volname] }

M	with no class, displays physical address boundaries and other information for each class of memory
class	restricts the display to class E, H, or S
address	is a two-character hexadecimal channel number or GPMC device address in the range 80 to FF
devmnc	is a device mnemonic. See Appendix A.
T,taskname	is the task name
taskno	is the task number assigned at activation

IPU — returns on-line or off-line status and notes which task, if any, is running

VOLU — with no volname, displays the status of all mounted volumes

volname — specifies the one- to sixteen-character volume name of a mounted volume where status is to be displayed

Response:

To STATUS M:

MEM low -- high {FREE / ALOC / SHAR} class [MALFUNC] [NONPRES]

class — E for memory below 128KW
H for high speed memory above 128KW
S for slow speed memory above 128KW

low — the low physical memory address boundary

high — the high physical memory address boundary

FREE — free memory

ALOC — allocated memory

SHAR — shared memory

MALFUNC — malfunctioning memory. This field is blank if memory is operational.

NONPRES — nonpresent memory. This field is blank if the specified memory is configured.

To STATUS CHAN,address:

chaddr numcont CONTROLLERS {CHANNEL INITIALIZED / CHANNEL NOT INITIALIZED}

devchan CONTROLLER contaddr UNITS units [I/O cntqueue] [CONTR'L MALF] [CONTROLLER INITIALIZED / CONTROLLER NOT INITIALIZED]

(the two controller lines are repeated for each controller assigned to the channel)

DEV devaddr {ON / OFF} {FREE / ALOC / SHAR / TSM} [IOQ devqueue] [DEV MALF] [taskno] [dest]

(the device line is repeated for each device on the controller)

chaddr — is the channel address

numcont — is the number of controllers assigned to the channel

CHANNEL INITIALIZED	is displayed if extended I/O initialization (INCH) has been performed for this channel
CHANNEL NOT INITIALIZED	is displayed if extended I/O initialization has not been performed for this channel
devchan	is the device type and channel address for the controller
contaddr	is the controller address
units	is the number of units on the controller
cntqueue	is the number of I/O requests queued for the controller. This field is blank if the IOQ is linked for the Unit Definition Table (UDT).
CONTR'L MALF	is displayed if a controller malfunction is detected. Otherwise, this field is blank. A controller malfunction occurs when a controller fails the initialization procedure. In addition, the appropriate I/O handler sets the device malfunction flag for all the devices associated with the controller.
CONTROLLER INITIALIZED	is displayed if extended I/O initialization has been performed for this controller
CONTROLLER NOT INITIALIZED	is displayed if extended I/O initialization has not been performed for this controller. These messages are only displayed for class F devices.
devaddr	is the device address and device type
ON	indicates the device is on-line
OFF	indicates the device is off-line
FREE	indicates the device is free
ALOC	indicates the device is allocated
SHAR	indicates the device is shared
TSM	indicates the device is in use by TSM
devqueue	is the number of I/O requests queued for this device. This field is blank if the IOQ is linked from the Controller Definition Table (CDT).
DEV MALF	is displayed if a device malfunction has been detected. Otherwise, this field is blank. A device malfunction occurs when a device fails to respond to the Lost Interrupt Recovery procedure or if the controller it is assigned to has a controller malfunction.

taskno — is the task number of the task the device is allocated to. This field is blank if the device is not allocated to. If the device is shared, the task number is not displayed.

dest — indicates the device's availability for automatic selection as a destination device for SLO and SBO files as follows:

RT	SLO	(real-time SLO files)
RT	SBO	(real-time SBO files)
BT	SLO	(batch SLO files)
BT	SBO	(batch SBO files)

To STATUS CONT, devmnc:

devchan — CONTROLLER contaddr UNITS units [I/O cntqueue] [CONTR'L MALF]
CONTROLLER INITIALIZED
CONTROLLER NOT INITIALIZED

DEV devaddr { ON / OFF } { FREE / ALOC / SHAR / TSM } [IOQ devqueue] [DEV MALF] [taskno] [dest]

(the device line is repeated for each device on the controller)

devmnc — is a device mnemonic. See Appendix A.

devchan — is the device type and channel address for the controller

contaddr — is the controller address

units — is the number of units on the controller

cntqueue — is the number of I/O requests queued for the controller. This field is blank if the IOQ is linked from the Unit Definition Table (UDT).

CONTR'L MALF — is displayed if a controller malfunction is detected. Otherwise, this field is blank. A controller malfunction occurs when a controller fails the initialization procedure. In addition, the appropriate I/O handler sets the device malfunction flag for all the devices associated with the controller.

CONTROLLER INITIALIZED — is displayed if extended I/O initialization is performed for this controller

CONTROLLER NOT INITIALIZED — is displayed if extended I/O initialization is not performed for for this controller. These messages are only displayed for class F devices.

devaddr — is the device address and device type

ON — indicates the device is on-line

OFF — indicates the device is off-line

FREE	indicates the device is free
ALOC	indicates the device is allocated
SHAR	indicates the device is shared
TSM	indicates the device is in use by TSM
devqueue	is the number of I/O requests queued for this device. This field is blank if the IOQ is linked from the Controller Definition Table (CDT).
DEV MALF	is displayed if a device malfunction is detected. Otherwise, this field is blank. A device malfunction occurs when a device fails to respond to the Lost Interrupt Recovery procedure or if the controller it is assigned to has a controller malfunction.
taskno	is the task number of the task the device is allocated to. This field is blank if the device is not allocated.
dest	this field indicates the device's availability for automatic selection as a destination device for SLO and SBO files as follows:

RT	SLO	(real-time SLO files)
RT	SBO	(real-time SBO files)
BT	SLO	(batch SLO files)
BT	SBO	(batch SBO files)

To STATUS D,devmnc:

$$\text{DEV devaddr } \left\{\begin{matrix}\text{ON}\\\text{OFF}\end{matrix}\right\} \left\{\begin{matrix}\text{FREE}\\\text{ALOC}\\\text{SHAR}\\\text{TSM}\end{matrix}\right\} \text{ [IOQ devqueue][DEV MALF][taskno][dest]}$$

devmnc	is a device mnemonic (see Appendix A)
devaddr	is the device address and the device type
ON	indicates the device is on-line
OFF	indicates the device is off-line
FREE	indicates the device is free
ALOC	indicates the device is allocated
SHAR	indicates the device is shared
TSM	indicates the device is in use by TSM
devqueue	is the number of I/O requests queued for this device. This field is blank if the IOQ is linked from the Controller Definition Table (CDT).

DEV MALF is displayed if a device malfunction has been detected. Otherwise, this field is blank. A device malfunction occurs when a device fails to respond to the Lost Interrupt Recovery procedure or if the controller it is assigned to has a controller malfunction.

taskno is the task number of the task the device is allocated to. This field is blank if the device is not allocated. If the device is shared, the task number is not displayed.

dest indicates the device's availability for automatic selection as a destination device for SLO and SBO files as follows:

RT	SLO	(real-time SLO files)
RT	SBO	(real-time SBO files)
BT	SLO	(batch SLO files)
BT	SBO	(batch SBO files)

To STATUS T,taskname or STATUS taskno:

taskno taskname ownername pseudonym priority type origin
MEM=class MAPBLKS=mapblocks priv res state
NWIO=requests RRCT=requests MRCT=requests [DEV=dev]
[GQID=General Wait Queue Message queuing ID]

taskname:

IS OUTSWAPPED	the task is outswapped
IS UNSWAPPABLE	the task is locked in memory
ABORT REQUESTED	a task abort is in progress
HAS MESSAGE RECEIVER	the task has a message receiver
HAS BREAK RECEIVER	the task has a break receiver
IS INDIRECTLY CONNECTED	the task is indirectly connected
IS A UNIQUE COPY LOAD MODULE	the task is unique
IS MULTICOPIED OR SHARED LOAD MODULE	the task is not unique
taskno	is the task number assigned at activation
taskname	is the task name
ownername	is the owner of the task

pseudonym	is the pseudonym of the task
priority	is the current priority level in the range 1 to 64
type	RT (real time, priority 1-54) TD (time distribution, priority 55-64)
origin	B (batch) T (terminal) Ø (independent)
class	E (memory below 128KW) H (high speed memory above 128KW) S (slow speed memory above 128KW)
mapblocks	number of 2K swappable, nonshared map blocks used by the task
priv	U (unprivileged) P (privileged)
res	N (nonresident) R (resident)
state	is a four-character identifier corresponding to the state of the task as follows:

Current task in execution (CURR)

Real-time priority - Ready-to-run (SQRT)

Time distribution levels - Ready-to-run (SQ55 - SQ64)

Current IPU task in execution (CIPU)

Requesting IPU task (RIPU)

Wait interactive (SWTI)

Waiting for wait I/O (SWIO)

Wait, sending message (SWSM)

Wait, sending run request (SWSR)

Wait, low speed output (SWLO)

Wait, suspended for message interrupt, time out, or resume (SUSP)

Wait for wait run request or time out (RUNW)

Wait, operator hold (HOLD)

Wait for any no-wait I/O, no-wait run request, or any message interrupt or break (ANYW)

Wait, disc space (SWDC)

Wait, peripheral (SWDV)

Wait, memory (MRQ)

Wait, memory pool (SWMP)

Preactivation phase (PREA)

Dispatach queue available for allocation by a task (FREE)

General wait queue - resourcemark (SWGQ)

NWIO — is the number of no-wait I/O requests outstanding

RRCT — is the number of run receiver requests outstanding

MRCT — is the number of message receiver requests outstanding

DEV — is displayed if the state is SWDV or SWDC. A channel and subaddress of zero means any device of the specified type.

GQID — is displayed if the state is SWGQ

Along with GQID, the reason why the task is waiting in the general wait queue is displayed. General wait queue messages are:

QUEUED FOR VOLUME RESOURCE
QUEUED FOR ART SPACE
QUEUED FOR VOLUME MOUNT
QUEUED FOR RESOURCEMARK LOCK
QUEUED FOR EVENTMARK
QUEUED FOR READ WAIT FOR WRITER
QUEUED FOR SHARED MEMORY TABLE
QUEUED FOR SYNC RESOURCE LOCK
QUEUED FOR MOUNTED VOLUME TABLE
QUEUED FOR DUAL-PORT LOCK

To STATUS IPU:

{ON / OFF} — TASK=taskname OWNER=ownername TASK#=taskno

ON — indicates the IPU is on-line

OFF — indicates the IPU is off-line

taskname — is the task name of the task running on the IPU

ownername — is the owner name of the task running on the IPU

taskno — is the task number of the task running on the IPU

If a task is not running on the IPU, the following message is displayed:

{ON / OFF} THERE IS NO TASK RUNNING IN THE IPU

If an IPU is not on the system, the following message is displayed:

THERE IS NO IPU ON THIS SYSTEM

To STATUS VOLU:

```
volname  devmnc  user  total  avail
      mounted
      volmsg
```

volname — is the one- to sixteen-character left-justified, blank-filled name of the mounted volume

devmnc — is the device the volume is mounted on. See Appendix A.

user — is the number of current users for the volume (nonpublic volumes only)

total — is the maximum number of logical blocks available for resource space allocation

avail — is the approximate number of logical blocks currently available for resource space allocation. This number includes some blocks allocated by the operating system.

mounted — indicates how the volume is mounted:

SYSTEM VOLUME
OPERATOR MOUNTED VOLUME
PUBLIC VOLUME

volmsg — miscellaneous information describing the volume. One or more of the following messages are displayed:

INHIBIT MOUNT MESSAGE
VOLUME DEVICE OFFLINE
VOLUME NOT SAFE FOR USE
DISMOUNT PENDING (NO MOUNTS ALLOWED)
DMAP LOCKED
SPACE MAP LOCKED
ROOT DIRECTORY LOCKED
MULTIPORTED VOLUME (PORT n)
MOUNTED FOR SHARED USE
MEMORY DISC
PORTS IN USE:n[,n...]
PORTED TO PREVIOUS RELEASE (3.2C OR BEFORE)

Usage:

STATUS VOLU,TBRD

Displays the current status of the volume TBRD.

STATUS M,E

Displays the current memory utilization map for E class memory.

STATUS CHAN,7E

Displays the status of channel 7E, the controllers assigned to the channel, and all the devices assigned to the controllers.

STATUS D,LP7EF8

Displays the status of the line printer device on channel 7E, subaddress F8.

STATUS T,PGMTEST

Displays the current status of all tasks named PGMTEST.

STATUS 02000001

Displays the status of task number 02000001.

STATUS CONT,DM0800

Displays the status of controller 0 assigned to DM08 and all devices assigned to the controller.

2.48 SYSASSIGN Directive

The SYSASSIGN directive establishes the availability of a device for automatic selection as the final destination device for printed (SLO) and punched (SBO) output. It can change the default System Input Device (SID) for batch.

Syntax:

SYSASSIGN $\left\{ \begin{array}{l} \{ON \atop OFF\} \text{ ,devmnc } , \{L \atop P\}, \quad \{R \atop B\} \\ \text{SID, devmnc [,density,parity]} \end{array} \right\}$

ON — makes the device available for automatic selection

OFF — makes the device unavailable for automatic selection

devmnc — is a device mnemonic. See Appendix A.

L — specifies the device is the destination for listed output (SLO files)

P — specifies the device is the destination for punched output (SBO files)

R — selects the device for real-time output

B — selects the device for batch output

density — is H for high density or L for low density if the specified device is a 7-track magnetic tape; otherwise, omit this parameter

parity — is E for even parity or O for odd parity if the specified device is a 7-track magnetic tape; otherwise, omit this parameter

Response:

LF (line feed)

Usage:

```
SYSASSIGN ON,LP7E,L,R
```

Makes the line printer on channel 7E, subaddress 00 available for automatic selection for spooled real-time SLO output.

```
SYSASSIGN SID,CR7801
```

Changes the default System Input Device to the card reader on channel 78, subaddress 01.

2.49 TIME Directive

The TIME directive displays the current date and time on the system console or a user terminal. Time is displayed in one of three formats, depending on how the time was entered during initialization.

Syntax:

TIME

Response:

{ mo/dd/yy hh:mm:ss
ddmonyy hh:mm:ss
dd-mo-yy hh:mm:ss }

mo is the two-digit decimal month

mon is the three-character ASCII month abbreviation

dd is the two-digit decimal day

yy is the two-digit decimal year

hh is the two-digit decimal hour (24-hour time)

mm is the two-digit decimal minute

ss is the two-digit decimal second

Usage:

If the date and time were entered in the format 03/29/84 10:30:03 at initialization, the TIME directive displays:

```
??TIME
03/29/84 11:59:35
```

If the date and time were entered in the format 26SEP84 12:50:10 at initialization, the TIME directive displays:

```
??TIME
26SEP84 13:41:47
```

If the date and time were entered in the format 29-08-84 08:03:05 at initialization, the TIME directive displays:

```
??TIME
29-08-84 08:10:00
```

2.50 TURNON Directive

The TURNON directive activates or resumes a task at a specified time and reactivates it at selected intervals by creating a timer table entry using a timer ID. This directive resumes a task only if that task is unique and already active. If the target task is multicopied and already active, another copy is activated.

Syntax:

TURNON filename,time,[reset],timerid

filename is the name of the permanent load module or executable image file name. It must be a system file.

time is the time of day the task is activated or resumed. The time should be entered as two-digit decimal units based on a 24-hour clock in the form hh:mm:ss.

reset is the time interval to elapse before resetting the clock at each time out. The time should be entered in the same format as the time parameter described above. The task is reactivated at each time out. If a reset value is not specified, the comma denoting the field must still be specified, and the task is activated or resumed only once.

timerid is the two-character ASCII name of the timer that is created

2.51 UNLOCK Directive

The UNLOCK directive recovers multiprocessor resources when a system or disc failure occurs on one of the processors. UNLOCK sends a run request to the multiprocessor shared volume recovery task J.UNLOCK. The system administrator attribute is needed to execute this directive.

When J.UNLOCK completes, reboot the off-line processor. The shared volumes can be remounted to the logical port, but volume clean-up must be inhibited.

Syntax:

UNLOCK

Response:

The following message is displayed to the system console, and prompts the operator for the port ID to be unlocked:

J.UNLOCK - ENTER SYSID OF PORT TO UNLOCK (DP0,DP1,MP0...MPF,ALL):

If the operator enters a specific port ID, then J.UNLOCK only resets access information pertaining to the specific processor. Resource descriptor locks are not reset without operator approval. This allows normal multiprocessor accesses by other processors during J.UNLOCK's execution.

If the user enters the "ALL" option, access information and resource descriptor locks are unconditionally reset. All processors sharing the multiprocessor volume should be inactive during an unlock with the ALL option.

UNLOCK displays a message for each resource descriptor that is processed:

```
J.UNLOCK -  PROCESSING RESOURCE DESCRIPTOR nnnnnnnn.
            RESOURCE TYPE: (file, directory, etc.)
            DIRECTORY NAME: (name of parent directory)
            RESOURCE NAME: (file name, directory name, etc.)
```

If a locked resource descriptor is not unlocked after 20 retries, the operator is prompted to confirm the unlock process:

```
J.UNLOCK -  RD nnnnnnnn STILL LOCKED.
            RESOURCE TYPE: (file, directory, etc.)
            DIRECTORY NAME: (name of parent directory)
            RESOURCE NAME: (file name, directory name, etc.)
            CONTINUE WITH UNLOCK? (Y or N)
```

If the operator replies Y (yes), the resource descriptor is unlocked, and all multiprocessor access information in the resource descriptor is reset. If the operator replies N (no), the resource descriptor is unchanged. The default is N.

WARNING: The operator should verify that the resource is not in use on any other processor before preceding with the unlock.

The following are J.UNLOCK error messages which are displayed on the system console as errors occur:

READ ERROR DURING MULTIPLE RD READ

STATUS RMnn - UNABLE TO MOUNT MULTIPORT VOLUME

STATUS RMnn - UNABLE TO ASSIGN MULTIPLE RD'S

ERROR STATUS xxxx ON RD xxxx

ERROR STATUS xxxxxxxx SENSE STATUS xxxxxxxx ON RD xxxx
(port ID) IN USE ON CPU. CONTINUE? (Y OR N)

This message is displayed when the port ID is in use with any mounted volume on this processor.

WARNING: The operator should verify that the resource is not in use on any other processor.

2.52 WAIT Directive

The WAIT directive puts a terminal into a special wait state so it can receive messages. WAIT can receive messages pertaining to one or more batch jobs that have been submitted before continuing terminal operation. Messages output by job control are displayed as the job is processed. To exit the wait state after all messages have been received, press the wakeup character used to logon.

If WAIT is not used, an inactive terminal is in a read condition. A read is not interrupted by messages unless the terminal times out. Carriage returns must be issued to get out of a read condition and receive a message.

There is no overhead associated with a terminal in the special wait state; it is the same as being logged off.

Syntax:

WAIT

Usage:

```
        TSM > RUN OPCOM
1       ??BATCH RUNX
          BATCH JOB SUBMITTED
        ??WAIT
2       0002 USER1 $JOB X
        0002 USER1 $EOJ X
3       <wakeup>
        ??
```

1 The job control file RUNX is submitted by the BATCH directive. The missing volume and directory components in the pathname imply the file RUNX should be located in the user's current working volume and directory.

2 The job number, owner name, and job name are displayed upon initiation. When the job completes successfully or with an abort, the batch end-of-job message is displayed. The form of the message is:

jobno owner {$JOB / $EOJ} jobname

This job is number 0002. It belongs to owner USER1, and the job name is X.

3 The wakeup character can be used to return control to OPCOM.

CHAPTER 3

VOLUME MANAGER (VOLMGR)

3.1 Introduction

The Volume Manager (VOLMGR) creates or deletes permanent disc file space, global partitions, and Datapool partitions, and provides backup copies of system and user files.

Files in user and system directories can be created, deleted, copied, and expanded. Directories can only be created or deleted.

VOLMGR can be activated in the interactive or batch modes. In the interactive mode, directives are accepted from either a terminal or a TSM select file. Static assignments do not have to be made for directive I/O. Static assignments must be made for saves and restores from magnetic tape or floppy disc.

3.2 Save Tape Format

The save tape consists of one or more save images with three consecutive end-of-file (EOF) marks indicating the logical end-of-tape (see Figure 3-1). A save image consists of a directory of files contained in the image, a resource descriptor for each file, and the actual files. Each save image is delineated by two consecutive EOF marks (see Figure 3-2). One save image is produced each time a SAVE directive is issued. Saved files are delineated by a single EOF mark. All tape records written will be a multiple of the disc block size (192 words) since saves to unformatted disc devices are possible. The smallest record will be one block, the largest record will be eight blocks (1536 words). Eight block records are written until the remaining block count is less than eight, then a record less than eight is written. File data is written to tape in unblocked format.

