MPX-32 ™

Processors and Utilities

Revision 3.5

Reference Manual Volume II

April 1990

323-001552-600

ENCORE

Encore Computer Corporation
6901 West Sunrise Boulevard
Fort Lauderdale, Florida 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.

Publication Order Number **323-001552-400**, (Revision 4, Release 3.4) was printed January, 1988.

Publication Order Number **323-001552-500**, (Revision 3.4U03) was printed October, 1989.

Publication Order Number **323-001552-600**, (Revision 3.5) was printed April, 1990.

This manual contains the following pages:

Title page
Copyright page
iii/iv through xxiii/xxiv
1-1 through 1-116
2-1 through 2-60
3-1 through 3-66
4-1 through 4-4
5-1 through 5-9/5-10
6-1 through 6-5/6-6
7-1 through 7-13/7-14
8-1 through 8-18
9-1 through 9-17/9-18
10-1 through 10-10
11-1 through 11-22
A-1 through A-7/A-8
B-1 through B-42
C-1 through C-32
D-1 through D-2
E-1 through E-2
F-1 through F-2
G-1 through G-2
H-1 through H-2
I-1/I-2
J-1 through J-2
K-1/K-2
L-1 through L-79/L-80
GL-1 through GL-10
IN-1 through IN-31/IN-32

Contents

Page

List of Figures

List of Tables

Documentation Conventions

Conventions used in directive syntax, messages, and examples throughout the MPX-32 documentation set are described below.

Messages and Examples

Text shown in `this distinctive font` indicates an actual representation of a system message or an example of actual input and output. For example,

```
VOLUME MOUNT SUCCESSFUL
```

or

```
TSM>!ACTIVATE MYTASK
TSM>
```

Lowercase Italic Letters

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

$NOTE *message*

means replace *message* with the desired message. For example,

```
$NOTE 10/12/89 REV 3
```

In system messages, lowercase italic 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 the input required to execute that directive. Uppercase bold letters indicate the minimum that must be entered. For example,

$ASSIGN *lfc* **TO** *resource*

means enter $AS or $ASSIGN followed by a logical file code, followed by TO and a resource specification. For example,

```
$AS OUT TO OUTFILE
```

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

```
TERMDEF HAS NOT BEEN INSTALLED
```

Brackets [ ]

An element inside brackets is optional. For example,

$CALL *pathname* [*arg*]

means supplying an argument (*arg*) is optional.

Multiple items listed within brackets means enter one of the options or none at all. The choices are separated by a vertical line. For example,

$SHOW [**CPU**TIME | **JOB**S | **USE**RS]

means specify one of the listed parameters, or none of them to invoke the default.

Items in brackets within encompassing brackets or braces can be specified only when the other item is specified. For example,

BACKSPACE **F**ILE [[**FIL**ES=] *eofs*]

indicates if *eofs* is supplied as a parameter, FIL= or FILES= can precede the value specified.

Commas within brackets are required only if the bracketed element is specified. For example,

LIST [*taskname*][, [*ownername*] [, *pseudonym*]]

indicates that the first comma is required only if *ownername* and/or *pseudonym* is specified. The second comma is required only if *pseudonym* is specified.

Braces { }

Elements listed inside braces specify a required choice. Choices are separated by a vertical line. Enter one of the arguments from the specified group. For example,

[**BLO**CKED={**Y** | **N**}]

means Y or N must be supplied when specifying the BLOCKED option.

Horizontal Ellipsis ...

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

$DEFM [*par*] [, *par*] ...

means one or more parameters (*par*) separated by commas can be entered.

Vertical Ellipsis

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

```
$DEFM SI,ASSEMBLE,NEW,OP
     .
     .
     .
$IFA %OP ASSM
```

means one or more directives have been omitted between the $DEFM and $IFA directives.

Parentheses ( )

In directive syntax, parentheses must be entered as shown. For example,

(*value*)

means enter the proper value enclosed in parentheses; for example, (234).

Special Key Designations

The following are used throughout the documentation to designate special keys:

<ctrl>	control key
<ret> or <CR>	carriage return/enter key
<tab>	tab key
<break>	break key
<bck>	backspace key
<del>	delete key

When the <ctrl> key designation is used with another key, press and hold the control key, then press the other key. For example,

<ctrl>C

means press and hold the control key, then press the C.

Change Bars

Change bars are vertical lines (|) appearing in the right-hand margin of the page for your convenience in identifying the changes made in MPX-32 Revision 3.5.

When an entire chapter has been changed or added, change bars appear at the chapter title only. When text within figures has changed, change bars appear only at the top and bottom of the figure box.

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:

- log on to MPX-32
- access any MPX-32 process
- 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
- construct loops in directive files to control processing
- nest directive files
- specify conditional alternative actions
- communicate with online users or the operator
- account for the use of computer resources
- logoff from MPX-32

The Terminal Services Manager (TSM) provides interactive, time-shared access to the MPX-32 system for terminals connected through IOP or MFP console ports, or CONCEPT 32/2000 windowing console ports, or ALIM, MFP, 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 online and send messages to terminals.

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. Bold print indicates the minimum valid abbreviations for each.

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:

Directive	Description
$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. Resource specification directives are:

Directive	Description
$ALLOCATE	overrides cataloged memory allocation for online tasks
$ASSIGN	supplies default assignments for logical file codes (LFCs) used by the task being executed/activated
$ASSIGN1	associates a permanent disk file (optionally unblocked) with an LFC
$ASSIGN2	associates a system SBO, SLO, SYC, or SGO file with an LFC
$ASSIGN3	associates a device with an LFC
$ASSIGN4	associates an LFC with another LFC
$CHANGE	establishes new default working directory and/or project group name
$CLEAR	clears previous $SHADOW, $ASSIGN, $OPTION, $ALLOCATE, and $EXTDMPX directives and terminates processing from a directive file before end-of-file
$CREATE	creates a permanent file
$DELETE	deletes a permanent file
$DISMOUNT	requests the logical dismount of a nonpublic volume from the current TSM context or requests the physical dismount of a user volume from a device
$EXTDMPX	dynamically overrides the SYSGEN and CatalogER assignment for the logical starting address of the TSA and extended MPX-32 when configured. When extended MPX-32 is not configured, this directive applies only to the TSA.
$MAPOUT	dynamically overrides the execution mode specified at SYSGEN when task was cataloged SYSMAP (task runs in the mapped out mode)
$MOUNT	requests the physical mount of a user volume to a device and the logical mount of a nonpublic volume to the current TSM context
$NOMAPOUT	dynamically overrides the execution mode specified at SYSGEN when task was cataloged SYSMAP.
$OPTION	provides options for 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

The following directives activate tasks and select directive files for execution:

Directive	Description
$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 name of a cataloged load module
$RUN	activates a task in the interactive or batch environment by specifying the pathname of a cataloged load module
$SELECT	executes a file of command processor directives

1.2.4 Terminal and Batch Control Directives

The following directives specify terminal and job environment conditions. They are valid in the interactive mode only.

Directive	Description
$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 screen length defined for a terminal
$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

The following directives receive or transmit general information regarding system usage:

Directive	Description
$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
$RECALL	retrieves one or more commands from the recall buffer for processing
$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 users
$SIGNAL	sends a message to a logged-on user or to all terminals
$WHO	displays a list of logged-on 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:

Directive	Description
$DEFM	defines parameters for a directive file
$CALL	transfers control of processing to a nested directive file (not valid in batch mode)
$DEFNAME	labels a point to branch to for conditional processing
%label	labels a point to branch to for conditional processing
$GOBACK	branches backward to a label in a directive file
$GOTO	branches forward to a label in a directive file
$IFA	branches to a label if the specified argument is absent
$IFP	branches to a label if the specified argument is present
$IFF	branches to a label if a condition is false
$IFT	branches to a label if a condition is true
$RESETF	sets flag(s) to false for conditional processing
$SET	assigns a new string value to a parameter
$SETI	assigns an integer value or the results of an arithmetic expression to a parameter
$SETF	sets flag(s) to true for conditional processing
$END	terminates a directive file without clearing $SHADOW, $ASSIGN, $OPTION, $ALLOCATE, or $EXTDMPX directives
$ENDM	defines the end of a directive file (optional)

1.2.7 Batch Input Spooling Directives

Batch 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:

Directive	Description
$OBJECT	precedes program object records
$SELECTD	spools batch records to the SYC file from the specified peripheral device. Data on the device is in library format.
$SELECTF	spools batch records to the SYC file from the specified permanent disk file. Data in the file is in library format.
$SELECTLD	spools batch records to the SYC file from the specified peripheral device
$SELECTLF	spools batch records to the SYC file from the specified permanent disk file
$SELECTS	resets alternate system input source levels established by previous $SELECT*x* directives
$$	terminates batch input stream
$$$	terminates batch input stream when continuous batch mode was requested

1.2.8 Logon Control Directives

Logon control directives are used to enable and disable logons to TSM devices. These directives are available only to the system administrator.