Read errors from disc are ignored while saving files as long as the full count is transferred. Warning messages are printed for saved files with read or write errors. The warning messages contain the full file pathname and the number of read and write errors that occurred. Any other type of disc I/O error, for example no data transfer or a partial transfer, terminates the save process and writes an appropriate message. If any tape I/O error occurs during a save operation VOLMGR backspaces the tape to the end of the last saved file, sets up a special RDTR record, builds a logical EOF, and exits with a V006 abort code posted.

The save tape is written in unblocked format to speed operations. Files are transferred to and from the save tape without regard to their contents, for example, files on disc will be opened unblocked. Since the RESTORE directive operates on one save image only, the save tape must be positioned to the desired save image by the SKIP IMAGE or BACKSPACE IMAGE directive before issuing the RESTORE directive.

Files are restored/copied to temporary disc files. If the destination file already exists and either the source or destination file is a fast file, the temporary file replaces the existing destination file (H.VOMM,23). If the source or destination file is not a fast file, the existing destination file is deleted and the temporary file is changed to a permanent file (H.VOMM,9).

This feature is useful when files are allocated by a resource identifier (RID). The RID of a nonfast file changes when the file is restored or copied, thereby invalidating any hard coded references to the file in its RID.

A phase error occurs if files referenced during internal directory preparation are nonexistent when a directive's actions are invoked. When this happens with the SAVE directive, a Resource Descriptor Tape Record (RDTR) and an EOF are written for directory entries without corresponding file data. The RDTR contains the full file pathname as recorded in the image directory and a code to indicate it was not saved.

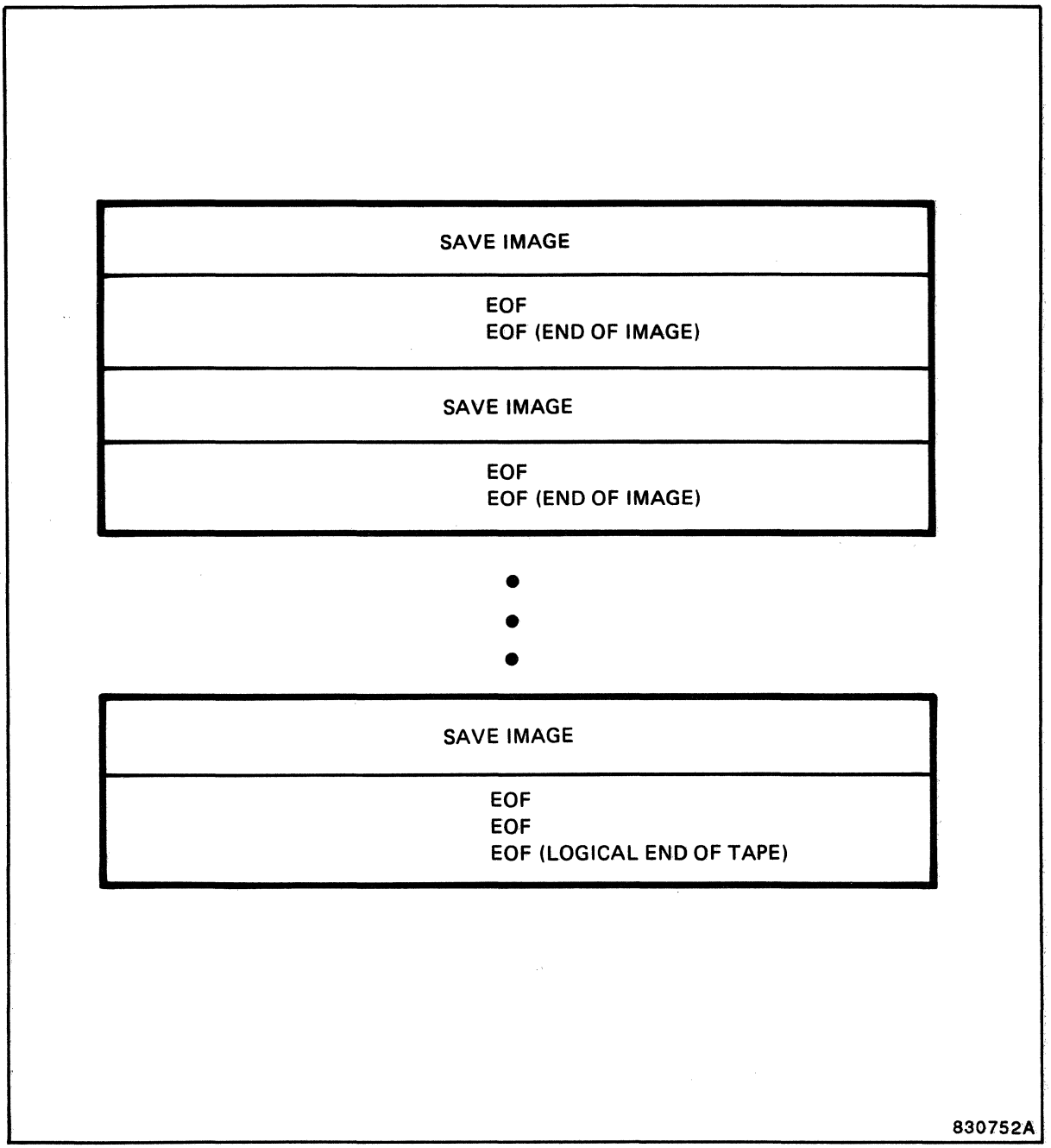


Figure 3-1. Save Tape Structure

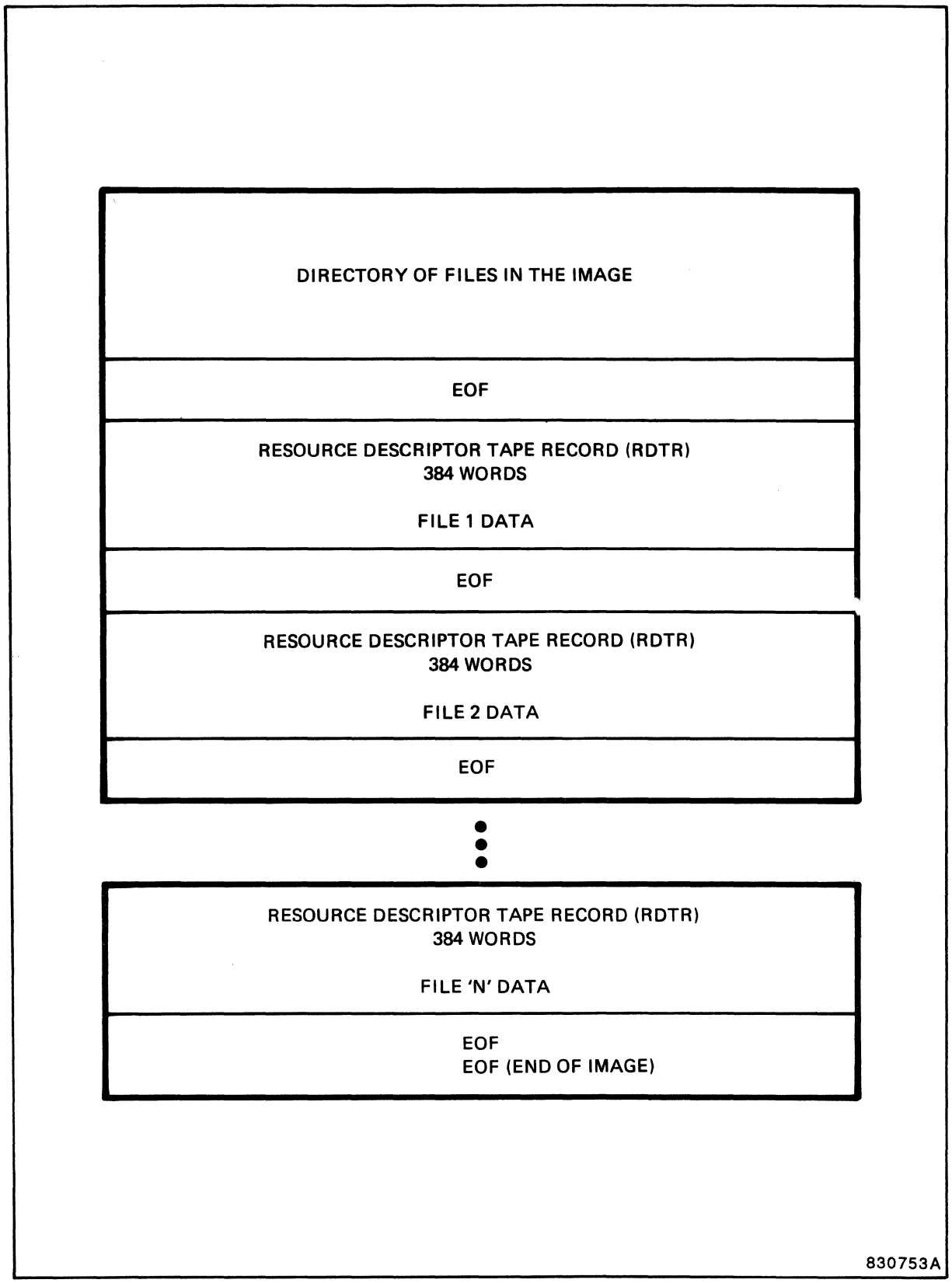


Figure 3-2. Save Image Structure

3.3 Save Image Directory

When a SAVE directive is issued, all derived file pathnames are sorted in ascending alphabetical order and written to the save tape in the form of a directory followed by a single EOF mark (see Figure 3-3). The relative position of every file in the save image is determined by the number of EOF marks to pass over in order to reach the file. To reach the first file in the save image, one EOF mark would be passed over. A copy of this directory is maintained on a temporary disc file for the actual file lookup and data transfer phases of the save operation. A file may only appear once in a save image. Multiple references to a file in a single SAVE directive are ignored.

Individual files can be recovered in a save image without the VOLMGR referencing the save image directory. Any single file in a save image can be identified by obtaining a list of the file pathname and other pertinent information. This single file can then be transferred to disc. This transaction takes place with the assumption that the save tape has already been positioned to the desired file within a save image; thus the save image directory is not used to locate the file. The directives that are used to position a tape are SKIP FILE, BACKSPACE FILE, RESTORE POSITION, and LOG SAVEFILE. Familiarity with the save tape and save image formats is necessary before using these directories because it is possible to position the save tape to something other than a saved file, i.e., an image directory or between consecutive EOF marks.

3.4 Resource Descriptor Tape Record (RDTR)

One RDTR which precedes the file's actual data is generated for each file saved. The first half of the RDTR contains the full pathname used to access the file when it was saved, an indicator of whether the file's data actually follows, a record type indicator to distinguish the RDTR from save image directories, and a Resource Create Block (RCB) for the file's directory. The RCB can be used on subsequent RESTOREs to recreate the directory. The second half of the RDTR contains an exact copy of the file's resource descriptor as it exists on the disc volume. Figure 3-4 illustrates the format of the Resource Descriptor Tape Record.

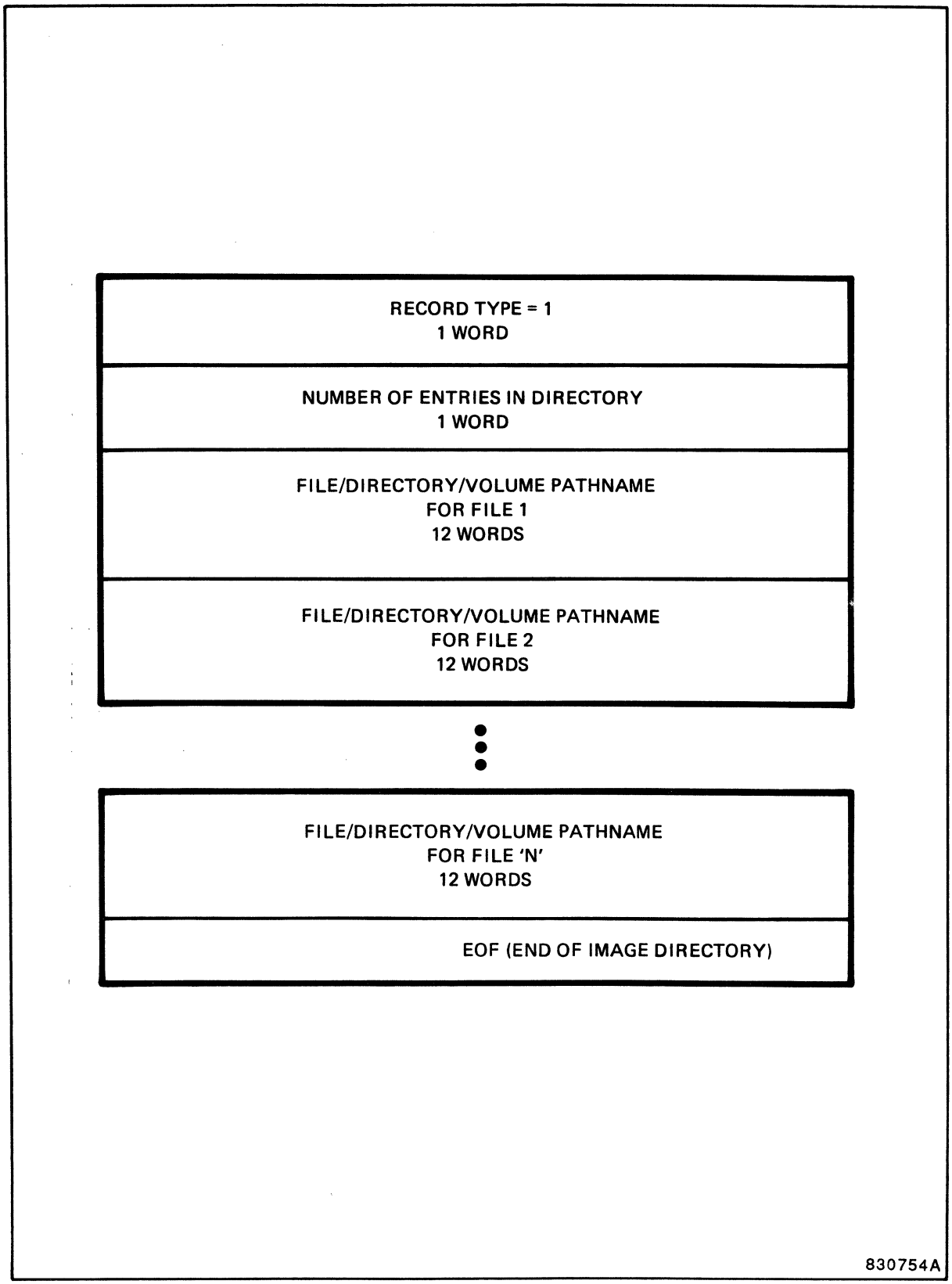


Figure 3-3. Save Image Directory Structure

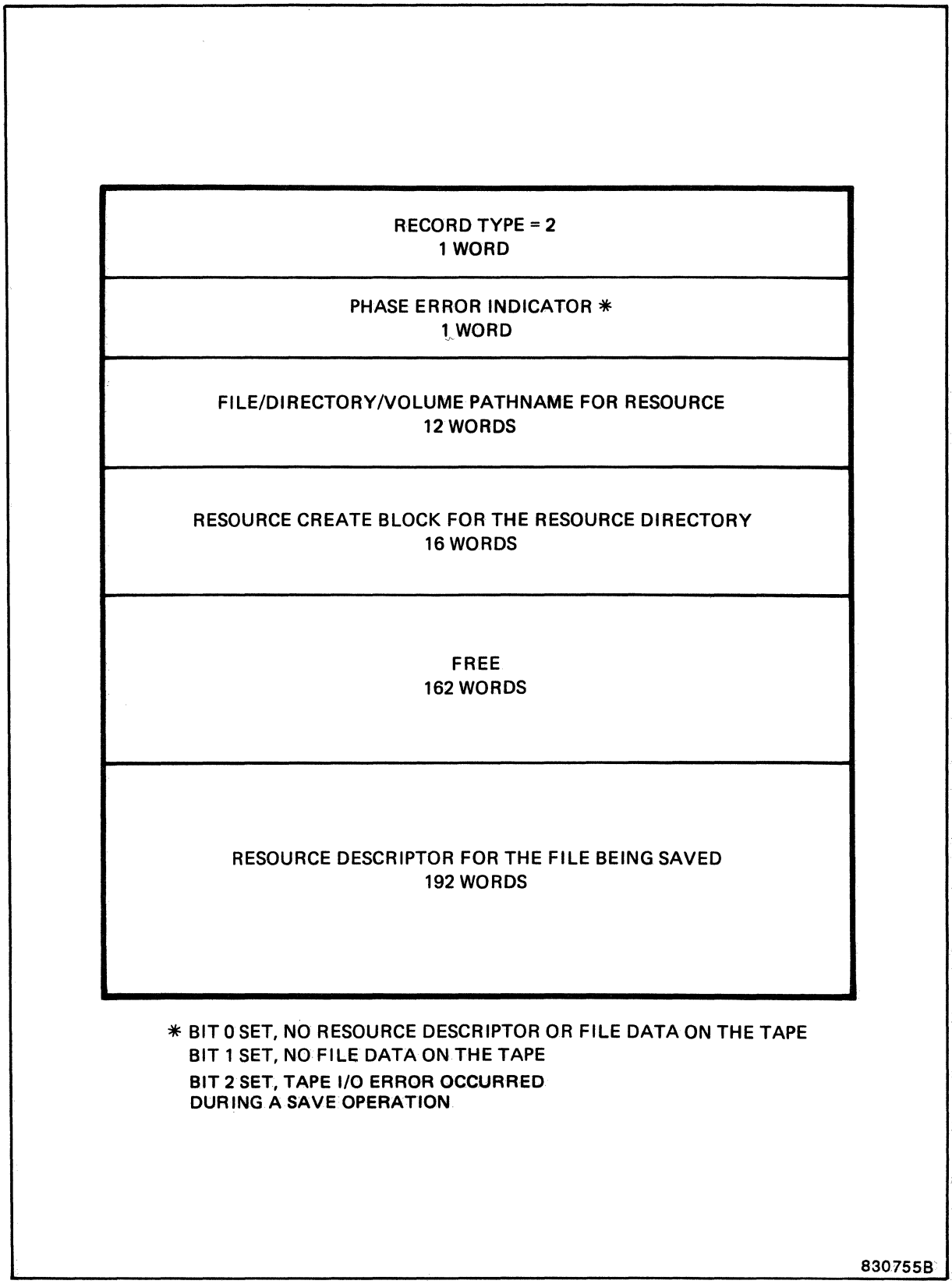


Figure 3-4. Resource Descriptor Tape Record (RDTR)

3.5 Directive Summary

VOLMGR directives are summarized below and described in detail on the following pages. Valid abbreviations are shown by underlining.

Directive	Function
BACKSPACE FILE	Moves a magnetic tape backward a specified number of EOF marks
BACKSPACE IMAGE	Moves a magnetic tape backward a specified number of save images
CLEAR	Clears default option values established by the SET directive
CONVERT	Restores post-RTM 6.0 and all MPX-32 File Manager save tapes to MPX-32 Release 2.x volume/directory structure format
COPY	Copies files to a specified volume or directory
CREATE COMMON	Defines a common area to the system
CREATE DIRECTORY	Creates a directory in the root directory
CREATE FILE	Creates permanent files on disc
DELETE COMMON	Deletes a common area definition from the system
DELETE DIRECTORY	Deletes a directory that does not contain any active entries
DELETE FILE	Deletes a file from a directory
EXIT	Terminates the Volume Manager
EXTEND	Increases the amount of space allocated to an existing file
HELP	Obtains a short description of each VOLMGR directive
LOG FILE	Produces a listing of specified disc files
LOG IMAGE	Produces a listing of all saved files in the current save tape image
LOG RESOURCE	Produces a listing of a resource
LOG SAVEFILE	Produces a Resource Descriptor Tape Record (RDTR) from a save tape
RENAME	Changes the name of an existing file
RESTORE DIRECTORY	Restores files from magnetic tape to disc

RESTORE POSITION	Restores one file from magnetic tape to disc
REWIND	Positions a magnetic tape to beginning of tape
SAVE	Saves files to magnetic tape
SAVE INCREMENTAL	Saves files created or modified since the last save was performed
SDT	Produces a System Distribution Tape
SDT MASTER	Produces a Master System Distribution Tape
SET	Establishes global default option values
SKIP END	Positions a magnetic tape to its logical end-of-tape
SKIP FILE	Moves a magnetic tape forward a specified number of EOF marks
SKIP IMAGE	Moves a magnetic tape forward a specified number of save images
TRUNCATE	Decreases the amount of space allocated to an existing file

VOLMGR directives that create files, directories, or memory partitions have a BRIEF option that controls the amount and type of output generated by the create operation. The remainder of this section describes the format of the output when the BRIEF option is requested or inhibited.