$DISABLE	disables logons to one or all TSM devices
$ENABLE	enables logons to one or all TSM devices

1.3 Logging On to TSM

To log on to MPX-32 from a terminal, press the wake-up character defined in the LOGONFLE for the terminal (such as <ctrl>E or ?). TSM responds:

```
MPX-32 RELEASE x.x patch iname TERMINAL SERVICES MANAGER
(C) COPYRIGHT 1989, ENCORE COMPUTER CORPORATION. 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 1- to 8-character owner name. All characters but the following can be used in owner names:

- blanks
- commas
- semicolons
- equal signs
- line feeds
- dollar signs
- exclamation points
- percent signs
- left or right parentheses

If another terminal is logged on with that owner name and multiple logons was not specified at SYSGEN, TSM displays the following message:

```
NAME IN USE
TRY AGAIN:
```

The `TRY AGAIN:` prompt is issued until a valid owner name not already in use is entered.

The characters of the owner name will appear on the terminal screen as they are typed unless echoing of owner name characters has been suppressed by the SYSGEN OWNERNAME directive. (See MPX-32 Reference Manual Volume III, Chapter 7, for more information on the OWNERNAME directive.)

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

Local Echoplex Terminals:

`ENTER PASSWORD`	Password characters echoed
or	
`ENTER KEY:`	Key characters echoed

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

`ENTER PASSWORD`	Password characters not echoed
or	
`ENTER KEY:`	Key characters not echoed

If a password or key was established in the M.KEY file, the password or key must be entered to gain access to the system. If the owner has a password in the M.KEY file, the user is prompted for the password, whether or not a key also exists. If the owner has no password, the `ENTER KEY` prompt appears. If a key does not exist and one is not to be established, enter a carriage return at the key prompt.

If an M.KEY file does not exist, no prompt is displayed.

When there is no password associated with the owner name, a key may be changed, deleted or added at logon. 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.

Owner keys and passwords have the same character restrictions as owner names (see above).

MPX-32 also provides the KEYWORD and the PASSWORD tasks that can be used during a logon session to change a key or a password, respectively.

1.3.1 Changing a Key

To change the key during a logon session using the KEYWORD task, enter:

```
TSM>KEYWORD
```

The KEYWORD task responds with the following prompts:

```
ENTER CURRENT KEYWORD : keyword
ENTER NEW KEYWORD : keyword
REENTER NEW KEYWORD : keyword
```

keyword specifies a 1- to 8-character alphanumeric key

If the key was changed correctly, the KEYWORD task exits and the `TSM>` prompt is displayed. If the current key or the new key is entered incorrectly, the KEYWORD task exits and displays the following message on the terminal:

```
INVALID KEYWORD, KEYWORD NOT CHANGED
```

Attempting to change the key without write access to the M.KEY file causes KEYWORD to display the following message:

```
ERROR ENCOUNTERED WHEN UPDATING M.KEY FILE.  CHECK FILE ACCESS.
```

1.3.2 Changing a Password

To change the password during a logon session, use the PASSWORD task as follows:

```
    TSM>PASSWORD
```

The PASSWORD task responds with the following prompts:

```
    ENTER CURRENT PASSWORD : password
    ENTER NEW PASSWORD : password
    REENTER NEW PASSWORD : password
```

password	specifies a 1- to 16-character alphanumeric password. To specify no password, enter the return key.

If the password successfully was changed, the PASSWORD task exits and displays the following message on the terminal:

```
    PASSWORD CHANGED
```

If the current password is not entered when prompted or the new password is entered incorrectly, the PASSWORD task exits and displays the following message on the terminal:

```
    INVALID PASSWORD, PASSWORD NOT CHANGED
```

Attempting to change the password without write access to the M.KEY file causes PASSWORD to display the following message:

```
ERROR ENCOUNTERED WHEN UPDATING M.KEY FILE, CHECK FILE ACCESS.
```

Notes:

The PASSWORD task does not echo the password.

The PASSWORD task can be overridden with the SYSGEN SAPASSWD directive. For more information see the System Generation (SYSGEN) chapter in Volume III of the MPX-32 Reference Manual.

If the owner name and/or key or password is not valid, TSM displays the following message:

```
    UNAUTHORIZED NAME, KEY, OR PASSWORD
    TRY AGAIN:
```

The `TRY AGAIN:` prompt is issued until a valid owner name and key or password are entered. Press the carriage return to log off the terminal.

If the owner name and key are valid, TSM checks to see if the key was changed. 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. A project group is changed with the $CHANGE PROJECT directive. When changed, any nonzero CPU and IPU execution time for the previous session is recorded in the M.ACCNT file under the old project group.

If an M.PRJCT file exists and an owner has a default project name not contained in M.PRJCT, no project name is established at logon.

If a user's default directory cannot be established at logon due to the M.KEY file not existing, or the M.KEY file lacking an entry for the user or an error occurring while accessing the M.KEY file, the user's working directory is set to @SYSTEM(SYSTEM).

If the M.KEY file exists, but the user's default volume cannot be located at logon, no default volume or directory is established for that user and the following occurs:

- a message displays at logon:

```
*YOUR DIRECTORY IS UNAVAILABLE, PLEASE CHANGE DIRECTORY
```

- the user can only execute TSM directives
- a $SHOW directive displays asterisks as the user's volume and directory name

The working directory specifies the pathname to be prefixed for file name specification. The working directory is changed by the $CHANGE DIRECTORY directive.

If the volume specified is nonpublic, a $MOUNT directive must be issued before issuing the $CHANGE DIRECTORY directive.

1.4 Operating Environments

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

1.4.1 Accessing the Batch Environment

Activate a task 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,...par16

TSM>$OPCOM
??BATCH jobfile

TSM>EDIT filecode
EDT>BATCH jobfile
```

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

```
jobno  owner   $EOJ  jobname
```

1.4.2 Accessing the Interactive Environment

Activate a task in the interactive environment by supplying the task name in response to the TSM prompt:

```
TSM>MYTASK
```

The task executes at the time-distribution priority of TSM and has special TSM support for breaks, wakeup characters, terminal assignments, and other functions. See the Executing Tasks Under TSM section of this chapter for more information.