BRIEF=Y or BRIEF=T are synonymous and produce less output than BRIEF=N or BRIEF=F, which are also synonymous. If the BRIEF option is not specified in a directive for which it is available, the default is BRIEF=Y. For directives that do not have a BRIEF option, the resulting output gives the directive being executed, the volume name, directory, file name, and file size. This format is shown in the first example below.

For directives that create or delete files, the resulting display is as follows:

- For BRIEF=Y or T (the default if not specified; also the format that is displayed for directives that do not have the BRIEF option):

  directive @volname^(dirname)filename size

directive	is the directive being performed
volname	is the volume being affected by the directive
dirname	is the directory being affected by the directive
filename	is the file being affected by the directive
size	is the size of the file

. For BRIEF=N or F:

```
directive  pathname
     RID=
     CREATED BY:              ON:                              BLOCKED=
     LAST SAVE=                        , LAST RESTORE=
     LAST CHANGED BY:                ON:
     TYPE=     , SHARE=    , SAVE=    , EOFM=   , ZERO=   , SIZE=
     EXTEND=    , MAXINC=     , MININC=      , MAXSIZE=
     SEGMENTS=       , EOF BLOCK=       , EOM BLOCK=
     ACCESS: OWNER   -
             PROJECT -
             OTHERS
```

directive — is the directive being performed

pathname — is the volume, directory, and file name being affected by the directive

RID= — is the resource identifier which specifies the volume name, binary date and time of creation, resource descriptor address, and resource type (i.e., permanent file, temporary file, etc.)

CREATED BY: — specifies the owner name under which the file was created and the date and time it was created

BLOCKED= — specifies how the file was last written. If =Y, blocked. If =N, unblocked

LAST SAVE= — specifies the date and time the file was last saved by the System Administrator

LAST RESTORE= — specifies the date and time the file was last restored

LAST CHANGED BY: — specifies the owner name which performed the most recent change to the file and the date and time it was changed

TYPE= — specifies the file type code, interpreted as two hexadecimal digits in the range 0 to FF

SHARE= — specifies if the file can be shared

SAVE= — specifies if the file can be saved. If =N and the file is to be saved, SAVN=Y must be specified on the directive line (see the SAVE directive).

EOFM= — specifies the status of EOF management. If =N, end-of-file equals end-of-medium. This attribute is required to be false (i.e., =N), if a block of a file is to be read before it is written to. An example is a hashing algorithm which leads to random access disc I/Os.

ZERO=	specifies if the file is to be zeroed upon creation and extension
SIZE=	specifies the size of the file in blocks
EXTEND=	specifies if the file is extendible. If =AUTO, the file is only automatically extendible. If =MANU, the file is only manually extendible. If =BOTH, the file is both automatically and manually extendible. If =NO, the file is not extendible.
MAXINC=	specifies maximum number of blocks the file can be extended
MININC=	specifies the minimum number of blocks the file can be extended
MAXSIZ=	specifies the maximum block size of the file. If =0, the size is limited only by available disc space.
SEGMENTS=	specifies the number of segments the file contains
EOF BLOCK=	specifies the block number written to if EOFM=Y
EOM BLOCK=	specifies the last physical block number of the file
ACCESS:	specifies the owner and project name of the file and what types of access owner, project, and others have to the file. Access rights are read, write, modify, update, append, and delete.

For directives that create or delete directories, the resulting display is as follows:

. For BRIEF=Y or T (the default if not specified):

directive @volname^(dirname) size

directive	is the directive being performed
volname	is the volume being affected by the directive
dirname	is the directory being affected by the directive
size	is the size of the directory

- For BRIEF=N or F:

```
directive  pathname
           RID=
           CREATED BY:              ON:
           LAST CHANGED BY:                 ON:
           TYPE=     , SHARE=    , EOFM=    , SIZE=   ,
           TOTAL ENTRIES=       , ACTIVE=        , AVAILABLE=
           SEGMENTS=        , EOF BLOCK=        , EOM BLOCK=
           ACCESS: OWNER   -
                   PROJECT -
                   OTHERS
```

directive	is the directive being performed
pathname	is the volume and directory being affected by the directive
RID=	is the resource identifier which specifies the volume name, binary date and time of creation, resource descriptor address, and resource type
CREATED BY:	specifies the owner name under which the directory was created and the date and time it was created
LAST CHANGED BY:	specifies the owner name which performed the most recent change to the directory and the date and time it was changed
TYPE=	specifies the resource type code, interpreted as two hexadecimal digits in the range 0 to FF
SHARE=	specifies if the directory can be shared
EOFM=	specifies the status of EOF management. If =N, end-of-file equals end-of-medium. Directories are automatically created with this attribute as false (EOFM=N).
SIZE=	specifies the size of the directory in blocks
TOTAL ENTRIES=	specifies the number of entries that can be created in the directory, the number of entries that are active, and the number of entries that are available for use
SEGMENTS=	specifies the number of segments the directory contains
EOF BLOCK=	specifies the block number written to if EOFM=Y
EOM BLOCK=	specifies the last physical block number of the directory
ACCESS:	specifies the owner and project name of the directory and what types of access owner, project, and others have to the directory. Access rights are read, delete directory, delete entry, add, and traverse.

For directives that create or delete memory partitions, the resulting display is as follows.

- For BRIEF=Y or T (the default if not specified):

```
directive @volname^(dirname)partname size
```

directive	is the directive being performed
volname	is the volume being affected by the directive
dirname	is the directory being affected by the directive
partname	is the name of the partition being affected by the directive
size	is the size of the partition

- For BRIEF=N or F:

```
directive  partname
           RID=
           CREATED BY:                ON:
           PART TYPE=           , MEM CLASS=        , SHAREABLE=
           TYPE=    , FIRST PAGE=        , LENGTH IN PAGES=
           ACCESS: OWNER    -
                   PROJECT  -
                   OTHERS
```

directive	is the directive being performed
partname	is the volume, directory, and partition name being affected by the directive
RID=	is the resource identifier which specifies the volume name, binary date and time of creation, resource descriptor address, and resource type
CREATED BY:	specifies the owner name under which the partition was created and the date and time it was created
PART TYPE=	specifies if the partition is static or dynamic
MEM CLASS=	specifies if the partition is class E, H, or S memory. The default is class S.
SHAREABLE=	specifies if the partition is shareable
TYPE=	specifies the resource type code, interpreted as two hexadecimal digits in the range 0 to FF
FIRST PAGE=	specifies the first page of the partition
LENGTH IN PAGES=	specifies the size of the partition in pages
ACCESS:	specifies the owner and project name of the partition and what types of access owner, project and others have to the partition. Access rights are read, write, and delete.

3.6 Accessing VOLMGR

VOLMGR is accessed in the batch or interactive modes in one of three ways:

```
$VOLMGR
$RUN VOLMGR
$EXECUTE VOLMGR
```

$RUN VOLMGR is valid only from the system directory.

When logical file code SYC is assigned to a terminal, a VOL > prompt is displayed.

VOLMGR exits when the EXIT directive is issued or when an EOF is read from the directive input file or device. If an EOF is encountered while processing a line continuation, VOLMGR exits with an error condition. If any errors are encountered during directive processing, VOLMGR posts an error condition (VO01) upon exiting.

A one-shot invocation of VOLMGR can be performed in the batch and interactive modes. One-shot invocation is performed by entering the load module name VOLMGR followed by a VOLMGR directive on the same line. If this is done from TSM, control automatically returns to TSM after the VOLMGR operation is complete and its output is displayed. In the batch mode, processing continues with the next job control statement.

3.7 File and Directory Size Allocations

When using the CREATE FILE or COPY directives and initial block allocation size is specified, the number specified is rounded up to the nearest allocation unit. The allocation unit is determined by the storage capacity of the disc. On an 80MB disc, the minimum allocation size is two blocks. On a 300MB disc, the minimum allocation size is four blocks. Therefore, if the size specified is SIZE = 1, the rounded up result would be two on an 80MB disc and four on a 300MB disc.

Similarly, when using the CREATE DIRECTORY directive and the maximum number of files to be listed within the directory is specified, the number specified is rounded up to the nearest allocation unit (maximum of 12 entries per block on all disc types). Therefore, if the number of entries specified is ENTRIES = 10, the rounded up result would be 24 on an 80MB disc (12 per block times 2 blocks) and 48 on a 300MB disc (12 per block times 4 blocks).

3.8 File and Directory Size Extensions

A file can be extended using the EXTEND directive until 32 segments have been filled. To extend a file more than 31 times or to restore file contiguity, the file must be saved with the SAVE directive and restored to disc with the RESTORE directive. By default, the restore operation will recreate the file with the original file size equal to the end-of-medium block of the file saved to tape. The restored file is contiguous, by default.

A directory cannot be extended. If the size of a directory must be increased, the directory must be deleted and then recreated with the appropriate number of entries required specified with a CREATE DIRECTORY directive. If the directory contains active entries, the active entries must be temporarily renamed to another directory by a RENAME directive while the directory is being recreated.

3.9 Pathnames and Directories

A pathname is a one- to three-part name that identifies the path to be taken to a volume, directory, or file. Wherever file names are valid, a complete pathname can be specified, or missing portions of a pathname are assigned defaults.

Wild card indicators can be used in the pathname for characters by specifying a question mark to replace individual characters or an asterick to replace a string of characters. The special meaning given to the wild card indicators is recognized by VOLMGR only, not by the operating system.

Wild card characters used in TO pathname parameters are interpreted positionally from left to right. See examples of the COPY, RENAME, and RESTORE DIRECTORY directives for details.

In a directive that explicitly or implicitly references a directory, the default directory is the current working directory unless overridden in the directive. Directories can only be created or deleted. Directives such as COPY, SAVE, and RESTORE cannot be used on directories.

Special characters separate each part of a pathname to specify which portion of the pathname the various names represent. Each name is a set of standard characters normally restricted to contain alphanumeric characters plus the special characters dot and underscore. If it is necessary to reference a file containing nonstandard characters, the file name must be enclosed in single quotes. This permits reference to all files except those containing the actual single quote character. Quoting a file name disables wild card capabilities because the wild card indicators ? and * may be part of a file name.

For example, DELETE 'AS*WRKFL' deletes the file named AS*WRKFL from the current working directory. However, DELETE *WRKFL deletes any file in the current working directory with WRKFL as the last five characters of the file name.

3.10 Logical File Code Assignments

Logical file code (LFC) assignments are specified in $ASSIGN job control statements before activating VOLMGR. VOLMGR uses three LFCs: SYC for directive input, SLO for listed output, and TAP for saving and restoring files.

3.10.1 Directive Input (SYC)

VOLMGR directives are assigned to logical file code SYC. The default assignment for SYC is:

```
$ASSIGN SYC TO SYC
```

In the interactive mode, directives are read from the terminal.

Optional assignments for SYC are:

```
$ASSIGN SYC TO {pathname  }
               {DEV=devmnc}
```

pathname	is the pathname of a file containing VOLMGR directives
devmnc	is the device mnemonic of a device containing VOLMGR directives

3.10.2 System Listed Output (SLO)

The listed output file contains an audit trail of the directives processed by VOLMGR during an interactive or batch session. Listed output is assigned to logical file code SLO. The default assignment for SLO is to logical file code UT:

```
$ASSIGN SLO TO LFC=UT
```

In the interactive mode, output is displayed at the user terminal. In the batch mode, output is directed to the SLO device.

Optional assignments for SLO are:

```
$ASSIGN SLO TO {pathname    }
               {DEV=devmnc  }
```

pathname is the pathname of a previously created file to contain VOLMGR listed output

devmnc is the device mnemonic of a device to contain VOLMGR listed output

3.10.3 Magnetic Tape (TAP)

All VOLMGR directives associated with saving or restoring files manipulate the device assigned to logical file code TAP. If the device assigned to TAP is not a magnetic tape, it must have the ability to be treated as a magnetic tape in that EOF or pseudo-EOF capabilities must be present when unblocked data is written.

Logical file code TAP is assigned as follows:

```
$ASSIGN TAP TO DEV=devmnc BLOC=N
```

devmnc is the device mnemonic of the device on which to perform the save or restore operation

3.10.4 Wildcard Characters

When a wildcard character (*) is specified in any command, the wildcard is converted to actual volume/directory/file names through the use of the temporary files. The logical file codes of these temporary files are I01, I02 and I03. The default sizes for these three temporary files are:

original size	200 blocks
maximum increment	400 blocks
minimum increment	100 blocks

In optimum conditions, VOLMGR can resolve up to 75,600 files. If the disc space is too fragmented to extend any segment, VOLMGR can resolve 1200 files of the original size.

3.11 Options

The VOLMGR is cataloged with the TSM TEXT option to echo input to the user's terminal or SLO as it is read from the SYC file and with the PROMPT option to write a VOL> prompt to the terminal when input is required.

3.12 Directives

A VOLMGR directive line consists of a directive verb. adverb, parameters. and options. A comma or blank delimits each component. Comments can be included by inserting an exclamation point anywhere on the line. The remainder of that line is ignored. An entire line may be a comment by placing an exclamation point as the first character of the line.

3.12.1 Verbs and Adverbs

The verb is the first item on the directive line. When a verb can perform more than one function, an adverb specifies which function to perform. For example:

```
CREATE COMMON
CREATE DIRECTORY
CREATE FILE
```

CREATE is the verb and COMMON, DIRECTORY, and FILE are adverbs.

Default adverbs are provided where applicable. Refer to the individual directive descriptions for their defaults.

3.12.2 Parameters

Parameters specify what is affected by the action of the directive, such as a pathname. directory name, or device name. Parameters are specified in positional order. Each parameter is optionally preceded by a keyword and equal sign. Multiple parameter groups are separated by commas.

3.12.3 Directive Options

Options can be specified in any order and are always specified by their keyword, an equal sign, and the value. Options can be specified at three levels:

- Default (lowest precedence) - Values are specified with the SET directive. This default value is applicable only when global or local option values have not been specified.
- Global (directive) - Values are specified on the directive line between the directive verb (adverb if present) and the first directive parameter. Global option values override default values.
- Local (parameter, highest precedence) - Values are specified on the directive line between the rightmost parameter and the terminator character. Local option values override global and default values.

Options have three types of values:

- Boolean - Values are true (T), false (F), yes (Y), and no (N). Values true and yes are synonymous. Values false and no are synonymous.

- Numeric - Values are either decimal or hexadecimal numbers, whichever is natural for a particular option. For example, file size is expressed as a decimal number (SIZE=N'99') but file type is expressed as a hexadecimal number (TYPE=X'A0'). Default values are decimal.

- ASCII String - Values are expressed as strings containing alphanumeric characters, plus the characters dot and underscore. If any other characters are used in the string, the entire string must be enclosed in single quotes. For example, OWNER=JOHN_DOE or OWNER='JOHN DOE' are acceptable. In either case, the ASCII value starts with the first nonblank character after the equal sign.

3.12.3.1 Global Options

Global options are entered on the directive line between the verb (adverb if present) and the first parameter group. The first parameter group includes all parameters and local options up to the first comma or to the end of the string if there is only one parameter). For example:

```
CREATE FILE SIZE=100 SJ.VOLMGR
```

CREATE is the verb, FILE is the adverb, SIZE=100 is the global option, and SJ.VOLMGR is the parameter.

To create more than one file of the same size, enter:

```
CREATE FILE SIZE=100 INDEX,POCKET,GLOSS
```

All three files will be created as 100 blocks in size.

3.12.3.2 Local Options

Local options are entered on the directive line between parameter groups. They are expressed as the option name keyword followed by an equal sign and the option value. Syntactically, local options are expressed the same as global options. The position in the directive line determines whether an option is global or local. VOLMGR distinguishes options from parameters by the keyword identity. For example, in the following directive line:

```
CREATE FILE SIZE=100 INDEX SIZE=50,POCKET,GLOSS
```

CREATE is the verb, FILE is the adverb, SIZE=100 is a global option, INDEX is a parameter (its file size is 50 blocks), SIZE=50 is a local option, and POCKET and GLOSS are parameters (their file size is 100 blocks).

If a default file size is established with the SET directive, the results would be as follows:

```
SET SIZE=50
CREATE FILE INDEX,POCKET,GLOSS
```

All three files are created as 50 blocks in size. In the following example:

```
SET SIZE=50
CREATE FILE INDEX SIZE=100,POCKET,GLOSS SIZE=150
```

File INDEX is created as 100 blocks in size, file POCKET is created with the default size of 50 blocks, and file GLOSS is created as 150 blocks in size.

3.12.3.3 Time Options

Some VOLMGR directives can be used with an optional time specification restriction. The following rules apply to time specification usage.

The system date is maintained as the number of days since January 1, 1960. Entering any earlier date results in an error condition.

Only one time option (CREATED, CHANGED, SAVED, or RESTORED) can be specified per partition or pathname.

Regardless of the format used as input for a date option, the date will be displayed in the format the date was entered at IPL.

Syntax options for date are:

```
mo/dd/yy
dd-mon-yy
```

Syntax options for time are:

```
hh:mm:ss
hh:mm
h
```

mo is the one- or two-digit decimal month

mon is the three-character ASCII month abbreviation

dd is the one- or two-digit decimal day

yy is the two-digit decimal year

hh is the one- or two-digit decimal hour (24-hour time)

mm is the one- or two-digit decimal minute

ss is the one- or two-digit decimal second

Leading zeros are not required in date or time.

The following rules apply to SINCE or BEFORE usage:

- Both date and time can be specified as shown in the individual directive syntax descriptions. At least one of them must be specified.

- If the time is specified but the date is not, a comma is not needed. The date defaults to the current date.
- If the date is specified but the time is not, the time defaults to midnight.
- All date formats (with or without leading zeros) apply even though only one is shown in the directive syntax description.
- If any part of the time is not specified, that part defaults to zero.

3.13 Directive Line Continuations

Directive lines can continue across as many lines as necessary by placing a hyphen as the last significant character on the line. This results in a prompt for more input when in the interactive mode, and another read when in batch mode or when reading a select file. Each time a directive line is continued, the continuation character is replaced by a space.

Up to 48 parameter prototypes can be specified per directive. A complete directive line can have a maximum of 768 characters. A directive that has been continued across several lines appears as one long string to the VOLMGR. All excess blanks are eliminated, so where more than one blank is used as a delimiter, only one blank is retained. This also pertains to trailing blanks.

The only restriction is that single names, keywords, or an entire pathname must appear on the same input line.

3.14 BACKSPACE FILE Directive

The BACKSPACE FILE directive backspaces a magnetic tape a specified number of EOF marks.

Syntax:

BACKSPACE FILE [[FILES=]n]

FILES= specifies the number of EOF marks to backspace. If not specified, the tape is backspaced one EOF.

Usage:

BAC F

Backspaces the magnetic tape one EOF mark.

BAC F FILE=2

BAC F 2

Both of these directives backspace the magnetic tape two EOF marks.

3.15 BACKSPACE IMAGE Directive

The BACKSPACE IMAGE directive backspaces a magnetic tape a specified number of save images.

Syntax:

BACKSPACE [IMAGE] [[IMAGES=]n]

IMAGES= specifies the number of save images to backspace. If not specified, the tape is backspaced one save image.

Usage:

BAC

Backspaces the magnetic tape one save image.

BAC IMAG=2

BAC I 2

BAC 2

All of these directives backspace the magnetic tape two save images.

3.16 CLEAR Directive

The CLEAR directive clears all default option values previously established by a SET directive.

Syntax:

CLEAR

3.17 CONVERT Directive

The CONVERT directive restores post-RTM 6.0 and all MPX-32 File Manager save tapes to Release 2.x volume/directory structure format. Wild card characters cannot be used with this directive.

The device where the data to convert reside must be specified by a static assignment in job control. The logical file code TAP is presumed to have been assigned by the user before invoking VOLMGR.

Syntax:

CONVERT [[VOLUME=]volname] [BRIEF=bool]

VOLUME= specifies the name of the volume to place the file on. If not specified, defaults to the user's current working volume. Partial conversions of a save image are not allowed. For example, files cannot be selected by name.

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

Notes:

Data in the old save image directory is converted as follows:

1. Volume name - The name of the volume where data is to be restored can be specified on the directive line. If a volume name is not specified, the default is to the user's current working volume.