1.4.3 Accessing the Real-Time Environment

Activate a task in the real-time environment through the Operator Communications (OPCOM) ACTIVATE or ESTABLISH directive. The task 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)       TSM>
??EXIT
TSM>
```

A task is also activated at its base priority in the real-time environment by the TSM $ACTIVATE directive. In batch mode, assignment and option directives can precede the $ACTIVATE directive 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.

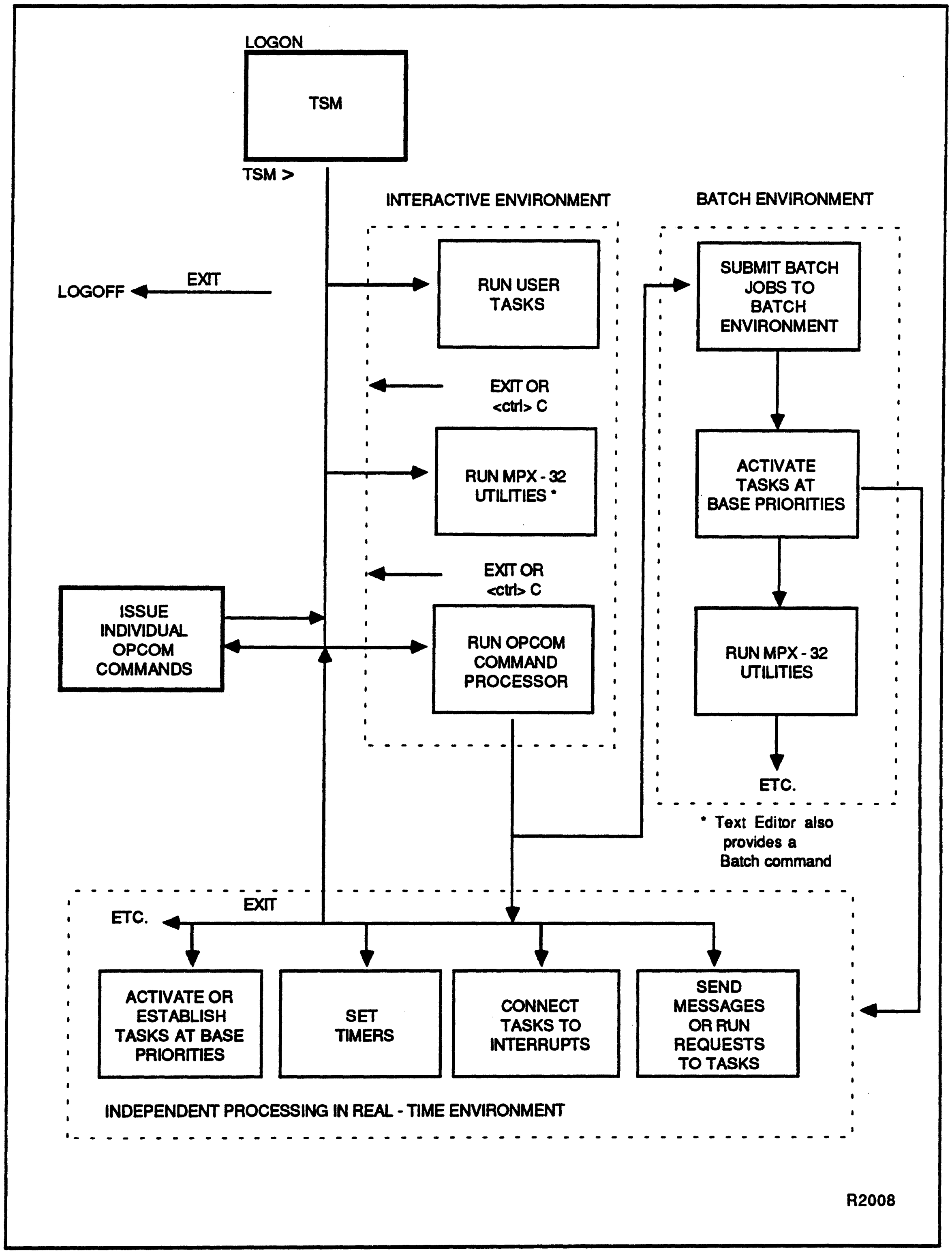

LOGON
TSM
TSM >
INTERACTIVE ENVIRONMENT
BATCH ENVIRONMENT
LOGOFF
EXIT
RUN USER TASKS
EXIT OR <ctrl> C
RUN MPX - 32 UTILITIES *
EXIT OR <ctrl> C
ISSUE INDIVIDUAL OPCOM COMMANDS
RUN OPCOM COMMAND PROCESSOR
SUBMIT BATCH JOBS TO BATCH ENVIRONMENT
ACTIVATE TASKS AT BASE PRIORITIES
RUN MPX - 32 UTILITIES
ETC.
* Text Editor also provides a Batch command
EXIT
ETC.
ACTIVATE OR ESTABLISH TASKS AT BASE PRIORITIES
SET TIMERS
CONNECT TASKS TO INTERRUPTS
SEND MESSAGES OR RUN REQUESTS TO TASKS
INDEPENDENT PROCESSING IN REAL - TIME ENVIRONMENT
R2008

Figure 1-1
Interactive/Batch/Real-Time Environments

1.5 Logging Off

The TSM prompt must be displayed to log off the terminal. Users in another processor or utility must specify the appropriate exit directive for the processor or use the <ctrl>C key sequence to return to the TSM prompt.

To log off, specify the TSM $EXIT directive in response to the TSM prompt. The following message displays:

```
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 also displays 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 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 a user who is not currently logged on. If a message is sent to all terminals, it is queued for output to terminals 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 TSM Special Keys

If an MPX.PRO file exists in the system or user directory and it enables command line recall and edit, the following special keys are available to TSM users:

- recall and edit keys assigned in the MPX.PRO file (see the Command Line Recall and Edit chapter in this manual)
- help key (definable through MPX.PRO)
- break key (<break>)
- wakeup character
- exit key (<ctrl>C)
- carriage return (<ret>) or <ctrl>M
- tab key (<tab>) or <ctrl>I
- delete (<del>) or rub

If no MPX.PRO file exists or if command line recall and edit is disabled, a smaller set of special keys is available to TSM users. These keys can delete a character or line of input, insert tabs, and perform other interactive functions. Table 1-1 lists and describes these TSM special keys.

Table 1-1
TSM Special Keys

Key	Description
carriage return (<ret>)	Terminates the current input or directive line. Also continues output listing after a pause at the bottom of the terminal screen.
<ctrl>H or <bck>	Deletes the character just typed (destructive backspace)*
rub or <del>	Deletes the current line. Must be used before a carriage return. The cursor moves to the next line.*
<ctrl>I or <tab>	Inserts blanks to next tab position*
<ctrl>C	Exits a processor when an EXIT directive is not available. Simulates EOF function from terminals.
<break>	Terminates I/O
wakeup	Re-establishes communication with a task or TSM
* When command line recall and edit is in effect, all keys except <tab>, <ctrl>C, <break>, and wakeup can be redefined.	

1.8 Executing Tasks Under TSM

Any cataloged load module runs when a TSM $RUN directive is issued. Tasks that are designed to run in any environment (interactive, 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 by TSM.

The $CLEAR directive clears assignments and options before a task is run. $CLEAR also terminates 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:

$directive [*arguments...*]

In most cases, the directive name 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 directive, 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 (the terminal in interactive mode). Write operations on the SYC file cannot be performed in the batch mode.

See the SYC and Terminal I/O section in this chapter for a description of close, open, and rewind operations on the SYC file.

1.8.2 Options

TSM options control various aspects of the user operating environment for tasks running interactively or in batch mode. See the $OPTION directive section in this chapter for more information on these options and their use.

1.8.2.1 Prompting Option

The TSM prompt option causes TSM to automatically precede any read from the terminal with the first three characters of the task name and a right angle bracket. The prompt option should be used only if the task does not write the prompt itself. This option is not valid in batch mode.

1.8.2.2 Lower Case Option

When running a task (e.g., the Volume Manager utility) interactively, the task can force a translation of lower case characters to upper case. The lower case option inhibits this translation and allows users to enter lower case characters. File names must be entered in all upper case. This option is not valid in batch mode.

1.8.2.3 Command/Nocommand Option

The command option echoes JCL directives to the terminal (for interactive tasks) or to the SLO file or device (for tasks in batch mode) as they are read from the SYC file. It is the default and can be overridden by the nocommand option.

The nocommand option inhibits the echoing of JCL directives as they are read from the SYC file.

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 the text option. This option affects only the task executed immediately after the option is set.

1.8.2.5 Retain Option

When activating a task, this option forces cataloged options to be ORed with the user-supplied activation time options. This option affects only the task executed immediately after the option is set.

1.8.2.6 Clear Option

When activating a task, this option forces all cataloged options to clear. This option affects only the task executed immediately after the option is set.

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. This is the default.

The noabort option inhibits the display of abort messages during a job or interactive session.

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 system console. This is the default and can be overridden by the unquiet option.

1.8.2.9 Unquiet Option

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

1.8.2.10 Error Option

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

1.8.2.11 Noerror Option

The noerror option disables the expansion of abort codes.

1.8.2.12 Wrap Option

The wrap option enables terminal line wrap. It is the default and can be overridden by the nowrap option. 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. If this option is not specified, the task is executed 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.

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 and can be 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.

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` to continue task execution. Enter `A` to abort the task. TSM 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` to attach the MPX-32 Debugger to the task. A debug prompt is then displayed. If the task is GCL or nonbase with a shared CSECT, the debugger is not attached and TSM displays the following message:

```
UNABLE TO ATTACH DEBUGGER - TASK CATALOGED NODEBUG, SHARED
TASK, or GCL TASK.
```

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

Enter `H` to place the task in a hold state. TSM displays the following message and prompt:

```
*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 a resource, enter the wakeup character used to log on. 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`. 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`. 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 the previous section.

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

1.8.5 Tabs

The M.KEY file can contain default tab settings for each 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 EDIT's SET TABS are effective while the user remains on the system. They are not saved at logoff. When the user logs on again, either the M.KEY tabs or the default tabs are set.

If command line recall and edit is disabled, during formatted input the tab character (<ctrl>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.

If recall and edit is enabled, the effect of the tab character depends on the current edit mode. In insert mode, the tab key shifts characters to the right. In modify mode, the tab key moves the cursor to the next tab without displacing characters.

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 disk 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
```

Pressing the return key causes TSM to write the next line on the screen and return to the task. If any character other than a 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. Precede all JCL directives with a dollar sign.

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 if the task uses the UT assignment for I/O. TSM turns control over to a task when it encounters a directive to activate the task.

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 SBO) are available to all tasks. SGO and certain options of SLO and SBO are available only within jobs.

1.10.3.1 SGO Files

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 SLO and SBO Files

SLO files are used for listed output. SBO files are used for punched output. Both are temporary, automatically extendible files created by the J.TSM command processor.

Access rights to 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=, SLOD=, SBOF=, SBOD=).

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 below.

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. If these control flags are set for a task's I/O, that task's I/O is not eligible for command line recall and edit processing.

1.11.1 Reads

On a read (M.READ), the maximum input is limited to the sizes listed in Table 1-2.

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

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.

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.

If command line recall and edit is disabled or no MPX.PRO file exists, the special control characters described in Table 1-1 are honored. Otherwise, TSM honors only the ASCII character codes defined in the MPX.PRO file, the break key, the wakeup character, and <ctrl>C. See the Command Line Recall and Edit chapter in this manual.

Reads directed to a TSM terminal in batch mode, such as $AS *lfc* TO LFC=UT, are redirected to read from SYC.

1.11.2 Writes

On a write (M.WRIT), the maximum output record 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 the TSM Screen Logic section in this chapter for more information. 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.

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 performs the open and TSM the close for UT.

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 a task in execution can test. 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

TSM automatically handles the following functions for an interactive task:

- 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
- returns the TSM prompt to the terminal when a task activated in the interactive environment aborts or exits
- break and wakeup interrupts to let the terminal user communicate with a task
- 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

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 line buffer and initializes the scanner to point to the first field.

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 R6 and R7.

R5 contains the character count of the last field found by the scanner. When the character count in R5 is zero and the delimiter in R4 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.

1.13.1.1 MPX-32 Delimiters

The MPX-32 delimiters recognized by the TSM scanner are:

- blank
- comma
- equal sign
- carriage return
- newline character (X'0A')
- left parenthesis
- right parenthesis
- semicolon
- exclamation point
- percent sign
- dollar sign
- apostrophe (X'27')

1.13.2 TSM Break Processor

When the TSM break processor (M.TBRKON/M_TBRKON) processes a pause or break from the terminal or calling task, it displays the following prompt:

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

Enter `C` to continue 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 online tasks and are called as a result of a hardware or software break. If a user transfer control word (TCW) is loaded in R2, it is printed along with the break message. See the MPX-32 Reference Manual, Volume I, Chapter 5 for additional information on the TCW.

1.14 JCL Directive Files

A directive file contains job control directives that control processing and execute tasks. Once activated, the file is referred to as a user's SYC file. A directive file can pass control to another directive file either permanently through chaining or temporarily through nesting. When TSM encounters an error in a directive file or a task activated from a file, it notifies the user and follows the same procedures as in normal interactive processing.

1.14.1 Executing Tasks from Directive Files

Tasks can be executed from a directive file using either the $EXECUTE directive, the $RUN directive, or simply the name of the task. If parameters are passed to an interactive task, the task accesses them through the M.TSCAN or M_TSCAN service.

While executing, a task can access records following the directive line within the directive file and use them as input. However, if the task encounters a directive preceded by a dollar sign ($), it interprets the line as a TSM directive and as an end-of-file for the task.

If the task exits, TSM continues processing the directive file, interpreting the next $*directive* as its own.

If the task aborts, TSM ignores subsequent records in the directive file until it encounters a $*directive.*

1.14.2 Chaining Directive Files

Directive files can be chained using the $SELECT directive within a directive file. $SELECT ends processing of the current directive file and transfers control to the selected directive file. Processing does not return to the original file. Any number of directive files can be chained together using $SELECT directives.

1.14.3 Nesting Directive Files

A directive file can be nested within another directive file using the $CALL directive. $CALL suspends processing of the current directive file and transfers control to another directive file or another instance of the current file. After the called file terminates, the suspended file resumes processing. $CALL allows users to nest up to eight levels of directive files and to pass and modify parameters among the files.

The following directives are global and affect all nested directive files: $ASSIGN, $ALLOCATE, $DISMOUNT, $LINESIZE, $MOUNT, $OPTION, $PAGESIZE, $RESETF, $SETF, $SHADOW, $SPACE, and $EXTDMPX.

The $CHANGE directive's DIRECTORY and PROJECT options are local and affect only the file executing the $CHANGE directive.

If TSM encounters a $SELECT directive within a called file, all nested directive files terminate processing and control transfers to the SELECTed directive file.

1.14.4 Processing Errors

When TSM processes a directive file, an error can occur either in the directive file or in a task or processor activated by the file. Errors within a directive file include supplying invalid parameters for a directive and unsuccessfully executing a directive.

1.14.4.1 Errors Within a Directive File

If an error occurs in a directive file, TSM displays an error message, the line where the error occurred, and the following instruction:

```
ENTER CORRECTION, 'END', 'CLEAR', OR <CR>
```

correction	specifies a correction line to be substituted for the line in error. *correction* must be a valid interactive command. The directive file resumes execution with the error flag reset.
END	immediately terminates the current directive file and returns to TSM if the current directive file is not nested. If the current file is nested, it terminates and control returns to the calling file with the error flag set. Parameters are not returned. All $ASSIGN, $ALLOCATE, $OPTION, $SHADOW and $EXTDMPX directives remain in effect.
CLEAR	terminates the directive file, clears all $ASSIGN, $ALLOCATE, $OPTION, $SHADOW and $EXTDMPX directives, and returns the user to TSM.
<CR>	ignores the line in error and resumes executing the file with the error flag reset.

1.14.4.2 Errors in Tasks Activated from a Directive File

If a task or processor activated from a directive file encounters an interactive error, TSM follows the same error and recovery procedure as in normal interactive processing. For a description of TSM breaks, see the Breaks section of this chapter. If the user selects ABORT in response to a break, TSM aborts the task and processes the next directive in the directive file.

1.15 Conditional Processing and Parameter Passing

The $IFP, $IFA, $GOBACK, $GOTO, $IFT, $IFF, $DEFNAME, $SET, $SETI, and *%label* directives provide the ability to:

- Substitute parameters. Up to 16 arguments 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 replacement by omitting an argument or by using an extra comma or other valid nonblank delimiter to indicate a missing argument.
- 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.
- Construct loops. In a directive file, users can define a processing loop, set a counter, then branch to and repeat the loop on command or when test conditions are met.

1.16 TSM/JCL Macro Looping

TSM directives can be used to construct controlled loops for processing within a directive file. Users construct a loop by labelling a set of directives, then using $GOBACK, $GOTO, or the GOTO/GOBACK option on $IFA, $IFP, $IFF, or $IFT to branch to the label. A counter can be set and incremented with the $SETI directive to control processing. The following are two examples of how a loop can be constructed:

Example 1:

```
$SETI %P1 = 10
$DEFN LOOP1
      .
      .
      .
$SETI %P1 = %P1 - 1
$IFT %P1 GT 0 GOBACK LOOP1
```

Example 2:

```
$SETI %P1 = 10
$DEFN LOOP1
$IFT %P1 EQ 0 GOTO ENDLOOP1
      .
      .
      .
$SETI %P1 = %P1 - 1
$GOBACK LOOP1
$DEFN ENDLOOP1
```

1.17 Parameter Replacement by Macro Directive Files

An inherent feature of the SYC file is the parameter replacement facility. A directive file that requires parameter replacement is called a macro. Up to 16 parameters are defined when a macro file is created. When the macro file is invoked, up to 16 arguments can be specified. These arguments are replaced in the text of the macro file as it is read from the SYC. Parameters within the text are identified by a percent sign (%) preceding the parameter name. The correlation between the called arguments and the parameters is established by the $DEFM directive. If parameter replacement is required, $DEFM must be the first record within the directive file. See the $DEFM directive description.

1.18 Concatenating a Value to a User-Supplied Argument

A directive file that uses parameter replacement 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 argument (*%par*) supplied by the user of the directive file.

1.19 Spooled Input Control by $SELECT*x*

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 disk resources, the input spoolers J.SSIN1 and J.SSIN2 copy data into a single SYC file. For one or more binary devices or disks, J.SSIN1 and J.SSIN2 also copy data into a single SGO file. The SYC and SGO files are temporary, unnamed spool files that are deleted after job processing is complete. Data flow of a batch job is shown in Figure 1-2.

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 Disk File (M.CDJS) system service, or the Batch Job Entry (M.BATCH) 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 $SELECT*x* directives that can be included anywhere in the job following the $JOB directive. When a $SELECT*x* directive is encountered during input spooling, the $SELECT*x* directive is not written to the SYC file, but is replaced by data from the device or file specified in the directive. However, $SELECT*x* directives that contain errors are written to the SYC file.