   Any one pre-MPX-32 Release 2.0 save image should contain only those files to be converted to the same volume. This can be done by saving pre-MPX-32 Release 2.0 files with the File Manager (FILEMGR) utility.

2. Directory name - The directory name is the same as the old user name. Directories are created if needed. However, if the VOLMGR creates the directory, the entry capacity cannot be user-specified.

3. File name - File names are unchanged. Files cannot be renamed during the conversion. If a file exists on the volume with the same name as one to be converted, VOLMGR attempts to delete the old file. If the delete fails, the saved file is not converted (restored).

New file attributes are as follows:

1. Owner name - The owner name is the same as the current logon owner name if a default name has not been established by a SET directive. If a default owner name has been established by a SET directive, that name is used.

2. Project group name - The project group name is the same as the current logon project group name if a default name has not been established by a SET directive. If a default project group name has been established by a SET directive, that name is used.

3. Access attributes - All converted files are completely unprotected for owner and project group. The established system access defaults apply to other.

Usage:

```
CON VOLU=TEST BRIE=N
```

All files in the save image are converted and stored on volume TEST. Their complete file status will be displayed.

```
CON
```

All files in the save image are converted and stored on the user's current working volume. Only their file name and size will be displayed.

3.18 COPY Directive

The COPY directive creates a file by copying an existing file. A file can be copied to the same volume or a different volume than the one on which it is currently located. Any options specified apply to the file being created. Wild card characters can be used with this directive.

The COPY directive cannot be used to concatenate several files into one.

Syntax:

COPY [FROM=]pathname [TO [=]]pathname [SIZE=n] [OWNER=name]

[PROJECTGROUP=name] [START=n] [MAXINC=n] [MININC=n]

[MAXSIZE=n] [FAST=bool] [BRIEF=bool] [REPLACE=bool]

[SHARED=bool] [NOSAVE=bool] [ZERO=bool] [EOFM=bool]

[AUTOEXT=bool] [MANEXT=bool] [CONTIGUOUS=bool]

```
[ ACCESS { OWNER        }([READ] [WRITE] [UPDATE] [MODIFY] [APPEND]  ]
         { PROJECTGROUP }[DELETE])                                   ]
         { OTHER        }                                            ]
```

FROM= specifies the pathname of the file to be copied

TO= specifies the pathname of the file to be created

SIZE= specifies the initial block allocation size of the file being created. If not specified, defaults to the size of the file specified in the FROM parameter.

OWNER= specifies the owner name to be associated with the file. If not specified, defaults to the owner name associated with the VOLMGR task.

PROJECTGROUP= specifies the project group name to be associated with the file. If not specified, defaults to the project group associated with the VOLMGR task.

START= specifies the starting block number of the file. If the file cannot be copied to the specified block number, a denial is given. If not specified, the file is copied to any available space on the volume.

MAXINC= specifies the size of each subsequent automatic increment of the file. If not specified, the default is 64 blocks.

MININC= specifies the minimum acceptable automatic increment size if the MAXINC value cannot be obtained. If not specified, the default is 32 blocks.

MAXSIZE= specifies the maximum size for an extendible file. If not specified, the size is limited only by available disc space.

FAST= specifies if the file's resource identifier is not to be changed when it is copied or restored using the VOLMGR. If not specified, the file will be copied to any available resource identifier.

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

REPLACE= If the file name specified in the TO parameter already exists, indicates whether it should be replaced (deleted). If not specified, a delete is not attempted on an existing file, and nothing is copied.

SHARED= specifies if the file can be used in the shared mode. If not specified. the default is shared.

NOSAVE= specifies if the VOLMGR no-save attribute applies to the file. If not specified, the file is savable in response to the VOLMGR SAVE directive.

ZERO= specifies if the file should be zeroed upon creation and extension. If not specified, the file is not zeroed.

EOFM= specifies the status of EOF management. If not specified. the system default is used.

AUTOEXT= specifies if the file can be automatically extended. If not specified, the file is automatically extendible.

MANEXT= specifies if the file can be manually extended. If not specified. the file is manually extendible.

CONTIGUOUS= specifies if extensions to the file are to be contiguous when possible. Initial allocation of a file is always contiguous. If not specified, extensions are not contiguous.

ACCESS= specifies the classes of users allowed access to the file and the types of access each class is allowed. If not specified, owner and project group have complete access and others have read only access.

The destination file (TO) is always (re)created. If the TO file already exists, the copy cannot take place unless the REPLACE option has been specified as true or yes. If the REPLACE option is specified, VOLMGR will attempt to delete the existing TO file. If the delete operation is not successful, a copy will not take place.

All of the TO file's attributes are implicitly the same as the FROM file's attributes, except the owner name and project group name will be those associated with the VOLMGR task. Also, the TO file's access rights will be the system default. Any of the TO file's attributes can be explicitly stated on the directive line to override the implicit values.

Usage:

```
COP FROM=@TEST(DIR1)INDEX TO=@TEST(DIR2)INDEX-
PROJ=011657 REPL=Y OWNE=USER1
ACCE=OW(R W U M A D)
```

Copies file INDEX on volume TEST in directory DIR1 to file INDEX on volume TEST in directory DIR2. The project group name is 011657, the existing version of it is replaced, the owner name is USER1, and the owner has complete access privileges.

```
COP TEMP PERM
```

Copies file TEMP from the current working volume and directory to file PERM in the current working volume and directory. Attributes for file PERM are the same as they were for file TEMP except that owner and project group is the name that is associated with the VOLMGR task. If a file named PERM already exists, the directive is invalid.

```
COP @ATLAS04(TEST)* @ATLAS02(TEST)* REPL=Y SHAR=Y
```

Copies all files on volume ATLAS04 in directory TEST to volume ATLAS02 in directory TEST. If any of them currently exist, they are replaced and all files are shareable.

```
COP @ATLAS04(MYDIR)SJ.* @ATLAS04(YOURDIR)* REPL=N
```

Copies all files on volume ATLAS04 in directory MYDIR that begin with the characters SJ. to volume ATLAS04 and directory YOURDIR. If any of the files currently exist, they are not replaced.

```
COPY @SOURCE(*)* @DESTINATION(*)SJ*
```

Copies all files on volume SOURCE to volume DESTINATION. The directories on volume DESTINATION are assumed to have been previously created. The first two characters of each file name are changed to SJ.

```
COPY @V1BACKUP(SYSTEM)* @V1(SYSTEM)*SJ
```

Copies all files on volume V1BACKUP in directory SYSTEM to volume V1 in directory SYSTEM. The characters SJ are appended to each file name.

3.19 CREATE COMMON Directive

The CREATE COMMON directive defines a global common partition, a Datapool partition, or a partition in the user's extended address space. Memory partitions defined are dynamically allocated when required by the task. They do not remain allocated in memory, however, like those defined by SYSGEN. Once a partition is defined by this directive. tasks can include the partition by calling the M.INCLUDE service. Refer to MPX-32 Reference Manual, Volume I, Chapter 3 for information on resource allocation.

Wild card characters cannot be used with this directive.

Syntax:

```
CREATE COMMON  [PATH=]pathname  [PROTGRAN=]n  [FIRSTPAGE=]n

[PROJECTGROUP=name]  [OWNER=name]  [REPLACE=bool]

[MEMCLASS={E|H|S}]  [BRIEF=bool]

[ACCESS={OWNER|PROJECTGROUP|OTHER}  ([READ] [WRITE] [DELETE])]
```

PATH= specifies the pathname of the partition being created in one of the following three forms: GLOBALnn, DATAPOOL, or extname.

If GLOBALnn, a global common partition (00-99) is created that is physically located in any class of memory (E, H, or S).

If DATAPOOL, a Datapool partition is created whose structure is defined by one or more Datapool dictionaries. Like global common, the datapool area is physically located in any class of memory (E, H, or S).

If extname, a one- to eight-character name is specified to use for a memory partition in a task's extended address space. This partition may be mapped into memory above the first 128KW logical address space available to a task. Since the partition is in extended memory, certain restrictions will apply. Partitions in a task's extended address space is physically located in any class of memory (E, H, or S).

PROTGRAN= specifies the number of 512-word protection granules to include in the partition. On a CONCEPT/32 computer, one map block is four protection granules. Unused physical protection granules within the last 2KW map block on a CONCEPT/32 computer allocated to the partition is write-protected from all sharing tasks. However, only one dynamic partition can be defined in any one map block.

FIRSTPAGE= specifies the starting protection granule where the partition is to be mapped (pages 0-1020 for the 32/27 and 32/87, pages 0-8188 for the 32/67 and 32/97). Protection granules in the first several map blocks should not be specified because they are used for the MPX-32 operating system. Protection granules located after the last loaded address of the operating system should be specified.

Protection granules for global and Datapool partitions are normally allocated from top down in a task's logical address space, or below any SYSGEN-created common partitions. In extended address space, the top map block is reserved for MPX-32 use.

PROJECTGROUP= specifies the project group name to be associated with the partition. If not specified, defaults to the project group associated with the VOLMGR task.

OWNER= specifies the owner name to be associated with the partition. If not specified, defaults to the owner name associated with the VOLMGR task.

REPLACE= if the partition name already exists, indicates whether it should be replaced (deleted). If not specified, a delete is not attempted on an existing partition and nothing is created.

MEMCLASS= specifies the class of memory for the partition. If not specified, the default is class S.

BRIEF= specifies if complete partition status is desired. If not specified, only the partition name and size are displayed.

ACCESS= specifies the classes of users allowed access to the partition and the types of access each class is allowed. If not specified, owner and project group have complete access and others have read only access.

Usage:

```
CRE C PATH=GLOBAL55 PROT=5 FIRS=75 PROJ=082950
```

Creates partition GLOBAL55 on the current working volume and directory. It is five pages in length and begins on page 75 in memory. The project group name associated with it is 082950.

```
CRE C DATAPOOL 5 75 OWNE=USER1 PROJ=061643 ACCE=OW(R W D)
```

Creates partition DATAPOOL on the current working volume and directory. It is five pages in length and begins on page 75 in memory. The owner name associated with it is USER1, the project group name is 061643, and the owner has complete access privileges.

```
CRE C @SYSTEM(SYSTEM)ATLAS 5 75 MEMC=H OWNE=USER2 REPL=Y-
ACC=OW ( R W D)
```

Creates partition ATLAS on the SYSTEM volume and directory. It is five pages in length and begins on page 75 in memory. It is class H memory, its associated owner name is USER2, and the owner has complete access privileges. If a partition currently exists by the same name it is replaced.

3.20 CREATE DIRECTORY Directive

The CREATE DIRECTORY directive creates an entry for a directory in the volume directory structure. Wild card characters cannot be used with this directive.

Syntax:

CREATE DIRECTORY [PATH=]pathname [ENTRIES=n] [BRIEF=bool]

[OWNER=name] [PROJECTGROUP=name] [SHARED=bool]

[ACCESS={OWNER / PROJECTGROUP / OTHER} ([READ] [DELDIR] [DELENT] [ADD] [TRAVERSE])]

PATH=	specifies the pathname to be associated with the directory
ENTRIES=	specifies the maximum number of files to be created in the directory. If not specified, the volume default is used (determined by disc allocation granularity).
BRIEF=	specifies if complete file status is desired. If not specified, only the file name and size are displayed.
OWNER=	specifies the owner name to be associated with the directory. If not specified, defaults to the owner name associated with the VOLMGR task.
PROJECTGROUP=	specifies the project group name to be associated with the directory. If not specified, defaults to the project group associated with the VOLMGR task.
SHARED=	specifies if the directory can be used in the shared mode. If not specified, the default is shared.
ACCESS=	specifies the classes of users allowed access to the directory and the types of access each class is allowed. If not specified, owner has complete access, project group has all but delete access, and others have read only and traverse access.

Usage:

```
CRE D PATH=TEST OWNE=USER1 ENTR=150
```

Creates directory TEST on the current working volume with owner name USER1. The directory can contain 150 files, and all other system defaults apply to the directory.

```
CRE D TEST2 BRIE=N PROJ=061643 SHAR=N OWNE=USER2-
ACCE=OW(R DELD DELE A T) ACCE=PR(R) ACCE=OT()
```

Creates directory TEST2 on the current working volume. The directory's full status is displayed. The project group name is 061643 and it is not shareable. The owner name is USER2 and the owner has full access privileges. The project group has read-only access and others have no access privileges.

3.21 CREATE FILE Directive

The CREATE FILE directive creates a file on disc. Wild card characters cannot be used with this directive.

Syntax:

CREATE [FILE] [PATH=]pathname [SIZE=n] [BRIEF=bool] [OWNER=name]

[PROJECTGROUP=name] [START=n] [MAXINC=n] [MININC=n] [MAXSIZE=n]

[FAST=bool] [REPLACE=bool] [SHARED=bool] [NOSAVE=bool]

[ZERO=bool] [EOFM=bool] [AUTOEXT=bool] [MANEXT=bool]

[CONTIGUOUS=bool] [SEGNUM = n]

[ACCESS={OWNER / PROJECTGROUP / OTHER} ([READ] [WRITE] [UPDATE] [MODIFY] [APPEND] [DELETE])]

PATH= specifies the pathname of the file to be created

SIZE= specifies the initial block allocation size of the file. If not specified, the default is 16 blocks.

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

OWNER= specifies the owner name to be associated with the file. If not specified, defaults to the owner name associated with the VOLMGR task.

PROJECTGROUP= specifies the project group name to be associated with the file. If not specified, defaults to the project group associated with the VOLMGR task.

START= specifies the starting block number of the file. If the file cannot be created at the specified block number, a denial is given. If not specified, the file is created in any available space on the volume.

MAXINC= specifies the size of each subsequent automatic increment of the file. If not specified, the default is 64 blocks.

MININC= specifies the minimum acceptable automatic increment size if the MAXINC value cannot be obtained. If not specified, the default is 32 blocks.

MAXSIZE= specifies the maximum size for an extendible file. If not specified, the size is limited only by available disc space.

FAST= specifies if the file's resource identifier is not to be changed when it is copied or restored using the VOLMGR. If not specified, the file is copied to any available resource identifier.

REPLACE= if the file name specified in the PATH parameter already exists. indicates whether it should be replaced (deleted). If not specified. a delete is not attempted on the already existing file, and nothing is created.

SHARED= specifies if the file can be used in the shared mode. If not specified, the default is shared.

NOSAVE= specifies if the VOLMGR no-save attribute applies to the file. If not specified, the file is savable in response to the VOLMGR SAVE directive.

ZERO= specifies if the file should be zeroed upon creation and extension. If not specified, the file is not zeroed.

EOFM= specifies the status of EOF management. If not specified, the default is yes.

AUTOEXT= specifies if the file can be automatically extended. If not specified, the file is automatically extendible.

MANEXT= specifies if the file can be manually extended. If not specified, the file is manually extendible.

CONTIGUOUS= specifies if extensions to the file are to be contiguous when possible. Initial allocation of a file will always be contiguous. If not specified, extensions are not contiguous.

SEGNUM= specifies the maximum acceptable segment numbers at creation time if one contiguous file space cannot be obtained. If not specified, the default is one segment.

ACCESS= specifies the classes of users allowed access to the file and the types of access each class is allowed. If not specified, owner and project group have complete access and others have read only access.

Notes:

Because the keyword FILE is optional for this directive, an ambiguity could arise when trying to create a file named FILE. For example:

(1) CREATE FILE PATH=@WORKING(USER1)FILE

and

(2) CREATE FILE

In (1), the directive is explicitly stated and the file would be created. In (2), the current working directory and volume are implied and ambiguity arises in the meaning of the string FILE. Is FILE the directive adverb or is it the file name? Since the adverb precedes the file name on the directive line, FILE will be interpreted as the adverb and the file name is not found, resulting in an error condition. To create a file named FILE, (1) above would have to be specified or:

```
CREATE PATH=FILE
```

(or)

```
CREATE F FILE
```

Usage:

```
CRE PATH=PROG1 SIZE=25 FAST=Y AUTO=Y
```

Creates file PROG1 on the current working volume and directory as 25 blocks in size. Its resource identifier is not changed when the file is copied or restored using the VOLMGR, and it is automatically extendible.

```
CRE PROG1 BRIE=N OWNE=USER2 ACCE=OW(R W U M A D)
```

Creates file PROG1 on the current working volume and directory. Its entire status is displayed, its owner is USER2, and the owner has full access privileges.

3.22 DELETE COMMON Directive

The DELETE COMMON directive deletes a specified common area from the system. Wild card characters can be used with this directive.

Syntax:

DELETE COMMON [PATH=] pathname [CONFIRM=bool] [BRIEF=bool]

[CREATED= {SINCE mo/dd/yy,hh:mm:ss | BEFORE dd-mon-yy,hh:mmss}]

PATH= specifies the pathname of the partition to be deleted

CONFIRM= specifies if confirmation is to be requested before deleting the partition. Operator response is required to the confirmation prompt. If not specified, confirmation is not given. This option is valid in the interactive mode only.

BRIEF= specifies if complete file status is desired. If not specified, only the partition name and size are displayed.

CREATED= if S is specified, all common areas created since the specified date and time are deleted. If B is specified, all common areas created before the specified date are deleted. If not specified, creation date and time are ignored.

Usage:

```
DEL C TEST2 CREA=S 3/31/81,08
```

Deletes partition TEST2 on the current working volume and directory if it was created since March 31, 1981 at 8:00 a.m.

```
DEL C CASE1 CREA=B 23-APR-81
```

Deletes partition CASE1 on the current working volume and directory if it was created before midnight on April 23, 1981.

```
DEL C PATH=SYSTEM(TEST)JOBA BRIE=N CONF=Y CREA=B 08:00:00
```

Deletes partition JOBA on the SYSTEM volume in directory TEST if it was created before 8:00 a.m. today. A confirmation message and the full status of the partition are displayed.

3.23 DELETE DIRECTORY Directive

The DELETE DIRECTORY directive deletes a directory that does not contain any active entries. Wild card characters can be used with this directive.

Syntax:

```
DELETE DIRECTORY [PATH=]pathname [CONFIRM=bool] [BRIEF=bool]
[ CREATED=  { SINCE mo/dd/yy,hh:mm:ss      } ]
            { BEFORE dd-mon-yy,hh:mm:ss    }
[ CHANGED=  { SINCE mo/dd/yy,hh:mm:ss      } ]
            { BEFORE dd-mon-yy,hh:mm:ss    }
```

PATH= specifies the pathname of the directory to be deleted

CONFIRM= specifies if confirmation is to be requested before deleting the directory. Operator response is required to the confirmation prompt. If not specified, confirmation is not given. This option is valid in the interactive mode only.

BRIEF= specifies if complete directory status is desired. If not specified, only the directory name and size are displayed.

CREATED= if S is specified, all directories created since the specified date and time are deleted. If B is specified, all directories created before the specified date are deleted. If not specified, creation date and time are ignored.

CHANGED= if S is specified, all directories changed since the specified date and time are deleted. If B is specified, all directories changed before the specified date are deleted. If not specified, changed date and time are ignored.

Usage:

```
DEL D APPJ BRIE=N
```

Deletes directory APPJ on the current working volume and displays its full status.

```
DEL D GLOSS CREA=S 04/29/81,14:30
```

Deletes directory GLOSS on the current working volume if it was created since April 29, 1981 at 2:30 p.m.

```
DEL D PATH=@VOL1^GLOSS CONF=Y
```

Deletes directory GLOSS on volume VOL1 and displays a confirmation message.

3.24 DELETE FILE Directive

The DELETE FILE directive deletes a file from a directory and deallocates the file space. To use this directive, the owner name running the VOLMGR task must have delete access to the file to be deleted and delete entry access to the directory where the file is located. Wild card characters can be used with this directive. If wild card characters are used, traverse access to the directory is required.

Syntax:

```
DELETE [FILE] [PATH=] pathname [CONFIRM=bool] [BRIEF=bool]
[ ( CREATED=   { SINCE  mo/dd/yy,hh:mm:ss  }
               { BEFORE dd-mon-yy,hh:mm:ss }
  | CHANGED=   { SINCE  mo/dd/yy,hh:mm:ss  }
               { BEFORE dd-mon-yy,hh:mm:ss }
  | SAVED=     { SINCE  mo/dd/yy,hh:mm:ss  }
               { BEFORE dd-mon-yy,hh:mm:ss }
  | RESTORED=  { SINCE  mo/dd/yy,hh:mm:ss  }
               { BEFORE dd-mon-yy,hh:mm:ss } ) ]
```

PATH= specifies the pathname of the file to be deleted

CONFIRM= specifies if confirmation is to be requested before deleting the file. Operator response is required to the confirmation prompt. If not specified, confirmation is not given. This option is valid in the interactive mode only.