Each $SELECT*x* directive that specifies a valid device or file establishes a new alternate level. A maximum of three alternate levels is provided. The conditions for which an alternate level is reset to a previous level (alternate or primary) are summarized in Table 1-3.

A device or file can be specified on a $SELECT*x* 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 (e.g. the specification MT1000 is not identical to MT10).

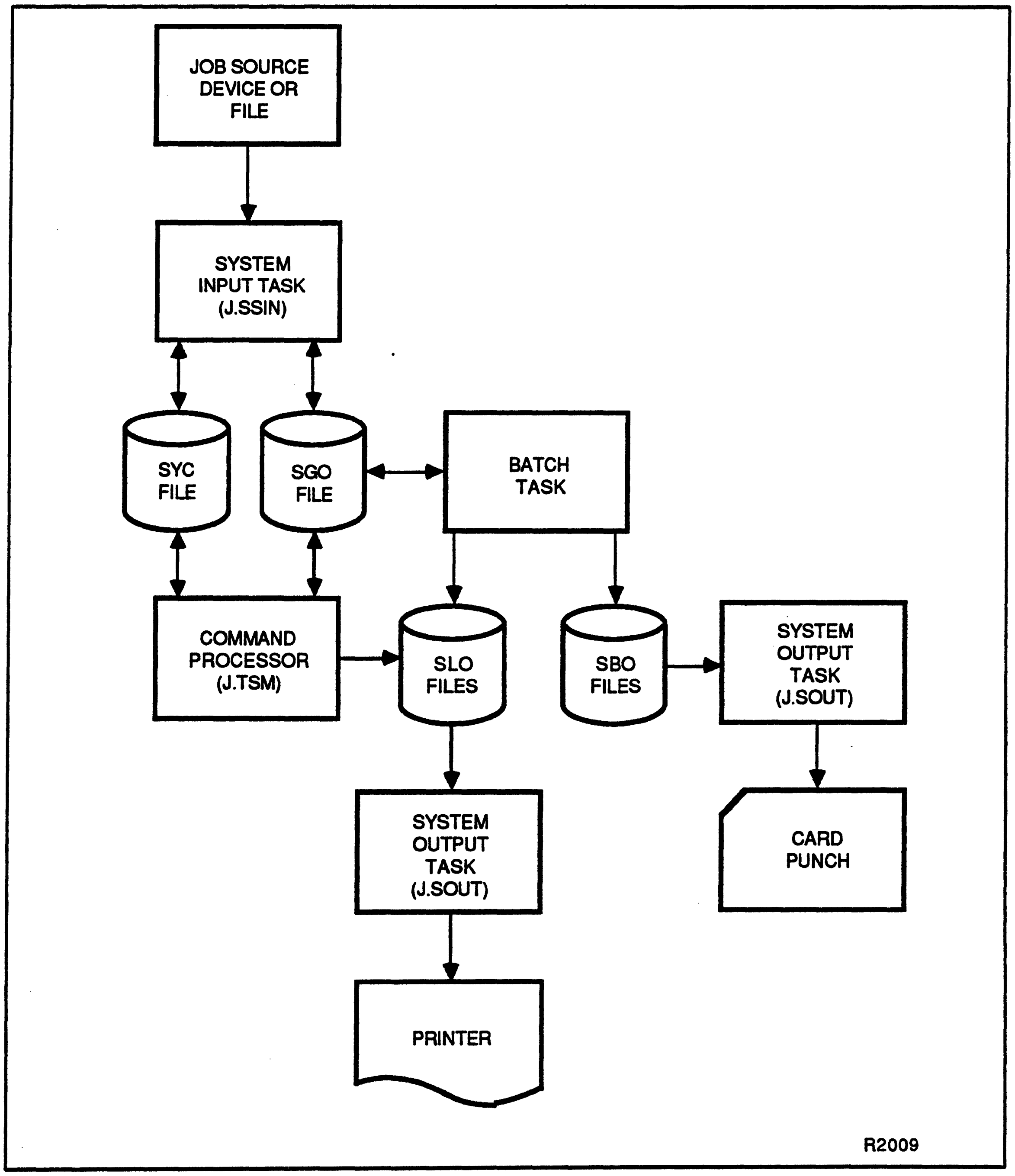

JOB SOURCE DEVICE OR FILE
SYSTEM INPUT TASK (J.SSIN)
SYC FILE
SGO FILE
BATCH TASK
COMMAND PROCESSOR (J.TSM)
SLO FILES
SBO FILES
SYSTEM OUTPUT TASK (J.SOUT)
SYSTEM OUTPUT TASK (J.SOUT)
CARD PUNCH
PRINTER
R2009

Figure 1-2
Data Flow for a Job

Table 1-3
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

1.20 Syntax of TSM Directives

The general syntax of TSM directives is:

$directive [*arguments...*]

- Directive names can be abbreviated to the letters in bold print in the directive descriptions in the following section.
- Valid delimiters between arguments are blanks, commas, equal signs, semicolons, or parentheses.
- Where arguments have an implicit order, insert an extra comma or other non-blank delimiter to mark the absence of an argument. For example:

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

- In interactive mode, the dollar sign preceding TSM directives can be omitted. In batch mode, the dollar sign must be specified.

1.21 $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**=*name*]
name specifies the owner name associated with the files to be displayed

[,**PROJ**=*proj*]
proj specifies the project group name associated with the files to be displayed

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

[,**ORIG**={**TSM.***nnnn*|**BATCH**}]
specifies the mode of operation in which the files to be displayed were used. *nnnn* specifies the command processor device number.

Examples

The following example displays statistics on all jobs with owner name `USER1` run on June 16, 1980 on any command processor device *20xx.*

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

The following example displays statistics on all jobs run under project number `082950`.

```
$ACCOUNT,PRO=082950
```

1.22 $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` displays and the next statement is processed.

Examples

The following example demonstrates activation of the job called `JOBFILE` and the message that is returned to the user's terminal showing the assigned task number.

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

1.23 $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)

Examples

With the following code, using the $ALLOCATE directive allows for increasing the size of BUFFERX and having a label attached to it.

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

1.24 $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, $RUN, $ACTIVATE, or a TSM activation without a preceding directive verb.

Syntax

$ASSIGN *lfc* **TO** *resource* [**ACC**ESS=([**R**EAD] [**W**RITE] [**M**ODIFY] [**U**PDATE] [**A**PPEND])]
[**BBU**F=*buffers*] [**BLO**CKED={**Y** | **N**}] [**DEN**SITY={**800** | **1600** | **6250** | **N** | **P** | **G**}]
[**ID**=*id*] [**MUL**TIVOL=*number*] [**PR**INT | **PUN**CH] [**SHA**RED={**Y** | **N**}]
[**SIZ**E=*blocks*]

Note: For syntax when assigning a logical file code to an ANSI labeled tape, see the Syntax for ANSI Labeled Tapes section that follows.

lfc is a 1- to 3-character logical file code to be used by a task for resource assignment

resource Supply one of the following:

SBO	specifies system binary output
SLO	specifies system listed output
SYC	specifies system control file
SGO	specifies system general object file
pathname	specifies the pathname of the resource
RID=*resid*	*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
TEMP[=(*volname*)]	*volname* is the volume name on which temporary space is to be allocated. The default is any volume.
DEV=*devmnc*	*devmnc* is the device mnemonic of a configured peripheral device. See Appendix A.
LFC=*lfc*	*lfc* is a 1- to 3-character logical file code used in the task.

[**ACC**ESS=([**R**EAD] [**W**RITE] [**M**ODIFY] [**U**PDATE] [**A**PPEND])]
specifies the type of access for *resource*. This must be a subset of access allowed at resource creation. The default is the access specified at resource creation. This option is only valid when assigning a resource using the *pathname*, RID, TEMP, or DEV parameters.

[**BBU**F=*buffers*]
buffers is the number of 192W blocking buffers if using a large blocking buffer. The default is one.

[BLOCKED=**{Y|N}]**

If Y, the resource is explicitly blocked. If N, the resource is explicitly unblocked. The default is blocked. This option is only valid when assigning a resource using the @ANSITAPE, *pathname*, RID, TEMP, or DEV parameters.

[DENSITY=**{800|1600|6250|N|P|G}]**

specifies density of high speed extended I/O (XIO) tape. This option is only valid when assigning a resource using the DEV parameter.

800 or **N**	specifies 800 bpi nonreturn to zero inverted (NRZI)
1600 or **P**	specifies 1600 bpi phase encoded (PE)
6250 or **G**	specifies 6250 bpi group coded recording (GCR) (default)

[ID=*id***]** *id* is an identifier for an unformatted medium. The default is SCRA (scratch). This option is only valid when assigning a resource using the DEV parameter.