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

CREATED= if S is specified, all files created since the specified date and time are deleted. If B is specified, all files created before the specified date are deleted. If not specified, creation date and time are ignored.

CHANGED= if S is specified, all files changed since the specified date and time are deleted. If B is specified, all files changed before the specified date are deleted. If not specified, changed date and time are ignored.

SAVED= if S is specified, all files saved since the specified date and time are deleted. If B is specified, all files saved before the specified date are deleted. If not specified, saved date and time are ignored.

RESTORED= if S is specified, all files restored since the specified date and time are deleted. If B is specified, all files restored before the specified date are deleted. If not is specified, restored date and time are ignored.

Usage:

```
DEL PATH=TEMP CONF=Y REST=S 5/31/81
```

Deletes file TEMP on the current working volume and directory if it was restored since midnight, May 31, 1981 and displays a confirmation message.

```
DEL S??? CONF=Y
```

Deletes all files on the current working volume and directory with four-character file names that begin with S. A confirmation message is displayed for each file that is deleted.

```
DEL @USER1(USER1)*WRKFL CONF=Y
```

Deletes all files ending in WRKFL on volume USER1 in directory USER1. A confirmation message is displayed for each file that is deleted.

3.25 EXIT Directive

The EXIT directive exits the VOLMGR and returns control to TSM.

Syntax:

```
EXIT
```

3.26 EXTEND Directive

The EXTEND directive increases the amount of space allocated to a file. The file contents are not affected and EOF marks remains the same. Wild card characters can be used with this directive.

Syntax:

EXTEND [PATH=] pathname [EXTSIZE=n] [BRIEF=bool]

PATH= specifies the pathname of the file to be extended

EXTSIZE= specifies the number of blocks to extend the file. If specified and that value cannot be obtained or if not specified, the default is the MAXINC size specified in the resource descriptor. If that value cannot be obtained, the MININC size specified in the resource descriptor is used.

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

Usage:

```
EXT TC?????? EXTS=2
```

All files created as manually extendible on the current working volume and directory beginning with TC containing file names with only eight characters are extended two blocks.

```
EXT @TEST(NEWPROG)*
```

Extends all files created as manually extendible on volume TEST in directory NEWPROG by the default increment size.

3.27 HELP Directive

The HELP directive obtains a short description of each directive used by VOLMGR.

Syntax:

HELP [directive]

directive specifies the directive to be described. If not specified, all directives and descriptions are displayed.

3.28 LOG FILE Directive

The LOG FILE directive generates a formatted listing of the specified disc files. Wild card characters can be used with this directive.

Syntax:

```
LOG FILE [LISTING=pathname] [PATH=pathname] [BRIEF=bool]
[ ( CREATED=  { SINCE  mo/dd/yy,hh:mm:ss  } )
[ (           { BEFORE dd-mon-yy,hh:mm:ss } )
[ ( CHANGED=  { SINCE  mo/dd/yy,hh:mm:ss  } )
[ (           { BEFORE dd-mon-yy,hh:mm:ss } )
[ ( SAVED=    { SINCE  mo/dd/yy,hh:mm:ss  } )
[ (           { BEFORE dd-mon-yy,hh:mm:ss } )
[ ( RESTORED= { SINCE  mo/dd/yy,hh:mm:ss  } )
[ (           { BEFORE dd-mon-yy,hh:mm:ss } )
```

PATH= specifies the pathname of the disc files to be logged

LISTING= specifies the pathname of a file to contain the listing. If not specified, the audit trail is used. This option can only be specified globally.

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

CREATED= if S is specified, all files created since the specified date and time are logged. If B is specified, all files created before the specified date are logged. If not specified, creation date and time are ignored.

CHANGED= if S is specified, all files changed since the specified date and time are logged. If B is specified, all files changed before the specified date are logged. If not specified, changed date and time are ignored.

SAVED= if S is specified, all files saved since the specified date and time are logged. If B is specified, all files saved before the specified date are logged. If not specified, saved date and time are ignored.

RESTORED= if S is specified, all files restored since the specified date and time are logged. If B is specified, all files restored before the specified date are logged. If not specified, restored date and time are ignored.

If wild card characters are used as part or in place of a file name, the following restrictions apply to the LOG FILE directive:

- If the owner name has read access to all directories, the directory portion of the pathname can be explicitly specified or can contain wild card characters.
- If the owner name does not have read access to all directories and wild card characters are used as part or in place of a directory name, the only directories searched are those the user has read access for.

If wild card characters are not used as part of a file name, the file name is explicitly specified; the owner only needs to have traverse access to the directory where the file is located.

Usage:

```
LOG F @ATLAS02(NEWDIR)????
```

Logs all files on volume ATLAS02 in directory NEWDIR containing file names with four characters and displays only their file names and sizes.

```
LOG F LIST=PROG PATH=@*(*)* BRIE=N CHAN=S 10:30
```

Logs all files on all public volumes that have been changed since 10:30 a.m. today and displays their status. The output of the log operation is in file PROG.

```
LOG F GLOSS
```

Logs file GLOSS on the current working volume and directory and displays only its file name and size.

3.29 LOG IMAGE Directive

The LOG IMAGE directive generates a listing of saved files in the current save tape image. The save tape must be positioned to the required save image before the directive is executed. Wild card characters cannot be used with this directive.

Syntax:

LOG IMAGE [LISTING=pathname]

LISTING= specifies the pathname of a file to contain the listing. If not specified, the audit trail is used.

Usage:

LOG I

Logs all saved files in the current save image.

LOG I LIST=LIST

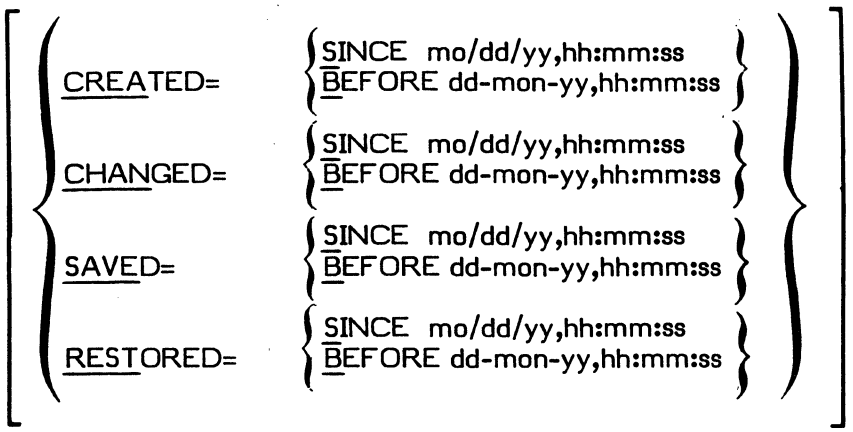
Logs all saved files in the current save image in file LIST.

3.30 LOG RESOURCE Directive

The LOG RESOURCE directive generates a formatted listing of a resource (directory, file, memory partition). Wild card characters cannot be used with this directive.

Syntax:

LOG [RESOURCE] [LISTING=pathname] [[PATH=] pathname] [BRIEF=bool] [ROOT=bool]



LISTING= specifies the pathname of a file to contain the listing. If not specified, the audit trail is used. This option can only be specified globally.

PATH= specifies the pathname of the resource to be logged. If not specified, logs files from the current working directory.

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

ROOT= specifies if the root directory is needed. When the resource specified is a root directory and the ROOT=T (or =Y) is specified, the volume root directory information is logged. If not specified, the default is ROOT=F (or =N).

CREATED= if S is specified, all files created since the specified date and time are logged. If B is specified, all files created before the specified date are logged. If not specified, creation date and time are ignored.

CHANGED= if S is specified, all files changed since the specified date and time are logged. If B is specified, all files changed before the specified date are logged. If not specified, changed date and time are ignored.

SAVED= if S is specified, all files saved since the specified date and time are logged. If B is specified, all files saved before the specified date are logged. If not specified, saved date and time are ignored.

RESTORED= if S is specified, all files restored since the specified date and time are logged. If B is specified, all files restored before the specified date are logged. If not specified, restored date and time are ignored.

Notes:

Because the keyword RESOURCE is optional for this directive, an ambiguity could arise when trying to log a resource named FILE, IMAGE, or RESOURCE since these are VOLMGR keywords. For example:

(1) LOG RESOURCE PATH=@VOL1(DIR1)FILE
LOG RESOURCE PATH=@VOL1(DIR1)IMAGE
LOG RESOURCE PATH=@VOL1(DIR1)RESOURCE

(2) LOG FILE
LOG IMAGE
LOG RESOURCE

In (1), the directive is explicitly stated and the specified resource would be logged. In (2), the current working directory and volume are implied and ambiguity arises in the meaning of the strings FILE, IMAGE, and RESOURCE. Are they directive adverbs or file names? Since the adverb precedes the file name on the directive line, they will be interpreted as the adverb and the file name will not be found, resulting in an error condition. To log a resource named FILE, IMAGE, or RESOURCE, (1) above would have to be specified or:

LOG PATH=FILE
LOG PATH=IMAGE
LOG PATH=RESOURCE

(or)

LOG R FILE
LOG R IMAGE
LOG R RESOURCE

Usage:

LOG

Logs all files on the current working volume and directory and displays only their file names and sizes.

LOG LIST=LIST @SYSTEM(TEST) CREA=S 14:00

Logs all files on volume SYSTEM in directory TEST that were created since 2:00 p.m. today and displays only their file names and sizes. Output from the log operation is in file LIST.

LOG R LIST=@ANYVOL(LOGOUT)OUTPUT SAVE=B 29-APR-81

Logs all files on the current working volume and directory that were saved before midnight on April 29, 1981, and displays only their file names and sizes. Output from the log operation is on volume ANYVOL, directory LOGOUT, file OUTPUT.

LOG

Logs all directory entries in the root directory of the current working volume.

LOG @TEST(NEWCODE)

Logs all files on volume TEST in directory NEWCODE.

LOG^ROOT=T BRIE=N

Logs the root directory information on the current working volume.

3.31 LOG SAVEFILE Directive

The LOG SAVEFILE directive reads a Resource Descriptor Tape Record (RDTR) from a save tape. The file pathname and other pertinent information is displayed. The save tape must be positioned to the file before the directive is executed. Wild card characters cannot be used with this directive.

Syntax:

```
LOG SAVEFILE [LISTING=pathname] [BRIEF=bool]
```

LISTING= specifies the pathname of a file to contain the listing. If not specified, the audit trail is used.

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

Usage:

```
LOG S
```

Logs one file on the save tape and displays only its file name and size.

```
LOG S LIST=STAT BRIE=N
```

Logs one file on the save tape on the current working volume and directory and displays its full status. Output from the log operation is in file STAT.

3.32 RENAME Directive

The RENAME directive changes the name of an existing file. Wild card characters can be used with this directive.

Syntax:

```
RENAME [FROM=]pathname [TO=]pathname [BRIEF=bool]
```

FROM= specifies the pathname of the file to be renamed

TO= specifies the pathname of the new file name

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

Notes:

If the file name specified in the TO parameter already exists, the rename does not take place and the following message is displayed:

```
UNABLE TO RENAME
```

The volume names in the FROM and TO parameters (if specified) must be the same.

Usage:

```
RENA TEMP PERM BRIE=N
```

Renames file TEMP on the current working volume and directory to PERM and displays its full status.

```
RENA FROM=^(ATLAS04)TEMP TO=^(ATLAS04)PERM
```

Renames file TEMP on the current working volume in directory ATLAS04 to PERM and displays only its file name and size.

```
RENA AA.* ZZ.*
```

Renames all files on the current working volume and directory that begin with AA., changing the first three characters of each file name to ZZ.

```
RENA BB* *XX
```

Renames all files on the current working volume and directory that begin with BB, appending the characters XX to each file name.

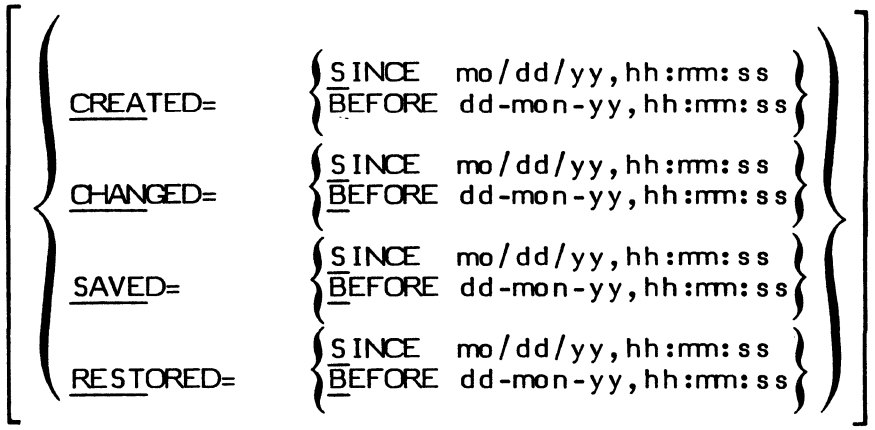
3.33 RESTORE DIRECTORY Directive

The RESTORE DIRECTORY directive restores specified files from a save tape to disc. Wild card characters can be used with this directive.

Syntax:

RESTORE [DIRECTORY] [VOLUME=volname] [[FROM=] pathname

[[TO=] pathname]] [SEGNUM = n]



[CONFIRM=bool] [NEWEST=bool] [BRIEF=bool]

VOLUME= specifies the volume where the file will be restored if it is a different volume from the one where the file was saved. If not specified, the file will be restored to the same volume where it was saved, if the volume exists. If the volume does not exist, the restore will not take place. This option must not be specified if the TO parameter is specified.

FROM= specifies the pathname the file to be restored was saved under.

If neither FROM or TO pathnames are specified, all files in the save image are restored to the volume and directory from which they were saved.

If the FROM pathname is specified and the TO pathname is not specified, files are restored to the FROM pathname.

TO= specifies the destination pathname where the file is restored. TO cannot be specified without FROM. See FROM= for more usage information.

SEGNUM= specifies the maximum acceptable segment numbers if contiguous restoration cannot be achieved. If this parameter is specified, the file's extension attributes are ignored temporarily.

CREATED= if S is specified, all files created since the specified date and time are restored. If B is specified, all files created before the specified date are restored. If not specified, creation date and time are ignored.

CHANGED= if S is specified, all files changed since the specified date and time are restored. If B is specified, all files changed before the specified date are restored. If not specified, changed date and time are ignored.

SAVED= if S is specified, all files saved since the specified date and time are restored. If B is specified, all files saved before the specified date are restored. If not specified, saved date and time are ignored.

RESTORED= if S is specified, all files restored since the specified date and time are restored. If B is specified, all files restored before the specified date are restored. If not specified, restored date and time are ignored.

CONFIRM= specifies if confirmation is to be requested before restoring the file. Operator response is required to the confirmation prompt. If not specified, confirmation is not given. This option is valid in the interactive mode only.

NEWEST= if the file name specified in the TO parameter already exists, indicates whether the newest version of it should be kept. If not specified, the file on tape is restored if it meets the specifications described below under Notes.

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

Notes:

Three aspects of RESTORE must be considered separately:

1. The eligibility of a file to be restored due to date constraints supplied as options on the directive line

2. The owner name, project group, and access attributes of the file specified in the TO parameter

3. The values of the created, changed, saved, and restored date in the resource descriptor of the restored file

The rules for these three aspects are described below.

1. Time constraints as options are handled as follows:

   a. CREATED, CHANGED, SAVED, RESTORED

      The file is eligible if the relevant date in the resource descriptor of the file on tape satisfies the comparison condition (SINCE/BEFORE) with the date/time specified in the directive.

b. CONFIRM

Confirmation is performed before any restoring is performed unless CONF=N is specified.

c. NEWEST

If the file does not already exist on disc, it is automatically eligible to be restored.

If the file does already exist on disc, the tape version is eligible if:

- The create dates match but the changed date on tape is newer than the changed date on disc
- The create date on tape is newer than the create date on disc

2. Attributes of the restored file are handled as follows:

a. The disc destination is implicit (TO parameter not specified)

This is a true restore of data over the original data, if the original data still exists, or into a file of the same name as the original file, if it no longer exists.

If the original directory name no longer exists, VOLMGR:

- Creates a directory with the same attributes as the directory from which the file was originally saved. This information is on the save tape.
- Restores the file with the same attributes as the original file, including owner and project group names

If the original directory name still exists, VOLMGR:

- Restores the file with the same attributes as the original file, including owner and project group names

When the user running the job is not either the System Administrator or the owner of the file(s) being restored, an error is generated.

b. The disc destination is explicit (TO parameter specified)

This is a copy of the tape file rather than a true restore. If the file already exists on disc, it is overwritten. If the file does not already exist on disc, it is created.

If the new directory name does not exist on disc:

- An error is generated. The directory must already exist since VOLMGR cannot make any reasonable defaults.

If the new directory name does exist on disc:

- When the user running the job is not the System Administrator, the file is copied with the same attributes as the original file except for owner and project group, which become the owner and project group of the user running the VOLMGR task.

- When the user running the job is the System Administrator, the file is copied with the same attributes as the original file, including owner and project group names.

3. Date field values in the resource descriptor of the new file are handled as follows:

   If the file from tape has been restored, the date fields in the new file's resource descriptor will be set as follows:

   a. CREATED date

      Set to the created date from the tape

   b. CHANGED date

      Set to the changed date from the tape

   c. SAVED date

      Set to the saved date on the tape if the System Administrator is running the restore. The saved date remains unchanged if it is not the System Administrator running the restore and the file already exists on disc. Otherwise, the saved date is set to zero. This allows a 'system manager' to control an incremental save/restore mechanism.

   d. RESTORED date

      Set to the current date and time.

Usage:

```
RES FROM=@ATLAS04(TEMP)* CONF=T
```

Restores all files in the save image that were saved from volume ATLAS04 and directory TEMP to volume ATLAS04 and directory TEMP. Requests confirmation for each file before it is restored.

```
RES VOLU=TESTCASE CONF=Y CREA=S 29-AUG-81,05:00
```

Restores all files in the save image created since 5:00 a.m. on August 29, 1981 to volume TESTCASE in the directory where they were saved. Requests confirmation before each restore operation and displays only the file name and size of each file.

```
RES CHAN=S 04/01/81 NEWE=Y
```

Restores all files in the save image that were changed since midnight on April 1, 1981 to the volume and directory where they were saved. If any file on the tape still exists on disc, the copy of the file on tape replaces the copy of the file on disc according to eligibility rules stated in 1c above.

RESTORE

Restores all files in the save image to the same volume, directory, and file name where they were saved.

```
RES FROM=@*(*)* TO=@SYSTEM(SYSTEM)*
```

Restores all files in the save image to volume SYSTEM in directory SYSTEM. All files in the save image should have unique file names. Otherwise, files saved from different volumes and directories with the same file names map to one file that is listed on volume SYSTEM in directory SYSTEM.

```
RES @V1(SYSTEM)F66* @VF77(F77)F77*
```

Restores all files in the save image beginning with the characters F66 that were saved from volume V1 in directory SYSTEM to volume VF77 in directory F77. The first three characters of each file name are changed to F77.

```
RES @V1(SYSTEM)* @VF77(F77)*F77
```

Restores all files in the save image that were saved from volume V1 and directory SYSTEM to volume VF77 and directory F77. The characters F77 are appended to each file name.

3.34 RESTORE POSITION Directive

The RESTORE POSITION directive restores one file from a magnetic tape to disc. The tape must be positioned to the desired file before the directive is executed. Wild card characters cannot be used with this directive.

Syntax:

RESTORE POSITION [[TO=] pathname]

TO= specifies the pathname of a file where the data is to be restored

Usage:

RES P

Restores the file on tape to the same volume and directory where it was saved, if they exist. If they do not exist, the restore does not take place.

RES P @TEST(SOURCE)'SS*WRKFL'

Restores the file on tape to volume TEST, directory SOURCE, file SS*WRKFL.

3.35 REWIND Directive

The REWIND directive positions a magnetic tape to its loading point (BOT).