[MULTIVOL=*number***]**

number is a volume number for a multivolume tape. The default is zero (not multivolume). This option is only valid when assigning a resource using the DEV parameter.

[PRINT**|PUN**CH**]**

indicates the file is to be printed (PRINT) or punched (PUNCH) after deassignment. This option is only valid when assigning a resource using the *pathname*, RID, or TEMP parameters.

[SHARED=**{Y|N}]**

If Y, the resource is explicitly shared. If N, the resource is exclusive. The default is implicitly shared. This option is only valid when assigning a resource using the *pathname*, RID, TEMP, or DEV parameters.

[SIZE=*blocks***]**

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

1.24.1 Syntax For ANSI Labeled Tapes

$ASSIGN *lfc* **TO @ANSITAPE(***lvid***)***file* [**ACC**ESS=([**R**EAD | **W**RITE | **U**PDATE | **A**PPEND])]
[**BLO**CKED={**Y** | **N**}] [**BSI**ZE=*bsize*] [**EXP**IRE={*date* | +*days*}]
[**FOR**MAT={**F** | **D** | **S**}] [**GENE**RATION=*gennum*] [**GENV**ERSION=*genvnum*]
[**PRO**TECT={**0** | **A-Z**}] [**REC**LENGTH=*recsize*]

lfc is a 1- to 3-character logical file code to be used by a task for resource assignment

@ANSITAPE(*lvid***)***file*
specifies an ANSI labeled tape

(*lvid*) is the 1- to 6-character logical volume identifier previously mounted by the ANSI labeled tape AMOUNT utility. Only one LFC can be assigned to an *lvid*. Before further assignments can be made, the M.DASN service must be used.

file is a 1- to 17-character file identifier

[**ACC**ESS=([**R**EAD | **W**RITE | **U**PDATE | **A**PPEND])]
specifies the type of access for an ANSI labeled tape. Access choices are:

Value	Description
R	read existing file (default)
W	create file at first unexpired file on tape
U	overwrite existing file with a new file of the same name
A	create file at end of tape

These access modes are mutually exclusive.

[**BLO**CKED={**Y** | **N**}]
If Y, the ANSI labeled tape is explicitly blocked. If N, it is explicitly unblocked. The default is blocked.

[**BSI**ZE=*bsize*]
bsize is read from the ANSI labeled tape on read access. For other types of access, *bsize* 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. The default is 2048 bytes.

[**EXP**IRE={*date* | +*days*}]

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. 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. The default is +30 days. **Caution:** If the number of days is not preceded by a plus (+), the number entered can be read as the date.

[**FOR**MAT={**F** | **D** | **S**}]

specifies the ANSI labeled tape record format. 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

[**GENE**RATION=*gennum*]

gennum is the 1- to 4-decimal digit ANSI labeled tape file generation number. On input (read access), this number must match the generation number of the ANSI tape file being assigned. On output (write, update, or append access), this value becomes the generation number of the new ANSI tape file. The default is one on output; no check on input.

[**GENV**ERSION=*genvnum*]

genvnum is the 1- or 2-decimal digit ANSI labeled tape file generation version number. On input (read access), *genvnum* must match that of the ANSI tape file being assigned. On output (write, update or append access), *genvnum* becomes the generation version number of the new ANSI tape file. The default is zero on output; no check on input.

[**PRO**TECT={**0** | **A-Z**}]

specifies protection for new ANSI labeled tape files. Zero specifies owner only access. A through Z are reserved by the ANSI specification for installation-specific protection. MPX-32 treats A through 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 owner name written by J.LABEL can be overridden. The default is no protection.

[**REC**LENGTH=*recsize*]

recsize is read from the ANSI labeled tape header on read access. For other types of access, *recsize* 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* (see the BSIZE=*bsize* option). The default is 80.

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.

Examples

The following example assigns logical file code `LIB` to the file `LIBRARY` in the current working directory. The file is specified as unblocked and explicitly shared.

```
$AS LIB TO LIBRARY BLO=N SHA=Y
```

The following example assigns system listed output to the terminal.

```
$AS SLO TO LFC=UT
```

1.25 $ASSIGN1 Directive

The $ASSIGN1 directive assigns permanent files for LFCs 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 1- to 3-character logical file code used in the task
filename	is a 1- to 8-character name of a file (in the current working directory or the system directory) to assign to the LFC Enter any of the optional parameters following the file name, as shown in the syntax statement. Options must be separated by a comma. If an option is omitted, a comma must be inserted in its position.
password	is ignored
U	specifies the file is 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.

Examples

The following example assigns the unblocked files `LIBRARY` and `DIRECTORY` to logical file codes `LIB` and `DIR` respectively.

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

1.26 $ASSIGN2 Directive

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

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.
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.
SYC	TSM automatically assigns SYC to UT if a directive file does not exist. Tasks should use $ASSIGN4 to equate any other LFCs to UT.
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 1- to 3-character logical file code used in the task
SBO,*cards*	specifies 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,*printlines*	specifies 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, separate each LFC assignment with at least one blank space.

Examples

The following example assigns logical file code `CAR` to the temporary punched output file, specifying a file size to accommodate `1000` cards.

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

The following example assigns logical file code `OUT` to the temporary file for listed output, with a size to accommodate `50` printed lines. Logical file code `OT2` is assigned to the system general object file.

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

1.27 $ASSIGN3 Directive

The $ASSIGN3 directive supplies device assignments for LFCs 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 1- to 3-character logical file code used in the task
devmnc	is the device mnemonic of a configured peripheral device (see Appendix A)
[*blocks* | *reel* [*,vol*]]	specifies information about the resource being assigned

blocks	is the number of disk blocks (192 words) to be allocated for this file
reel	specifies a 1- to 4-character identifier for the reel. The default is SCRA (scratch).
vol	is the volume number for a multivolume tape. The default is 0 (not multivolume).

[,U]	specifies the tape or disk is unblocked. 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 $ASSIGN3 directive, separate each LFC assignment with at least one blank space.

Examples

The following example assigns LFC `IN` to the magnetic tape residing on channel `10` subaddress `00` identified as `SRCE`. The tape is unblocked. A second assignment of LFC `OT` to the paper tape device is also made.

```
$A3 IN=M91000,SRCE,,U OT=PT
```

1.28 $ASSIGN4 Directive

The $ASSIGN4 directive associates one or more LFCs 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.

An LFC assigned to LFC 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 *newlfc=currlfc* [*newlfc=currlfc*] ...

newlfc=currlfc
is a pair of logical file codes. Any number of LFC to LFC assignments can be specified. In the interactive mode, the user's terminal is preassigned to UT.

newlfc	is the new assignment
currlfc	is the LFC already associated with a file or device in any previous $ASSIGN directive (including $ASSIGN4).

Examples

The following example demonstrates that the $ASSIGN4 directive can be used to assign logical file code `IN2` to logical file code `IN`. `IN` was previously assigned to the file `PERMDISC` using an $ASSIGN1 directive statement.

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

1.29 $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 the Batch Stream Memory Pool Interaction section of this chapter 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.

When messages or signals that specify an owner name are issued by a directive file being run via $BATCH, they go to all terminals logged on with the owner name to which the message is sent.

$BATCH is similar to the OPCOM BATCH directive except only $BATCH can pass parameters.

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

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

Syntax

$BATCH *pathname* [*arg*] ...

pathname	is the pathname of a permanent file containing TSM directives
[*arg*]	is a parameter to pass for the directive file. Up to 16 arguments can be passed.

Response

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

jobnumber ownername `$JOB` *jobname*

1.30 $CALL Directive

The $CALL directive suspends processing of the current directive file and transfers control to another directive file or another instance of the current file. $CALL allows users to nest up to 8 levels of directive files and to pass and modify parameters among the files. After a called file completes execution or terminates with an error, control returns to the calling file.

A file can pass up to 16 arguments to the file that it calls. Each argument passed is matched positionally to a parameter defined on the called file's $DEFM line. Modifying a passed parameter within a called file also modifies the corresponding parameter in the calling file.

The following directives are global and affect all nested directive files: $ASSIGN, $ALLOCATE, $DISMOUNT, $LINESIZE, $MOUNT, $OPTION, $PAGESIZE, $RESETF, $SETF, $SHADOW, $SPACE, and $EXTDMPX. The $CHANGE directive's DIRECTORY and PROJECT options are local and affect only the file executing the $CHANGE directive.