Syntax:

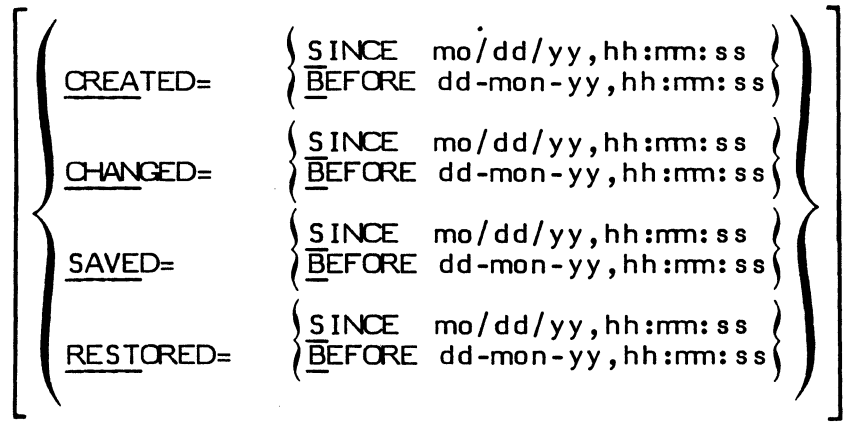
REWIND

3.36 SAVE Directive

The SAVE directive saves individual files as a single save image. Wild card characters can be used with this directive.

Syntax:

SAVE [PATH=]pathname [CONFIRM=bool] [SAVN=bool] [BRIEF=bool]



PATH= specifies the pathname of the files to be saved

CONFIRM= specifies if confirmation is requested before each file is saved. Operator response is required to the confirmation prompt. If not specified, confirmation is not given. This option is valid in the interactive mode only.

SAVN= specifies if the VOLMGR no-save attribute should be overridden. If not specified, files created with the no-save attribute enabled are not saved.

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

CREATED= if S is specified, all files created since the specified date and time are saved. If B is specified, all files created before the specified date are saved. If not specified, creation date and time are ignored.

CHANGED= if S is specified, all files changed since the specified date and time are saved. If B is specified, all files changed before the specified date are saved. If not specified, changed date and time are ignored.

SAVED= if S is specified, all files saved since the specified date and time are saved. If B is specified, all files saved before the specified date are saved. If not specified, saved date and time are ignored.

RESTORED= if S is specified, all files restored since the specified date and time are saved. If B is specified, all files restored before the specified date are saved. If not specified, restored date and time are ignored.

Notes:

Four aspects of SAVE must be considered separately:

1. The eligibility of a file to be saved due to date constraints supplied as options on the directive line

2. Who can save files

3. The values of the created, changed, saved, and restored date in the resource descriptor of the saved file

4. Incremental saves

The rules for these four aspects are described below.

1. Time constraints as options are handled as follows:

   a. CREATED, CHANGED, SAVED, RESTORED

      The file is eligible if the relevant date in the resource descriptor of the file on disc satisfies the comparison condition (SINCE/BEFORE) with the date/time specified in the directive.

   b. CONFIRM

      Confirmation is performed before any saving is performed if CONF=Y is specified.

   c. SAVE

      If SAVN=Y is specified, any files created with the NOSAVE option can be saved.

2. Read access is required to save files.

3. Date field values in the resource descriptor of the saved file:

   a. CREATED, CHANGED, and RESTORED dates

      These dates remain unchanged on disc and tape.

   b. SAVED date

      On tape, the date always reflects the date of the save.

      On disc, the date will only be updated if the System Administrator is executing the save. This allows a 'system manager' to control an incremental save/restore mechanism.

4. Incremental saves are performed by using the SAVE INCREMENTAL directive. The dates on the tape are modified as specified in 3 above.

A file's data space is only saved to EOF as recorded in the resource descriptor. Any I/O error which occurs during the save process invalidates that save image. If an I/O error occurs during the save process, the VOLMGR immediately terminates the save process and backspaces the tape to the end of the previous save image or the beginning-of-tape, whichever is applicable.

Usage:

```
SAVE *
```

Saves all files in the current working volume and directory.

```
SAVE @*(*)*
```

Saves all files in all volumes and all directories.

```
SAV @SYSTEM(*)* BRIE=N
```

Saves all files in all directories on volume SYSTEM and displays full status for each file.

```
SAV PATH=@TEST(TEMP)XX.???? REST=B 15:30
```

Saves all files on volume TEST in directory TEMP that begin with XX., contain file names with only seven characters, and were restored before 3:30 p.m. today. Only the file name and size of each file is displayed.

Note: If the system administrator is the logged on user and an I/O error occurs during the save process, all files processed before the tape I/O error occurred were saved on the tape. All files after the I/O error can be processed with another SAVE directive on another tape.

3.37 SAVE INCREMENTAL Directive

The SAVE INCREMENTAL directive saves files created or changed since the last SAVE directive was performed. The saved date and time in the file's resource descriptor is compared with the changed or created date and time in the file's resource descriptor. If the changed or created date and time is later than the saved date and time, the file is saved.

On disc, the saved date only updates if the System Administrator is executing the save. On tape, the saved date always reflects the date of the save. The created, changed, and restored dates remain unchanged on disc and tape.

Wild card characters can be used with this directive.

Syntax:

SAVE INCREMENTAL [PATH=] pathname

PATH= specifies the pathname of the file to be saved

Usage:

SAVE I @NEWVOL(*)*

Saves all files in all directories on volume NEWVOL that have been created or changed since the last SAVE was performed.

SAVE I SOURCE

Saves file SOURCE on the current working volume and directory if it has been created or changed since the last SAVE was performed.

Note: If the system administrator is the logged on user and an I/O error occurs during the SAVE INCREMENTAL process, all files processed before the tape I/O error occurred were saved on the tape. All files after the I/O error can be processed with another SAV I directive on another tape.

3.38 SDT Directive

The SDT directive is most commonly used for creating a System Distribution Tape (SDT). A system distribution tape contains a bootstrap loader, a system image in pseudo load module form, and other essential load modules that are activated directly from the tape. Load modules written to tape by the SDT directive cannot be logged or restored like save files because there is no directory information retained about them. To position the tape past the output generated by the SDT directive, use the SKIP IMAGE directive.

The SDT directive writes a bootstrap loader to the tape, followed by user-specified load module files. When IPL is from the SDT, the bootstrap loader is read into main memory and given control by the firmware. The bootstrap loader then loads and relocates load module files from the SDT. Load modules are loaded starting at physical address X'800' or optionally at the physical address contained in register three. Loading continues until a load module contains a transfer address. At this point, the bootstrap loader passes control to the loaded module through a branch and link. The module has the option of returning to the bootstrap loader through a TRSW to optionally load more modules. Since boot programs cannot process a multivolume header, a user SDT should not be created on a multivolume tape.

Wild card characters cannot be used with this directive.

Syntax:

```
SDT [PATH=] pathname [ [WEOF=bool] [,[ PATH=] pathname
[WEOF=bool] ] ]...
```

PATH= specifies the pathname of the file containing the system image. More than one pathname can be specified (see Notes below).

WEOF= specifies an end-of-file is to be written after the associated file. This option can only be specified locally. If not specified, an end-of-file is not written.

Notes:

For a user magnetic tape or floppy disc SDT, one PATH parameter specifying the system image is required. When additional file pathnames are not specified, the default set of load modules and format is as follows:

Bootstrap loader (standard)
System image
J.VFMT
End-of-file mark
J.MOUNT
J.SWAPR
VOLMGR
End-of-file mark
End-of-file mark

For a master floppy disc SDT, PATH=@SYSTEM(SYSTEM)FLOP.SYS is required. The format shown on the previous page is the same format generated for the master SDT. However, the name FLOP.SYS causes master SDT processing upon booting the system.

When additional file pathnames are specified after the system image pathname to create a custom-built SDT, the additional files which are specified must be load modules, and the last file pathname must specify option WEOF=Y to enable the SDT directive to properly format the save image with two end-of-file marks. See Usage.

The logical file code TAP is assumed to be assigned to the desired device before invoking VOLMGR. The device assignments accepted by the SDT directive must be for magnetic tape or floppy disc.

Usage:

```
SDT PATH=@SYSTEM(SYSTEM)MPXSYS99 WEOF=N-
      ,PATH=@SYSTEM(SYSTEM)J.VFMT WEOF=Y-
       ,PATH=@SYSTEM(SYSTEM)J.MOUNT WEOF=N-
      ,PATH=@SYSTEM(SYSTEM)J.SWAPR WEOF=N-
      ,PATH=@SYSTEM(SYSTEM)VOLMGR WEOF=Y
```

Generates a user SDT that matches the default.

Note: For this format, a Y must be entered in the Volume Manager's WEOF parameter in order for the SDT to be properly generated.

```
SDT PATH=@SYSTEM(SYSTEM)IMAGEFILE
```

Generates a user SDT with the system image located on volume SYSTEM, directory SYSTEM, file IMAGEFILE.

```
SDT PATH=@SYSTEM(SYSTEM)IMAGEFILE WEOF=N-
,PATH=@SYSTEM(SYSTEM)LOADMOD1 WEOF=N-
,PATH=@SYSTEM(SYSTEM)LOADMOD2 WEOF=Y
```

Generates a custom-built user SDT with the following format:

```
Bootstrap loader (standard)
IMAGEFILE
LOADMOD1
LOADMOD2
End-of-file mark
End-of-file mark
```

```
SDT PATH=@SYSTEM(SYSTEM)FLOP.SYS
```

Performs master SDT processing when the system is booted.

3.39 SDT MASTER Directive

The SDT MASTER directive creates a master System Distribution Tape (SDT). Load modules written to tape by the SDT MASTER directive cannot be logged or restored like save files because there is no directory information retained about them. To position the tape past the output generated by the SDT MASTER directive, use the SKIP IMAGE directive.

Syntax:

SDT MASTER

The tape format is as follows:

Bootstrap Loader
MSTR.27
End-of-file mark
MSTR.75
End-of-file mark
MSTR.87
End-of-file mark
J.VFMT
End-of-file mark
J.MOUNT
J.SWAPR
VOLMGR
End-of-file mark
End-of-file mark

The logical file code TAP is assumed to be assigned to the desired device before invoking VOLMGR. The device assignments accepted by the SDT MASTER directive must be for magnetic tape or floppy disc.

The system image files MSTR.27, MSTR.75 (a null file), and MSTR.87 must exist and be located on the system volume in the system directory before issuing this directive. These three files contain specific processing required to perform the SDT boot. These files should not be modified.

The SDT MASTER directive should only be used with magnetic tape SDTs. To make a master floppy disc SDT, the SDT directive with PATH=@SYSTEM(SYSTEM)FLOP.SYS must be used. To generate the FLOP.SYS image, use the directive file DIR.27FL provided with the release copy of MPX-32. The special file name FLOP.SYS causes master SDT processing upon booting the system.

For more information on a master SDT, see the MPX-32 Reference Manual, Volume III, Chapter 4.

3.40 SET Directive

The SET directive establishes global default option values for VOLMGR directives where the specific options are applicable.

Syntax:

```
SET [PROJECTGROUP=name] [START=n] [MAXINC=n] [MININC=n]
[MAXSIZE=n] [OWNER=name] [LISTING=pathname] [SIZE=n]
[REPLACE=bool]  [SHARED=bool] [NOSAVE=bool]
[ZERO=bool] [EOFM=bool]
[FAST=bool] [[MEMCLASS= {E|H|S}]  [CONFIRM=bool] [NEWEST=bool]
[BRIEF=bool] [AUTOEXT=bool] [MANEXT=bool] [CONTIGUOUS=bool]
[ { CREATED=  {SINCE  mo/dd/yy,hh:mm:ss  | BEFORE dd-mon-yy,hh:mm:ss}
  | CHANGED=  {SINCE  mo/dd/yy,hh:mm:ss  | BEFORE dd-mon-yy,hh:mm:ss}
  | SAVED=    {SINCE  mo/dd/yy,hh:mm:ss  | BEFORE dd-mon-yy,hh:mm:ss}
  | RESTORED= {SINCE  mo/dd/yy,hh:mm:ss  | BEFORE dd-mon-yy,hh:mm:ss} } ]
```

PROJECTGROUP= specifies a project group name. If not specified, defaults to the project group name associated with the VOLMGR task.

START= specifies a starting block number. If the specified block number cannot be used, a denial is given. If not specified, any available space on the volume is used.

MAXINC= specifies a size for subsequent automatic increments. If not specified, the default is 64 blocks.

MININC= specifies a minimum acceptable automatic increment size if MAXINC value cannot be obtained. If not specified, the default is 32 blocks.

MAXSIZE= specifies a maximum size for an extendible file. If not specified, the size is limited only by available disc space.

OWNER=	specifies an owner name. If not specified, defaults to the owner name associated with the VOLMGR task.
LISTING=	specifies the pathname of a file to contain the listings. If not specified, the audit trail is used.
SIZE=	specifies a block size for initial allocation. If not specified, the default is 16 blocks.
REPLACE=	if a resource currently exists, indicates whether it should be replaced (deleted). If not specified, the existing resource remains unchanged.
SHARED=	specifies if a resource is to be used in the shared mode. If not specified, the default is shared.
NOSAVE=	specifies if the VOLMGR no-save attribute applies to a resource. If not specified, the resource will be savable in response to the VOLMGR SAVE directive.
ZERO=	specifies if a resource is to be zeroed on creation and extension. If not specified, the file is not zeroed.
EOFM=	specifies the status of EOF management. If not specified, the system default is used.
FAST=	specifies if a file's resource identifier is not to be changed when the file is copied or restored using the VOLMGR. If not specified, the file will be placed in any available resource identifier.
MEMCLASS=	specifies the class of memory. If not specified, the default is class S.
CONFIRM=	specifies if confirmation is to be given upon processing a directive. Operator response is required to the confirmation prompt. If not specified, confirmation is not given. This option is valid in the interactive mode only.
NEWEST=	if a resource already exists, indicates whether the newest version of it should be kept.
BRIEF=	specifies if complete file status is desired. If not specified, only the file name and size are displayed.
AUTOEXT=	specifies if the file can be automatically extended. If not specified, the file is automatically extendible.
MANEXT=	specifies the file can be manually extended. If not specified, the file is manually extendible.
CONTIGUOUS=	specifies if extensions to the file are to be contiguous when possible. Initial allocation of a file will always be contiguous. If not specified, extensions are not contiguous.

CREATED= if S is specified, resources created since the specified date and time are to be processed. If B is specified, resources created before the specified date are deleted/saved. If not specified, creation date and time are ignored.

CHANGED= if S is specified, resources changed since the specified date and time are to be processed. If B is specified, resources created before the specified date are deleted/saved. If not specified, changed date and time are ignored.

SAVED= if S is specified, resources saved since the specified date and time are to be processed. If B is specified, resources created before the specified date are deleted/saved. If not specified, saved date and time are ignored.

RESTORED= if S is specified, resources restored since the specified date and time are to be processed. If B is specified, resources restored before the specified date are deleted/saved. If not specified, restored date and time are ignored.

Usage:

```
SET PROJ=REL2.0 STAR=25 SIZE=5 FAST=Y CONF=N-
BRIE=Y AUTO=Y MANE=Y CONT=Y
```

These default option values apply unless overridden on the directive line.

3.41 SKIP END Directive

The SKIP END directive positions a save tape at its logical end-of-tape.

Syntax:

SKIP END

3.42 SKIP FILE Directive

The SKIP FILE directive moves a save tape forward a specified number of EOF marks. If the specified number is more than the number of files on the save tape, the SKIP FILE directive positions the save tape at its logical end-of-tape.

Syntax:

SKIP FILE [[FILES=]n]

FILES= specifies the number of EOF marks to skip. If not specified, the save tape is positioned forward one EOF.

Usage:

SKI F

Moves the magnetic tape forward one EOF mark.

SKI F FILE=2

SKI F 2

Both of these directives move the magnetic tape forward two EOF marks.

3.43 SKIP IMAGE Directive

The SKIP IMAGE directive moves a save tape forward a specified number of save images.

Syntax:

SKIP [IMAGE] [[IMAGES=] n]

IMAGES= specifies the number of save images to skip. If not specified, the save tape is positioned forward one save image.

Usage:

SKI

Moves the magnetic tape forward one save image.

SKI IMAG=2

SKI 2

Both of these directives move the magnetic tape forward two save images.

3.44 TRUNCATE Directive

The TRUNCATE directive decreases the space allocated for a file that has been extended after the file is created. If any part of a segment is used by the file, that entire segment is reserved for the file. Wild card characters can be used with this directive.

Syntax:

TRUNCATE [PATH=]pathname [BRIEF=bool]

PATH= specifies the pathname of a file whose space is to be decreased

BRIEF= specifies if complete file status is desired. If not specified, only the file name and size are displayed.

Usage:

TRU @NEWVOL(NEWDIR)INDEX

Truncates file INDEX on volume NEWVOL in directory NEWDIR.

TRU *

Truncates all files on the current working volume and directory.

3.45 Errors and Aborts

Error messages are generated whenever a directive cannot be processed. A prompt displays for the next directive to be issued rather than aborting.

Aborts codes and their messages are described in Appendix C.

CHAPTER 4

COMPRESS

4.1 General Description

COMPRESS collects object modules into a single file. The resulting file can be cataloged, then executed. The use of COMPRESS to build an object file for system generation is described in Volume III.

4.2 Accessing COMPRESS

COMPRESS can be accessed in the interactive or batch modes.

Syntax:

```
COMPRESS
```

When entered, COMPRESS writes the following to the logical file code LO:

```
==================================
! MPX PATHNAME COMPRESS UTILITY !
==================================
```

COMPRESS then reads the input file of object module pathnames. The object modules are allocated and copied into the output file. An audit trail of COMPRESS activity is written to the output file. When COMPRESS is finished concatenating the input object modules in the interactive mode, the TSM prompt is displayed.

4.3 Logical File Code Assignments

There are three logical file codes (LFCs) associated with COMPRESS: Input File (IN), Output File (OT), and System Listed Output (LO).

4.3.1 Input File (IN)

The input file is a file of ASCII pathnames, one per record/line. Each line in the file can be a maximum of 72 characters, and contains one object module pathname and its optional description. The pathname and optional comments must be separated by one or more blanks.

The pathnames can be collected in an EDIT file, then stored. The input file is assigned to logical file code IN.

IN Default and Optional Assignments

The default assignment for IN is to the system file JH.32:

```
$ASSIGN IN TO JH.32 BLOC=Y
```

The optional assignment for IN is to a pathname:

```
$ASSIGN IN TO pathname
```

pathname is the pathname of a file containing pathnames of object modules to be concatenated

4.3.2 Output File (OT)

The output file contains the concatenated object modules output by COMPRESS. The output file is assigned to logical file code OT.

OT Default and Optional Assignments

The default assignment for OT is to the system file OH.32:

```
$ASSIGN OT TO OH.32 BLOC=Y
```

The optional assignment for OT is to a pathname:

```
$ASSIGN OT TO pathname
```

pathname is the pathname of a file to contain the concatenated object modules

4.3.3 Listed Output File (LO)

The listed output file contains a COMPRESS audit trail. The audit trail lists:

- the files copied
- the number of records per file
- allocation or read errors

The listed output file is assigned to logical file code LO.

LO Default and Optional Assignments

The default assignment for LO is to logical file code UT:

```
$ASSIGN LO to LFC=UT
```

There are two optional assignments for LO:

```
$ASSIGN LO TO {pathname}
              {DEV=devmnc}
```

pathname is the pathname of a file to contain the COMPRESS audit trail

devmnc is the device mnemonic of a device to contain the COMPRESS audit trail

4.3.4 LFC Summary

The following is a table of LFCs used by COMPRESS and their default and optional assignments.

Table 4-1
COMPRESS LFC Summary

LFC	Default Assignment	Optional Assignment
IN	JH.32	pathname
OT	OH.32	pathname
LO	LFC=UT	pathname DEV=devmnc

4.4 Error Messages

When one of the following error messages is displayed to the listed output file, COMPRESS exits and control is returned to TSM.

COMPRESS - NONOBJECT RECORD ON PATH pathname

The pathname is not for an object module. Delete the pathname from the file assigned to logical file code IN.

COMPRESS - UNEXPECTED END OF FILE ON PATH pathname

Improper format for an object module. A DF record was not found before the end of the object module.

COMPRESS - READ ERROR STATUS = status ON PATH pathname

The returned status is defined in the FORTRAN 77+ Reference Manual.

4.5 Example

In the following example, COMPRESS uses file OBJPATH for input. COMPRESS output is written to COMPOUT:

```
$AS IN TO OBJPATH
$AS OT TO COMPOUT
$COMPRESS
```

The following listed output is displayed on logical file code LO:

```
================================
! MPX PATHNAME COMPRESS UTILITY !
================================

COPIED - 59 RECORDS FROM PATH @SYSTEM ^ (OBJECT)OJ.F1
COPIED -  7 RECORDS FROM PATH @SYSTEM ^ (OBJECT)OJ.F2
COPIED - 22 RECORDS FROM PATH @SYSTEM ^ (OBJECT)OJ.F3
```

CHAPTER 5

RAPID FILE ALLOCATION UTILITY (J.MDTI)

5.1 Overview

Rapid File Allocation is a mechanism which allows copies of Resource Descriptors (RDs) for selected permanent files to reside in memory, in the Memory Resident Descriptor Table (MDT). This eliminates the disc access normally necessary to allocate these files; faster allocation of the files results. If an MDT exists, it is searched first for the RD for the file being allocated. If that file's RD is contained in the MDT, the file is allocated immediately. Because the MDT is searched for all files, this can impose a slight overhead for all other disc allocations. (See Section 5.6 for information on avoiding this overhead.)

The Rapid File Allocation mechanism is implemented through the following elements:

MDT	Memory Resident Descriptor Table
H.MDT	Module included in resident operating system
J.MDTI	MDT initialization task
M.MDTF	Default J.MDTI input file containing pathnames and Resource IDs (RIDs) for initialization
User files	Alternate input files to enter/append file names in MDT.

The SYSGEN MDT directive sets the MDT size, optionally specifies the MDT starting block in memory, and installs H.MDT as part of the resident operating system. The MDT initialization task, J.MDTI, uses an input file to enter pathnames and RIDs in the MDT. J.MDTI is automatically activated at system initialization by SYSINIT. After system initialization, the MDT can be updated or reinitialized by activating J.MDTI from TSM or a user task.

The MDT is not mapped into a task's logical address space or the resident operating system. The MDT is located in the highest available contiguous memory space, unless another area is specified by the SYSGEN MDT directive. Code for MDT processing services is located in the module H.MDT.

The MDT's size -- the number of 192 word RDs it is to contain -- must be specified in the SYSGEN MDT directive. To accommodate collision resolution, J.MDTI allocates 25% more space than requested. Space is allocated even if the input file, M.MDTF, is empty. After allocation, all of the MDT space -- including the extra 25% -- can be used; however, this is not recommended.

Files whose pathnames are to be added to the MDT must:

be permanent files
reside on **public**, single-ported volumes.

An attempt to enter files without these characteristics results in an abort when the input file is processed by J.MDTI.

The instructions for setting up input files are located in section 5.5.

Accessing J.MDTI from TSM or user tasks to append the MDT is described in the following sections.

5.2 Accessing J.MDTI

At system initialization, J.MDTI is automatically activated by SYSINIT. J.MDTI attempts to use the default file @SYSTEM(SYSTEM)M.MDTF as input for the MDT. If M.MDTF is present and contains valid pathnames, the MDT is initialized and the file RDs are entered. If M.MDTF is present but contains no options or pathnames, the MDT is initialized and J.MDTI returns control to the operating system. If M.MDTF is not present, J.MDTI aborts without initializing the MDT. After system initialization, J.MDTI can be accessed by TSM or a user task to append the MDT.

To enter or append file RDs to the MDT after system initialization, create an alternate input file following the rules in Section 5.5. J.MDTI can then be activated from TSM using the input file's name as a parameter on the command line.

J.MDTI can also be activated by a run request sent from a task. The input file's name is passed as a run parameter.

Note: If a parameter is not supplied in either case, J.MDTI uses @SYSTEM(SYSTEM)M.MDTF as the default input file.

The following activates J.MDTI from TSM and uses M.MDTFA1 as an alternate input file.

```
TSM>J.MDTI M.MDTFA1
```

See Section 5.5.1 for further demonstration.

Access to J.MDTI can be restricted on an owner name basis by setting the appropriate bit in the M.KEY file.

5.3 Logical File Code Assignments

There are five logical file codes associated with J.MDTI: Input File (MDT), Temporary File (TMP), Resource Identifiers (RID), Pathname Output (RPN), and Listed Output (SLO).

5.3.1 Input File (MDT)

The input file contains the pathnames and RIDs of the files whose RDs are to reside in the MDT. The input file is dynamically assigned to logical file code MDT. There are no optional assignments for MDT.

5.3.2 Temporary File (TMP)

If the BLD option is present in the input file, an intermediate storage area is assigned to logical file code TMP. This area is used to create a file that replaces the input file when processing completes. The new file contains the RIDs that were created by J.MDTI. The default assignment for TMP is to a temporary file. There are no optional assignments for TMP.

5.3.3 Resource Identifiers (RID)

The resource identifiers to be added to the MDT are assigned to logical file code RID, if the BLD option is present. There are no optional assignments for RID.

5.3.4 Pathname Output (RPN)

The audit trail contains the pathnames that were input to J.MDTI. The pathname information to be echoed is assigned to logical file code RPN.

RPN Default and Optional Assignments

The default assignment for RPN is to logical file code SLO:

```
$ASSIGN RPN TO SLO
```

There are three optional assignments for RPN:

```
$ASSIGN RPN TO { pathname  }
               { DEV=devmnc}
               { LFC=UT    }
```

5.3.5 Listed Output (SLO)

The listed output file contains an audit trail of the J.MDTI run. The audit trail lists the commands processed and any errors that occurred. The listed output file is assigned to logical file code SLO.

SLO Default and Optional Assignments

The default assignment for SLO is to logical file code SLO:

```
$ASSIGN SLO TO SLO
```

There are three optional assignments for SLO:

```
$ASSIGN SLO TO { pathname  }
               { DEV=devmnc}
               { LFC=UT    }
```

5.3.6 LFC Summary

The following is a table of LFCs used by J.MDTI and their default and optional assignments.

Table 5-1
J.MDTI LFC Summary

LFC	Default Assignment	Optional Assignment
MDT	M.MDTF	N/A
TMP	temporary file	N/A
RPN	SLO	pathname DEV=devmnc LFC=UT
RID	resource identifier	N/A
SLO	SLO	pathname DEV=devmnc LFC=UT

5.4 Exiting J.MDTI

When J.MDTI processing is complete, control automatically returns to TSM.

5.5 Input Files

Input files contain the pathnames and Resource IDs (RIDs) of the files whose RDs are to reside in the MDT. It is not necessary to include the RIDs, but initialization of the MDT is expedited if they are present.

Alternate input files may reside on any volume, but the default input file, M.MDTF, must be on the system volume in the system directory. Input files must be stored uncompressed and unnumbered to a file name other than M.MDTF.

Input files are composed of the following elements:

Processing Options
File Pathnames
Blank lines
Comments

The first line of the input file can be used to specify processing options. The option keywords shown below are separated by one or more spaces, and are order-independent.

LOF Disables the initialization/append log. If this option is present, no file pathnames, commands, or error messages are echoed to SLO. If not specified, logging occurs.

BLD RIDs are to be fetched by J.MDTI and added to the input file as well as the MDT. This causes the input file to be rewritten with the RID following each file's pathname. If an RID exists in the input file, it is compared to the disc RID for that file, and if they do not match, J.MDTI replaces the erroneous entry. If an invalid entry is found, that entry is commented out of the input file.

RID J.MDTI expects that RIDs are already specified for all file pathnames in the input file; an error is generated for any pathname that does not have a corresponding RID. This option is overridden by the BLD option, if present.

HLT Halt on error. If a nonfatal error occurs during initialization processing, processing terminates, and the MDT is not built. If RDs are being appended to the MDT when a nonfatal error occurs, any RDs that were appended are deleted, and the MDT is left as it was before processing began.

If the MDT is being reinitialized (option INI), all RDs are deleted and the MDT remains allocated.

INI Reinitializes the MDT. This option zeros all entries, then adds the files specified in the input file. This option is not required in M.MDTF at system initialization time.

MDT initialization is expedited by having the file RIDs present in the input file. If the BLD option is specified, it is replaced by an RID option when the input file is rewritten. Subsequent system startups are significantly faster.

If RIDs are not present, and the BLD option is not used, J.MDTI has additional overhead to contend with every time the system is rebooted because the input file is not rewritten to contain the RIDs.

All file pathnames in the input file must be fully qualified pathnames. Only one pathname per line is allowed. Wildcard characters in pathnames are not supported.

Blank lines, and comments preceded by an exclamation point, are supported.

See the next section for further demonstration.

5.5.1 Usage Examples

The following example shows the possible contents of M.MDTF.

```
RID LOF
!PRIMARY INPUT FILES
@SYSTEM(SYSTEM)M.TEST1          RID=DISC1,2345,102365,243,A
@DISC2(DIRE3)M.TEST2            RID=DISC2,4236,224389,221,A
```

After IPL, the RDs for M.TEST1 and M.TEST2 are in the MDT.

To add RDs to the MDT, use an alternate input file, like the one shown in this example.

```
HLT BLD RID
!BLD OPTION CANCELS THE USE OF THE RID OPTION FOR THIS RUN
!BLD OPTION IS REMOVED FROM THE INPUT FILE AFTER THE RUN
!INPUT FILE WILL CONTAIN THE RID
!FOR M.TEST3 AFTER THE RUN
@DISC1^(DIRE2)M.TEST3
@DISC2^(DIRE3)M.TEST4           RID=DISC2,5342,312723,423,A
```

When J.MDTI is called by TSM using the name of this alternate input file, M.TEST3 and M.TEST4 are added to the MDT.

NOTE: The RIDs in the examples are for example purposes only. To run these examples, replace the RIDs listed here with actual RIDs.

5.6 Rapid File Allocation Programming Considerations

The MDT entries are duplicates of the RDs which reside on disc. RDs are dynamic structures; modifications to the RDs to update the resource attributes are automatically mirrored to the MDT entries. The integrity of the MDT structure is dependent on the maintenance of resources through the H.VOMM module. Tasks that modify disc structures directly, not using the H.VOMM services, can destroy the MDT's integrity; direct changes made to resource descriptors are not reflected in the MDT.

When a file whose RD is contained in the MDT is renamed or deleted, its corresponding MDT entry is deleted. In the case of a renamed file, no new MDT entry is made. The new name can be re-entered by activating J.MDTI with an alternate input file.

Establishing an MDT imposes some overhead on all file location operations (i.e. finding, logging, allocating) because the MDT must be searched for an entry each time. This overhead can be avoided when locating resources which are known not to be in the MDT. The system services M.LOGR and M_LOGR do not search the MDT. The M.LOC service **does** search the MDT first, before accessing the disc.

5.7 Errors

J.MDTI error codes are prefixed by the characters RF. A summary is contained in Appendix C.

Check the audit trail for a list of errors generated during the run.

Console Messages

The following messages are written to the console when J.MDTI is initializing the MDT.

Initializing memory descriptor table.

Memory descriptor table initialization complete.

The following messages are written to the console when J.MDTI is appending the MDT.

Appending memory descriptor table.

Memory descriptor table append complete.

The following errors can occur during J.MDTI processing.

J.MDTI: Fatal error has been detected. Check SLO output.

J.MDTI: Warning, error has occurred in the input file. Check SLO output.

J.MDTI: Error opening SLO file. All SLO output will be suppressed.

Listed Output

All of the following messages are written to logical file code SLO and form the audit trail.

These particular messages are followed by further information on errors:

The following descriptors were added to the memory descriptor table:

The following descriptors were deleted from the memory descriptor table

All current input files have been deleted from the memory descriptor table

The following error messages are written to logical file code SLO if errors are generated:

OJ.MDTI: Cannot delete the following record from the memory descriptor table.

OJ.MDTI: Error adding the following record to the memory descriptor table.

OJ.MDTI: Error on input record listed below in column: column number.

OJ.MDTI: Error on input record listed below. Volume not in mounted volume table.

OJ.MDTI: Error on input record listed below. Option RID set with invalid RID.

OJ.MDTI: Error on input record listed below. Cannot allocate resource descriptor.

OJ.MDTI: Error on input record listed below. Cannot log resource descriptor.

OJ.MDTI: Error on input record listed below. RID does not match pathname.

OJ.MDTI: Error on input record listed below. Pathname does not match descriptor.

OJ.MDTI: Error occurred writing to temporary file. FCB status: status.

OJ.MDTI: Unable to assign or open input file. Reason: reason

OJ.MDTI: Error on input file option line. Valid options: LOF, HLT, RID, BLD

OJ.MDTI: Unable to read input file. FCB status: status

OJ.MDTI: Unable to process run parameters. Pathname too long.

OJ.MDTI: Fatal error trying to locate system volume.

OJ.MDTI: Error occurred in creating temporary file for option BLD output.

OJ.MDTI: Error occurred in allocation of temporary file for option BLD.

OJ.MDTI: Error occurred in locating the system volume.

OJ.MDTI: Error! Starting block number too great. Check SYSGEN BLOCK= directive.

OJ.MDTI: Error! Not enough memory to build MDT.

OJ.MDTI: All valid input files have been processed.

OJ.MDTI: Error occurred in renaming temporary file to input file.

OJ.MDTI: Warning. Volumes should be public and not multiported discs.

CHAPTER 6

SHADOW UTILITY (J.SHAD)

6.1 General Description

The Shadow Utility (J.SHAD) specifies the portions of a task's logical address space (excluding the TSA) that are to be located in shadow memory. The specified portions are shadowed each time the task is activated until the shadowed area is redefined by the Shadow Utility. Specification alters the load module's RRS by adding type 11 RRSs (assign to shadow memory) to the load module's preamble. The portions to be shadowed are specified using relative or absolute addresses.

6.2 Accessing J.SHAD

J.SHAD is accessed in the interactive and batch modes.

Syntax:

J.SHAD

When accessing J.SHAD interactively, the J.S> prompt is displayed. Enter the name of the task to be shadowed:

J.S> taskname

Enter Shadow directives at the further prompts. The Shadow directives are processed as they are read.

Note: When J.SHAD is activated, any existing type 11 RRS is cleared.

6.3 Logical File Code Assignments

There are three logical file codes (LFCs) associated with the Shadow Utility: Input File (SYC), Output File (UT), and Load Module File (INT). See Table 6-1.

6.3.1 Input File (SYC)

The input file contains the J.SHAD directives. Input is assigned to logical file code SYC.

SYC Default and Optional Assignments

In the interactive mode, the default assignment for SYC is to logical file code UT. In the batch mode, the default assignment for SYC is to the System Control file (SYC). There is no optional assignment to SYC.

6.3.2 Output File (UT)

The output file contains error messages. Output is assigned to logical file code UT.

UT Default and Optional Assignments

In the interactive mode, the default assignment for UT is to logical file code UT. In the batch mode, the default assignment for UT is to the system file code SLO. There is no optional assignment to UT.

6.3.3 Load Module File (INT)

The load module file contains the load module specified by the pathname that is read from the first record of the SYC. The load module is dynamically assigned to logical file code INT.

INT Default and Optional Assignments

There are no default or optional assignments for INT.

6.3.4 LFC Summary

The following is a table of LFCs used by J.SHAD and their default and optional assignments.

Table 6-1

J.SHAD LFC Summary

LFC	Default Assignment	Optional Assignment
SYC (interactive)	UT	None
SYC (batch)	SYC	None
UT (interactive)	UT	None
UT (batch)	SLO	None
INT	None	None

6.4 Shadow Utility Directives

6.4.1 EXIT Directive

The Exit directive exits the Shadow Utility.

Syntax:

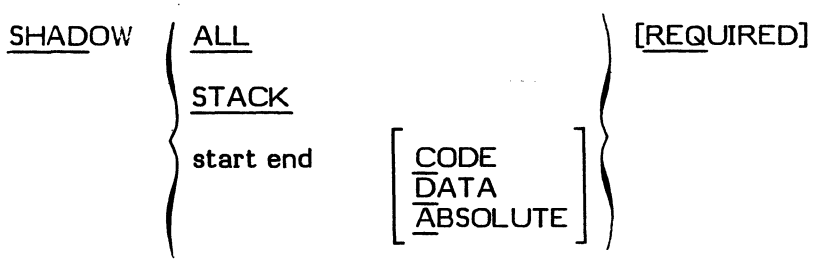
EXIT

Note: The Shadow Utility is also exited by entering CTRL C in the interactive mode or a $ card in the batch mode.

6.4.2 SHADOW Directive

The SHADOW directive specifies portions of a task to be located in shadow memory.

Syntax:



start — is the hexadecimal logical starting address of the logical address space to be shadowed

end — is the hexadecimal logical ending address of the logical address to be shadowed

CODE — specifies that the start and end addresses are relative to the task's code section origin, as opposed to the default assumption that they are relative to the task's start

DATA — specifies that the start and end addresses are relative to the task's data section origin, as opposed to the default assumption that they are relative to the task's start

ABSOLUTE — specifies that the start and end addresses are absolute. They are relative to logical address zero.

ALL — specifies that the entire task is to be shadowed except the TSA

STACK — specifies that the task's stack is to be shadowed. This parameter applies only to base mode tasks and is ignored for nonbase mode tasks.

REQUIRED specifies that shadow memory is required by the logical address space and the task can wait until shadow memory is available. If REQUIRED is not used and if shadow memory is not available, then E or S class memory is allocated to the logical address space.

Note: MPX-32 does not support shadowing of the Task Service Area (TSA).

6.5 Error Messages

When one of the following error messages is displayed, J.SHAD aborts with an SH01 abort code.

BLANK FIELD IS NOT ALLOWED
Blank fields are not allowed. Substitute a required argument for the blank field.

UNRECOGNIZED CHARACTER
An invalid character was found in a single character argument.

TOO MANY DIGITS
The numerical value entered is out-of-range.

NONNUMERIC VALUE
A nonnumeric value is supplied where a numeric value was expected.

EXTRA FIELD
An unexpected argument was supplied.

RRS SPACE IS UNAVAILABLE
Required RRS space in the specified task's preamble was not available for the shadow type RRS entry.

SHADOW END ADDR < START ADDR
The end address supplied is less than the starting address.

UNABLE TO OPEN OUTPUT
J.SHAD is unable to open the output file. This is caused by the lack of disc space for SLO in the batch mode.

UNABLE TO OPEN INPUT
J.SHAD is unable to open the input file.

UNABLE TO OPEN LOAD MODULE
J.SHAD is unable to open the specified load module name. This is caused by specifying a nonexisting name for the desired file or not having the required access to modify the file.

6.6 Examples

Shadow all of a task with shadow memory not required.

```
TSM> J.SHAD
J.S> @SYSTEM^(SYSTEM)MYTASK
J.S> SHAD ALL
J.S> EXIT
TSM>
```

Shadow stack and code area of a task with shadow memory required.

```
TSM> J.SHAD
J.S> @SYSTEM^(SYSTEM)BASETASK
J.S> SHADOW STACK REQUIRED
J.S> SHAD 0 2048 C REQ
J.S> X
TSM>
```

Clear shadow RRSs in task.

```
TSM> J.SHAD
J.S> USERTASK
J.S> EXIT
TSM>
```

Shadow all of a task in batch with shadow memory not required.

```
$JOB SHADOW
$J.SHAD
USERTASK
SHADOW ALL
$EOJ
$
```

CHAPTER 7

ANSI LABELED TAPES

7.1 General Information

The ANSI labeled tape utilities allow the usage of ANSI labeled tapes on MPX-32 Release 3.3 and later systems. These MPX-32 systems process ANSI labeled tapes according to implementation level four of ANSI standard X3.27-1978. A copy of the standard can be obtained from the American National Standards Institute, Inc., 1430 Broadway, New York, NY 10018.

Utilities associated with ANSI labeled tapes are:

- Dismount ANSI Labeled Tape Utility (ADMOUNT)
- Mount ANSI Labeled Tape Utility (AMOUNT)
- Display ANSI Labeled Tape Utility (ASTAT)
- Log ANSI Labeled Tape Utility (AVOLM)
- Label ANSI Tape Utility (J.LABEL)

For more information, see the individual sections in this chapter.

The ANSI standard allows 17 character file names. The ANSI labeled tape utilities also read tapes with lowercase letters and underscores in the labels. ANSI labeled tapes must be used for data transfers to non-MPX-32 systems that require ANSI labeled tapes. The ANSI labeled tape processing is slower than the regular single volume and multivolume modes, and is performed by a user-transparent task called J.ATAPE. All tape positioning operations are performed by the system.

ANSI labeled tapes can be collected into sets. Each tape set consists of one or more tapes and can contain multiple files. Files can be continued from one tape to the next. A tape set is identified by a logical volume identifier (LVID) that is specified when the tape is mounted or assigned.

7.2 Usage

Before generating an ANSI labeled tape set, each tape must be labeled with a Volume Identifier (VID) using the Label ANSI Tape Utility (J.LABEL). The label contents can be logged by the Log ANSI Labeled Tape Utility (AVOLM) to ensure that the character strings are in accordance with the ANSI standard.

After the tape set is labeled, it is mounted using the Mount ANSI Labeled Tape Utility (AMOUNT). This utility allows an owner to associate a Logical Volume Identifier (LVID) with a tape set. To check the currently mounted LVIDs and the associated owner names, use the Display ANSI Labeled Tape Utility (ASTAT).