If TSM encounters a $SELECT directive within a called file, all nested directive files terminate processing and control transfers to the SELECTed file.

$CALL is valid only from interactive directive files.

Syntax

$CALL *pathname* [*arg*] ...

pathname	is the pathname of a permanent directive file. Can be another instance of the current file.
[*arg*]	is an argument (parameter or string) to pass to the called directive file. Up to 16 arguments can be passed.

Errors

An error in a called directive file causes $CALL to display an error message, the line where the error occurred, and the following instruction:

```
ENTER CORRECTION, 'END', 'CLEAR', OR <CR>
```

correction	substitutes the entered correction line for the line in error. TSM then resets the error flag and continues executing the called directive file. *correction* must be a valid interactive directive (macro directives are not valid). Directive processing continues with the next directive in the directive file.
`END`	immediately terminates the current directive file, sets the error flag, and returns to the calling file. Parameters are not returned to the calling file.
`CLEAR`	terminates all nested directive files, clears all $ASSIGN, $ALLOCATE, $OPTION, $SHADOW, and $EXTDMPX directives, and returns the user to the interactive mode
`<CR>`	pressing the return key ignores the line in error, resets the error flag, and resumes executing the called file

An attempt to execute a $CALL directive from the eighth level of nesting generates the following error message that displays to the terminal:

```
NESTING LIMIT HAS BEEN EXCEEDED
```

Examples

The following example calls file `TECHPRO` and passes it the parameters `%P1`, `%P2`, and `%P3`.

```
$CALL TECHPRO %P1 %P2 %P3
```

1.31 $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 on hold.

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

The project group name is used for accounting and to establish access restrictions for resources. When a project group name 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 name.

The working directory specifies 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 may be issued in one directive as long as a *pathname* and a *projname* are specified.

Issuing the $CHANGE directive without specifying a project for PROJECT or a pathname for DIRECTORY will re-establish your logon project or directory (this must be done individually).

Syntax

$CHANGE [**DIR**ECTORY[=*pathname*]] [**PRO**JECT[=*projname* [,*key*]]]

[**DIR**ECTORY[=*pathname*]]
pathname specifies the pathname of the new default directory.

Note: Avoid parentheses in pathnames, especially if the directory pathname is followed by the project key.

[**PRO**JECT[=*projname* [,*key*]]]
specifies a new project group name

projname	is the name of the new default project group name
key	is the 1- to 8-character key associated with the project name. If not specified, the default is no key.

Examples

The following example changes the current working directory to `NEWDIR` on volume `NEWVOL1`.

```
$CHANGE DIR=@NEWVOL1 NEWDIR
```

The following example changes the default project group to `NEWPROJ`.

```
$CHAN PRO=NEWPROJ
```

The following example changes the default project group name to `061643` and current working directory to `ATLAS` on the current working volume.

```
    $CHANGE PRO=061643 DIR=^ATLAS
```

The following example re-establishes your logon directory.

```
    $CHAN DIR
```

The following example re-establishes your logon project.

```
    $CHAN PROJ
```

1.32 $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 user.

Syntax

$CLEAR

1.33 $CONTINUE Directive

The $CONTINUE directive resumes execution of a task that was placed on hold 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

1.34 $CREATE Directive

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

Syntax

$CREATE *pathname*

pathname	is 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
```

Examples

The following example creates file `INDEX` on the current working volume and directory.

```
$CREA INDEX
```

The following example creates file `INDEX` on volume `ATLAS04` in directory `SOURCE`.

```
$CREA @ATLAS04(SOURCE)INDEX
```

1.35 $DEBUG Directive

The $DEBUG directive debugs a task (load module or executable image). The appropriate interactive debugger 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 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.

1.36 $DEFM Directive

The $DEFM directive defines parameters that TSM uses when executing a directive file. A parameter can be a substitution string for part of a value or a value supplied as a variable for a directive. Up to 16 parameters can be defined for a file and passed arguments at run time.

An argument passed to a parameter can be up to 72 characters in length. Its actual length is limited by the length of other arguments. The combined number of characters of all arguments in a file cannot exceed 256. Arguments containing embedded blanks, commas, or other delimiters must be enclosed within single quote marks.

The default value of a parameter is the 1- to 16-character string that defines the parameter in the $DEFM directive. Only parameters defined on the $DEFM line can receive an argument at run time. Parameters created dynamically within a file cannot receive input values.

When accessing a directive file at run time, any arguments passed in with the directive file name must be listed in the order they are defined in the file's $DEFM line. To mark the absence of an argument in the run command, insert an extra comma or other non- blank delimiter (equal sign, semicolon, or parenthesis).

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

Syntax

$DEFM [*par*][,*par*] ...

[*par*]	is a 1- to 16-character alphanumeric string that defines a parameter. Up to 16 parameters can be defined for a file.

Examples

The following directive file makes assignments and selects standard options for assembly. `SI` is the parameter indicating a base file name. If the user omits the first argument, `SI` is the name of the input file for the assembly. The macro selects `1` as the default option by the second parameter of the $DEFM directive.

ASM directive file

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

At the `TSM>` prompt, the user enters:

```
$SELECT ASM SRCE
```

Expansion of the directive file as executed is:

```
$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 ASM file to be identified as a directive file and not a load module.

1.37 $DEFNAME and %*label* Directives

The $DEFNAME and *%label* directives establish a label to branch to for conditional processing.

Following $DEFNAME or *%label*, the user supplies the sequence of directives to process when a conditional branch occurs. A branch is the result of an $IF*x* directive that references the label.

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

Syntax

$ {DEFNAME | %} *label*

label	is a 1- to 8-character alphanumeric string that establishes a label to branch to.

Examples

The following example demonstrates two ways to establish the label `CKDIR`.

```
$DEFN CKDIR
```

or

```
%CKDIR
```

1.38 $DELETE Directive

The $DELETE directive deletes a permanent file. This directive can be used when a task is on hold. If the delete attempt fails, directive file error processing or batch job termination occurs.

Syntax

$DELETE *pathname*

pathname	is the pathname of the file to delete

Examples

The following example deletes the file `INDEX` on the current working volume and directory.

```
$DELE INDEX
```

The following example deletes the file `INDEX` on volume `ATLAS04` in directory `SOURCE`.

```
$DELE @ATLAS04(SOURCE)INDEX
```

1.39 $DISABLE Directive

The $DISABLE directive disables logons to one or all defined terminals (except the system console). The system console cannot be disabled with this directive. The $DISABLE directive is available only to the system administrator. If any other user attempts to use this directive, the message `*UNAUTHORIZED USER` appears on the user's terminal.

If this directive is used to disable a terminal that is in use, the terminal will be serviced by TSM until its user logs off. If the directive is used to disable a terminal that is not in use, TSM service is discontinued immediately.

When TSM service to a terminal is discontinued, the terminal's screen clears and the following message appears:

```
SERVICE DISABLED
```

The terminal will remain offline until it is enabled using the $ENABLE directive described in this chapter.

Syntax

$DISABLE [*ccaa*]

ccaa	is the channel and subaddress of the terminal to be disabled. *cc* is the physical channel number in hexadecimal. *aa* is the physical channel subaddress in hexadecimal.

If *ccaa* is not specified, all terminals except the system console are disabled.

Examples

The following example disables logons to device `7EA0`.

```
$DISA 7EA0
```

1.40 $DISMOUNT Directive

The $DISMOUNT directive requests the logical dismount of a nonpublic volume, and optionally, the physical dismount of a user volume (public or nonpublic). The $DISMOUNT directive can be used to successfully remove a volume from its allocated device when the volume is not present on the actual device.

For logical dismounts, the nonpublic volume is dismounted from the current TSM environment. If the volume is not mounted to any other TSM environments, it is also logically dismounted from J.TSM. The VAT entry for the volume is cleared in J.TSM and the use count is decremented in the volume's MVT entry.

For physical dismounts, TSM first attempts a logical dismount of the volume. Only nonpublic volumes require logical dismount. If other users are currently active on the volume, the physical dismount is postponed and new physical and TSM logical mount requests for the volume are denied.

When there are no users on the volume, the physical dismount is performed. A prompt issues to the system console when the dismount is complete to inform the operator that the volume can be physically removed from the drive. The operator must respond to the prompt. When the volume is physically dismounted, the last user to request its physical dismount receives a message stating the date and time of dismount. No operator interaction occurs if the NOMSG option is specified or if the system-wide SNOP or OPCOM SIMM options are enabled.