Once mounted, a tape set is accessible only by tasks running with the same owner name as the mounter. Access to a tape set is made by specifying the LVID.

Assignment to files on an LVID is by TSM, or M.ASSN or M_ASSIGN with an RRS type ten. If a new file is required for write or append access, it is created when the file is opened. Similarly, if a specified file must be located for read or update access, the tapes are searched for the file when the file is opened. Files on an LVID can be accessed in any order for the read mode. The system dynamically builds a file directory of the tape's contents as it scans. This allows rapid access with minimum tape switching after a file location is known. The Log ANSI Labeled Tape Utility (AVOLM) produces a list of the contents of an ANSI labeled tape set for the user.

If a file is accessed in a mode other than read, the size of the file contents can change. Because of this, a file accessed in a mode other than read becomes the last file on an LVID.

After a tape set has been labeled, mounted, and assigned, the only performable I/O operations are: OPEN, READ, WRITE, CLOSE, ASSIGN, DEASSIGN, and WEOF.

When a tape set is no longer being used, dismount the LVID associated with the tape set with the Dismount ANSI Labeled Tape Utility (ADMOUNT). This removes the LVID and its directory from the system.

7.2.1 File Records

Files on ANSI labeled tapes consist of records that can be fixed length, variable length or spanned format. The record format and length is specified at assignment. See $ASSIGN directive. Records are also specified blocked or unblocked at assignment. If blocked records are specified, the record block is filled with as many records as possible. Any remaining block space is padded with circumflexes (^).

Fixed-length Records

Fixed-length records on an ANSI labeled tape set contain only user-readable data. The fixed-length records are equal in length.

Variable-length Records

Variable-length records on an ANSI labeled tape set are preceded by a four-byte Record Control Word (RCW). The RCW contains the length of the record plus the length of the RCW in ASCII. For example, if the record length is 80, the RCW is 0084. If the record format is not specified, the default is variable-length records.

Spanned Records

Spanned records can be split over record blocks and tapes belonging to the same tape set. These records can be any length and are preceded by a Segment Control Word (SCW). The SCW consists of a Segment Control Byte (SCB) followed by an RCW. The SCB allows a logical record to be split over a number of physical records by defining the record segment. Values contained in the SCB are defined as follows:

ASCII character	Description
0	Record begins and ends in this block
1	Record begins but does not end in this block
2	Record does not end or begin in this block
3	Record ends but does not begin in this block

7.2.2 Tape Drives for ANSI Labeled Tapes

Tape drives can be specified for ANSI labeled tapes by the SYSGEN DEVICE directive. See Table 7-1 for more information. No restrictions apply for the system administrator.

Table 7-1
Tape Drives for ANSI Labeled Tapes

Tape drive specification	Result
No drives are specified as ANSI	All tape drives can process ANSI and non-ANSI tapes
One or more drives are specified as ANSI	Tape drives specified as ANSI can only process ANSI tapes. Tape drives not specified as ANSI can only process non-ANSI tapes.

7.2.3 ANSI Labeled Tape Labels

There are two groups of ANSI tape labels: system labels and user labels. Both groups of labels consist of headers and trailers. Detailed information on various header and trailer formats can be found in the ANSI standard. The system labels are generated internally and by the Label Utility (J.LABEL). User labels are optional and can be read and written as follows:

Read ANSI Tape User Labels

To read ANSI tape user labels, set the DFI bit in the FCB when the file is opened. If any ANSI tape user header labels are present, they are transferred to a user-specified location, one per read request. The end of user header labels are indicated when the system returns an end of file (EOF). From this point, DFI is ignored, and the file data is returned for each read request until the system signifies end of file data by returning an EOF. If subsequent reads are performed with DFI set, any user trailer labels are returned to a user-specified location. The end of the user trailer labels is then indicated by the system returning an EOF.

Write ANSI Tape User Labels

To write ANSI tape user labels, set the DFI bit in the FCB on each write to the file. The system then accepts the user header labels that conform to the ANSI standard. When all ANSI tape user header labels have been written, the DFI bit must be reset. This causes subsequent records to be treated as file data. If required, user trailer labels can be written after a write end of file (WEOF). Any additional write operations with the DFI bit set are user trailer labels.

7.2.4 ANSI Tape Interchange with Other Systems

The ANSI standard specifies four levels of implementation. MPX-32 is fully implemented to level four. For information on implementation levels see Table 7-2. A tape's level of implementation must be compatible with the target system's level of implementation. For example, level one, two, three, and four tapes can be processed on a level four target system. However, tapes with spanned length records cannot be processed on a target system lower than level four.

Table 7-2
ANSI Tape Implementation Levels

Level	Description
1	Single file of fixed length records on single or multivolume tape sets
2	Multiple files of fixed length records on single or multivolume tape sets
3	Multiple files of fixed or variable length records on single or multivolume tape sets
4	Multiple files of fixed, variable, or spanned length records on single or multivolume tape sets

7.2.5 ANSI Labeled Tape Messages

The following messages are produced by the system, and are displayed to the operator's console:

```
PREPARE FOR ANSI MOUNT
lvid=vid [,vid... ] [DEV=devmnc]
```

When an AMOUNT is issued, this message is displayed. The second line of the message is the AMOUNT parameters.

```
MOUNT vid ANSI ON devmnc REPLY R, A, H, OR DEVICE:
```

On the first I/O request from the user, this message is displayed prompting the user to mount the tape. To continue processing the task, enter R (resume). To abort the task, enter A (abort). To hold the task, enter H (hold).

```
DISMOUNT vid ANSI FROM devmnc
```

When a tape change is required, the tape rewinds and this dismount message is displayed.

```
INCORRECT ANSI TAPE vid MOUNTED FOR vidl
```

If the incorrect tape is mounted, this message is displayed.

SUBSTITUTE ANSI TAPE vid FOR vid1? (Y/N)

When the system administrator is the requestor and the VID of the tape that is being mounted does not correspond to the VID specified by the AMOUNT utility, this message is displayed. Respond Y or N depending on the desired effect.

THAT TAPE IS NOT ANSI FORMAT

When a non-ANSI tape is mounted on an ANSI tape drive, this message is displayed.

ABORT TASK REQUESTING TAPE (Y/N)?

When an incorrect tape is mounted or the mounted tape is non-ANSI, this message is displayed. Respond Y or N depending on the desired effect.

FINISHED WITH ANSI LVID=lvid

When an ADMOUNT is issued, the tape rewinds and a normal dismount message is displayed followed by this message.

ANSI TAPE vid DOES NOT CONTINUE CURRENT FILE

When the tape is not the continuation of the file currently being processed, this message is displayed. Mount the correct tape.

7.2.6 Examples

Tape Labeling

The following job labels three tapes. The first is accessible by all owners. The second and the third are accessible only by OWNER1. Tape labeling is processed interactively or in batch mode.

```
TSM>
TSM>J.LABEL SYS001
TSM>J.LABEL MULTAA OWNER=OWNER1 PROT = 0
TSM>J.LABEL MULTAB OWNER=OWNER1 PROT = 0
TSM>
```

Tape Writing

The following job writes files to a multivolume tape set. The first file on the tape has fixed length, blocked records and is accessible only by OWNER1. Each record consists of 132 characters. Seven records are packed into a block with 100 bytes of padding. This file cannot be overwritten until April 6, 1986. The second file has variable length blocked records. This file cannot be overwritten until the termination date of the first file or 30 days after the date the job was run, whichever is earlier.

```
$AMOUNT SAMTAP=MULTAA,MULTAB
$ASSIGN SI TO TEST.DATA.01
$ASSIGN SO TO @ANSITAPE(SAMTAP)FILE#TEST#DATA#01 GENE=1-
GENV=2 ACCE=(A) FORM=F RECL=132 BSIZE=1024-
PROT=0 EXPIRE=86096
$MEDIA
```

```
COPY,SI,SO
EXIT
END
$ASSIGN SI TO TEST.DATA.02
$ASSIGN SO TO @ANSITAPE(SAMTAP)FILE2 ACCE=(A)
$MEDIA
COPY,SI,SO
EXIT
END
$ADMOUNT SAMTAP
```

Tape Reading

The following job copies the contents of two files produced by the previous job to the printer. The format in the files is deduced from the tape. The files can be accessed in any sequence.

```
$AMOUNT ATLIST=MULTAA,MULTAB
$AS SI1 TO @ANSITAPE(ATLIST)FILE2
$AS SO TO SLO
$MEDIA
COPY,SI1,SO
EXIT
END
$AS SI1 TO @ANSITAPE(ATLIST)FILE#TEST#DATA#01
$AS SO TO SLO
$MEDIA
COPY,SI1,SO
EXIT
END
$ADMOUNT ATLIST
```

7.3 ANSI Labeled Tape Utilities

The following are the ANSI labeled tape utilities and are documented in this section:

- Dismount ANSI Labeled Tape (ADMOUNT) Utility
- Mount ANSI Labeled Tape (AMOUNT) Utility
- Display ANSI Labeled Tape (ASTAT) Utility
- Log ANSI Labeled Tape (AVOLM) Utility
- Label ANSI Tape (J.LABEL) Utility

Each utility is documented in alphabetical order in this section.

Each utility has one LFC called OUT that is assigned to LFC=UT at CATALOG time.

7.3.1 Dismount ANSI Labeled Tape Utility (ADMOUNT)

The Dismount ANSI Labeled Tape Utility (ADMOUNT) dismounts an ANSI labeled tape by removing its LVID from the system tables. Removal of the LVID causes the tape to rewind and dismount. ADMOUNT can be activated only by the owner that activated the corresponding AMOUNT utility.

Syntax:

ADMOUNT lvid

lvid is the one- to six-character identifier specified in the corresponding AMOUNT

On successful completion of ADMOUNT, the following message is displayed to lfc OUT and ADMOUNT is exited:

ANSI VOLUME DISMOUNTED

ADMOUNT Error Messages

The following are the ADMOUNT error messages and their explanations:

ERROR - VID MUST BE 1 THROUGH 6 CHARACTERS IN LENGTH

ERROR - LOGICAL VOLUME IDENTIFIER MUST BE SPECIFIED

ERROR - ONLY ONE PARAMETER CAN BE SPECIFIED

ERROR - NOT MOUNTED

The specified LVID has not been mounted by this owner.

ERROR - VOLUME IS ACTIVE

There is an active assignment to a file on the LVID.

ERROR - INVALID J.ATAPE START STATUS
RETURNED STATUS IS nn

J.ATAPE could not be run requested. See M.SRUNR in Volume I, Chapter 6 for information on the returned error status.

ERROR - NO J.ATAPE TABLE SPACE IS AVAILABLE FOR ANSI PROCESSING

There is insufficient memory space for the ANSI labeled tape processing. Wait for available memory.

7.3.2 Mount ANSI Labeled Tape Utility (AMOUNT)

The Mount ANSI Labeled Tape Utility (AMOUNT) mounts an ANSI labeled tape by associating an LVID with a set of tapes. The LVID must be unique at the time of issuing the request. The LVID remains mounted until a corresponding ADMOUNT is issued for the same LVID by the same owner name.

Syntax:

AMOUNT lvid=vid [,vid....] [DEV=devmnc]

lvid is a one- to six-character identifier that accesses a set of tapes

vid is a one- to six-character volume identifier of each tape that specifies the order that the tapes are accessed. Up to ten volume identifiers can be specified.

devvmnc specifies the two-, four- or six-character device mnemonic of the tape drive to be used

On successful completion of AMOUNT, the following message is displayed to LFC OUT, and AMOUNT is exited:

ANSI LOGICAL VOLUME MOUNTED

AMOUNT Error Messages

The following are the AMOUNT error messages and their explanations:

ERROR - USE THE FORMAT AMOUNT LVID=VID [,...] [DEV=DEVMNC]

ERROR - LVID MUST BE FOLLOWED BY =VID [,...]

ERROR - VID MUST BE 1 THROUGH 6 CHARACTERS IN LENGTH

ERROR - LAST PARAMETER (IF PRESENT) MUST BE DEV=DEVMNC

ERROR - FOLLOW 'DEV' WITH =DEVMNC

ERROR - DEVMNC MUST BE 2, 4, OR 6 CHARACTERS LONG

ERROR - DEVICE SPECIFIED IS NOT MAGNETIC TAPE

ERROR - CHANNEL OR SUBCHANNEL MUST BE HEXADECIMAL

ERROR - DEVICE IS NOT CONFIGURED IN SYSTEM

ERROR - TOO MANY VOLUME IDS SPECIFIED MAX=10

ERROR - INVALID J.ATAPE START STATUS
RETURNED STATUS IS nn

J.ATAPE could not be run requested. See M.SRUNR in Volume I, Chapter 6 for information on the returned error status.

ERROR - NO J.ATAPE TABLE SPACE IS AVAILABLE FOR ANSI PROCESSING

There is insufficient memory to process the ANSI labeled tape. Wait for available memory.

ERROR - DUPLICATE LVID

The specified LVID is already in use. Choose a different LVID or wait until the specified LVID is available.

7.3.3 Display ANSI Labeled Tape Utility (ASTAT)

The Display ANSI Labeled Tape Utility (ASTAT) displays the currently mounted LVIDs and their associated owner names to LFC OUT. The response is a headed list of owner names and LVIDs. If no volumes are currently mounted, only the list's headers are displayed.

Syntax:

```
ASTAT
```

Example

```
TSM> ASTAT
     OWNER          LOGICAL VOLUME ID     } Header

     OWNER1         VOL1
     OWNER2         VOL2                  } List
```

ASTAT Error Messages

The following are ASTAT error messages and their explanations:

```
DUPLICATE LVID
```

The specified LVID is already in use. Choose a different LVID or wait until the specified LVID is available.

```
ERROR - INVALID J.ATAPE START STATUS
RETURNED STATUS IS nn
```

J.ATAPE could not be run requested. See M.SRUNR in Volume I, Chapter 6 for information on the returned error status.

```
ERROR - NO J.ATAPE TABLE SPACE IS AVAILABLE FOR ANSI PROCESSING
```

There is insufficient memory for ANSI labeled tape processing. Wait for available memory.

7.3.4 Log ANSI Labeled Tape Utility (AVOLM)

The Log ANSI Labeled Tape Utility (AVOLM) performs any of the following functions:

- List format information and volume identifier's file identifier
- List the contents of an ANSI labeled tape set

AVOLM checks the label contents to ensure that the contents are valid character strings according to the ANSI standard.

When AVOLM is performed on an ANSI labeled tape with an underscore in the label, a format error is displayed and the file can be restored.

Syntax:

AVOLM [vid [,vid...]] [DEV=devmnc] [BRIEF={Y / N}]

vid — is a one- to six-character volume identifier of each tape. Up to ten volume identifiers can be specified. If more than one VID is specified, they must form a tape set.

devmnc — specifies the two-, four-, or six-character device mnemonic of the tape drive to be used.

BRIEF — If BRIEF=N is specified, each label on the tape is displayed and the number of data blocks used is verified. If BRIEF=Y is specified, the following information is produced:

- VOL1 — vid, owner, label standard version number. The VOL1 label appears at the beginning of each tape. No owner is displayed if the owner option was not specified at J.LABEL.
- HDR1 — file identifier, generation number, generation version number, sequence number, section number, creation date, termination date, format, block size, record size, and the level of the ANSI tape. HDR1 information is produced once per file section on a tape set.

If not specified, the default is Y.

AVOLM Error Messages

The following are the AVOLM error messages:

ERROR - OPTIONS MUST FOLLOW THE VID LIST
ERROR - VID MUST BE 1 THROUGH SIX CHARACTERS IN LENGTH
ERROR - FOLLOW 'BRIE' BY Y OR N
ERROR - UNKNOWN OPTION
ERROR - TOO MANY VID'S SPECIFIED
ERROR - REMAINING VIDS DO NOT CONTINUE THE TAPE SET
ERROR - DEVMNC MUST BE 2, 4, OR 6 CHARACTERS IN LENGTH
ERROR - DEVICE SPECIFIED IS NOT MAGNETIC TAPE
ERROR - CHANNEL OR SUBCHANNEL MUST BE HEXADECIMAL
ERROR - DEVICE IS NOT CONFIGURED IN SYSTEM

7.3.5 Label ANSI Tape Utility (J.LABEL)

The Label ANSI Tape Utility (J.LABEL) writes the initial header labels to new tapes. J.LABEL can be used only by the system administrator. The ANSI tape labeling process is completed only once per tape because the label is preserved by the system. All ANSI tapes must be labeled with J.LABEL to be recognized by the system.

Syntax:

$J.LABEL vid [DEV=devmnc] [OWNER=ownername] [PROTECT=p]

vid — is a one- to six-character ANSI volume identifier

devmnc — specifies the two-, four-, or six-character device mnemonic of the tape drive to be used

ownername — is the owner name for the tape

p — is a one-character identifier of the protection to be applied. Zero specifies owner only access. A...Z are reserved by the ANSI specification for installation specific protection. MPX-32 treats A...Z as owner-only protection. If not specified, the default is no protection.

On successful completion, the processor is exited.

J.LABEL Errors

The following are J.LABEL error messages:

ERROR - USE FORMAT $J.LABEL VID [DEV=DEVMNC] [OWNER=OWNERNAME] [PROT=P]

ERROR - OPTION MUST BE FOLLOWED BY '='

ERROR - OPTION KEYWORD NOT RECOGNIZED

ERROR - INVALID OWNER NAME LENGTH

ERROR - FOLLOW 'DEV' WITH =DEVMNC

ERROR - PROTECTION MUST BE 1 CHARACTER

ERROR - INVALID PROTECTION VALUE

ERROR - MAGNETIC TAPE ASSIGNMENT FAILURE
This error is generaged if the specified device is off-line.

ERROR - DEVMNC MUST BE 2, 4, OR 6 CHARACTERS LONG

ERROR - DEVICE SPECIFIED IS NOT MAGNETIC TAPE

ERROR - CHANNEL OR SUBCHANNEL MUST BE HEXADECIMAL

ERROR - DEVICE IS NOT CONFIGURED IN SYSTEM

ERROR - SYSTEM ADMINISTRATOR USE ONLY

Gould Inc., Computer Systems Division
6901 W. Sunrise Blvd.
P. O. Box 409148
Fort Lauderdale, FL 33340-9148
Telephone (305) 587-2900

GOULD
Electronics

Users Group Membership Application

USER ORGANIZATION: _____

REPRESENTATIVE(S): _____

_____

_____

ADDRESS: _____

_____

TELEX NUMBER: _____ PHONE NUMBER: _____

NUMBER AND TYPE OF GOULD CSD COMPUTERS: _____

_____

OPERATING SYSTEM AND REV. LEVEL: _____

_____

APPLICATIONS (Please Indicate)

1. EDP
   - A. Inventory Control
   - B. Engineering & Production Data Control
   - C. Large Machine Off-Load
   - D. Remote Batch Terminal
   - E. Other

2. Communications
   - A. Telephone System Monitoring
   - B. Front End Processors
   - C. Message Switching
   - D. Other

3. Design & Drafting
   - A. Electrical
   - B. Mechanical
   - C. Architectural
   - D. Cartography
   - E. Image Processing
   - F. Other

4. Industrial Automation
   - A. Continuous Process Control Op.
   - B. Production Scheduling & Control
   - C. Process Planning
   - D. Numerical Control
   - E. Other

5. Laboratory and Computational
   - A. Seismic
   - B. Scientific Calculation
   - C. Experiment Monitoring
   - D. Mathematical Modeling
   - E. Signal Processing
   - F. Other

6. Energy Monitoring & Control
   - A. Power Generation
   - B. Power Distribution
   - C. Environmental Control
   - D. Meter Monitoring
   - E. Other

7. Simulation
   - A. Flight Simulators
   - B. Power Plant Simulators
   - C. Electronic Warfare
   - D. Other

8. Other

Please return to:

Users Group Representative

Date: _____

Gould Inc., Computer Systems Division Users Group. . .

The purpose of the Gould CSD Users Group is to help create better User/User and User/Gould CSD communications.

There is no fee to join the Users Group. Simply complete the Membership Application on the reverse side and mail to the Users Group Representative. You will automatically receive Users Group Newsletters, Referral Guide and other pertinent Users Group activity information.

Fold and Staple for Mailing

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 947 FT. LAUDERDALE, FL

POSTAGE WILL BE PAID BY ADDRESSEE

GOULD INC., COMPUTER SYSTEMS DIVISION
ATTENTION: USERS GROUP REPRESENTATIVE
6901 W. SUNRISE BLVD.
P. O. BOX 409148
FT. LAUDERDALE FL 33340-9970

Fold and Staple for Mailing

(Detach Here)

GOULD
Electronics