If the SYSGEN CMPMM option was specified at system initialization, public volumes cannot be dismounted from the running system. If the SYSGEN CMIMM option was specified, nonpublic volumes:

- can be physically dismounted without supplying a device specification
- can be implicitly physically dismounted as the result of their last user logging off TSM or exiting a batch job
- cannot be physically dismounted by TSM if they were mounted with the OPCOM MOUNT command

Messages generated by the $DISMOUNT directive in interactive mode go to all terminals logged on with the same owner name as the owner who issues the $DISMOUNT command.

A public volume dismount request is valid only from the system administrator (SA).

Syntax

$DISMOUNT *volname* [**FROM** *devmnc*] [**OPT**ION=[**NOM**SG] [**PUB**LIC]]

volname is the volume to be dismounted

[**FROM** *devmnc*]
devmnc specifies the device where the volume is mounted (see Appendix A). Physical dismount requests must specify a device.

[**OPT**ION=[**NOM**SG] [**PUB**LIC]]

NOMSG	if specified, operator response is not required at the system console. **Note:** This option is ignored if the NOMSG option is specified in the MOUNT directive.
PUBLIC	is specified if the volume to be physically dismounted is a public volume

Response

All messages display to the terminal in interactive mode or to SLO in batch mode.

1.40.1 Logical Dismount

When a logical dismount of a volume completes, the following message displays:

```
LOGICAL DISMOUNT OF VOLUME volname COMPLETE
```

1.40.2 Physical Dismount

When a physical dismount of a public volume is requested, the system console displays the following warning message:

```
WARNING - ATTEMPTING DISMOUNT OF A PUBLIC VOLUME
```

If there are users active on a requested volume (nonpublic or public), the dismount is postponed and a message displays to the user terminal:

```
PHYSICAL DISMOUNT OF VOLUME volname PENDING.
USERS CURRENTLY ACTIVE ON VOLUME.
```

When there are no users active on the volume, the physical dismount completes. The following message displays to the system console to prompt the operator to physically remove the dismounted volume from the drive, if removable:

```
CONFIRM PHYSICAL DISMOUNT OF VOLUME volname
FROM device
REPLY R TO RESUME:
```

The operator must respond with a character to confirm the dismount. Confirmation is required in order for further mount or dismount activity to continue in the system.

When the physical dismount completes, a message displays to the terminal and the system console:

```
PHYSICAL DISMOUNT OF VOLUME volname
FROM device COMPLETE.
```

Errors

If an error occurs and the volume cannot be dismounted, the following message displays to the terminal:

```
UNABLE TO DISMOUNT VOLUME volname REASON RMxx
```

Examples

The following example physically dismounts the nonpublic volume `TB00` from its device when the volume's use count reaches zero.

```
$DISM TB00 FROM DM0800
```

The following example logically dismounts nonpublic volume `TB00` from the local TSM environment. If this context is the last logical TSM user, then J.TSM is no longer a user of the volume and the volume's use count is decremented.

```
$DISM TB00
```

1.41 $ENABLE Directive

The $ENABLE directive enables logons to one or all defined terminals. This directive does not affect the system console. The $ENABLE directive is available only to the system administrator. If any other user attempts to use this directive, the message `*UNAUTHORIZED USER` appears on the user's terminal.

When the $ENABLE directive is used to enable an offline terminal, the terminal's screen clears and the following message appears:

```
RING IN FOR SERVICE
```

Syntax

$ENABLE [*ccaa*]

ccaa	is the channel and subaddress of the terminal to be enabled. *cc* is the physical channel number in hexadecimal. *aa* is the physical channel subaddress in hexadecimal.

If *ccaa* is not specified, all terminals (except the system console) are enabled.

Examples

The following example enables logons to device `7EA0`.

```
$ENAB 7EA0
```

1.42 $END Directive

The $END directive terminates a directive file without clearing $SHADOW, $ASSIGN, $OPTION, $ALLOCATE, or $EXTDMPX directives. If used within a nested directive file, arguments are not returned to the caller and the error flag is set in the caller's directive file.

This directive is valid in interactive mode or interactive directive files.

Syntax

$END

1.43 $ENDM Directive

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

Syntax

$ENDM

1.44 $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.

$EOJ is treated like an $ENDM directive if used in a nested directive file.

Syntax

$EOJ

1.45 $ERR Directive

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

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 4-character abort code

Examples

The following example illustrates displaying the description for abort code `AS02`.

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

1.46 $EXECUTE Directive

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

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.

Examples

The following example activates the Text Editor utility.

```
$EXEC EDIT
```

The following example activates the load module or executable image file `TEST1`, located on the system volume in the system directory.

```
$EXEC TEST1
```

1.47 $EXIT Directive

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

Syntax

$EXIT

1.48 $EXTDMPX Directive

The $EXTDMPX directive can be used at runtime to determine the logical starting address of the TSA and extended MPX-32 (if configured). When extended MPX-32 is not configured, this directive applies only to the TSA.

The NOTSA or TSA option is ignored when the load module has been cataloged in the compatible mode or using the TSA keyword in the Cataloger EXTDMPX directive. The NOTSA or TSA option is effective only when the load module has been cataloged using the SYSTSA keyword in the Catalog EXTDMPX directive. When this requirement has been met, a TSM or M.PTSK request will override the SYSGEN request.

This directive is ignored if the task is shared.

For more information on moving the TSA, refer to the Extended TSA section in Chapter 3 of the MPX-32 Reference Manual, Volume I.

Syntax

$EXTDMPX {*logmapbl* | **MAX**ADDR | **MIN**ADDR} [,**NOT**SA | ,**TSA**]

logmapbl	is the decimal logical map block number between 64 and 2047 that specifies a starting map block in the task's logical address space where the TSA (optionally) and extended MPX-32 (if configured) are positioned. The NOTSA/TSA keyword controls positioning of the TSA.
MAXADDR	positions the TSA (optionally) and extended MPX-32 (if configured) at the top of the task's logical memory. The NOTSA/TSA keyword controls positioning of the TSA.
MINADDR	positions the TSA and extended MPX-32 (if configured) at the bottom of the task's logical memory above MPX-32 (when mapped in), and below the task's DSECT. The TSA keyword defaults to NOTSA for MINADDR.
NOTSA	directs the logical position of the TSA to be above MPX-32 (when mapped in) and below extended MPX-32 (if configured and at MINADDR), and below the task's DSECT.
TSA	directs the repositioning of the TSA in accordance with the MAXADDR, or *logmapbl* specification used. For MAXADDR, the TSA is located at the top of the task's logical memory followed by extended MPX-32 (if configured). For *logmapbl*, the TSA logically starts at *logmapbl* followed by extended MPX-32 (if configured).

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, or `NOTSA` or `TSA` is specified without the address parameter, an `ILLEGAL BLANK FIELD` message is displayed on the user's terminal. If `NOTSA` or `TSA` keywords are incorrectly spelled the entire directive is considered incorrect, and the error message `*ILLEGAL ENTRY` is generated.

Examples

The following example specifies logical map block 65 as the starting address of extended MPX-32 for the current task.

```
$EXTDMPX 65
```

The following example specifies that extended MPX-32 is mapped into the task's logical address space at the end of the TSA.

```
$EXTD MINADDR
```

The following example specifies that extended MPX-32 is mapped at the task's highest available logical address space.

```
$EXTDMPX MAXA
```

The following example positions extended MPX-32 only at map block 64.

```
$EXTD 64,NOTSA
```

The following example positions the TSA followed by extended MPX-32 starting at map block 64.

```
$EXTD 64,TSA
```

The following example positions the TSA followed by extended MPX-32 at MINADDR.

```
$EXTD MINADDR,TSA
```

The following example positions the TSA followed by extended MPX-32 at MAXADDR.

```
$EXTDMPX MAXA,TSA
```

1.49 $GOBACK Directive

The $GOBACK directive branches backward to a specified label in a directive file and continues processing. The directive can be used with $SET or $SETI to construct loops within a directive file.

$GOBACK is valid only from interactive and batch directive files.

Syntax

$GOBACK *label*

label	is a 1- to 8-character alphanumeric string defined with $DEFNAME or *%label* before executing the $GOBACK directive. The label marks a point to go to in a directive file to continue processing.

Errors

If $GOBACK reaches the beginning of file (BOF) without locating the specified label, it generates the following message and terminates processing of the file:

```
LABEL label NOT FOUND
```

label	is the label searched for

Examples

In the following example, `$GOBACK` is used with `$SETI` to construct a processing loop.

```
$SETI %P1 = -10
$DEFN LOOP
$IFT %P1 GE 0 GOTO ENDLOOP
    .
    .
    .
$SETI %P1 = %P1 + 1
$GOBACK LOOP
$DEFN ENDLOOP
```

1.50 $GOTO Directive

The $GOTO directive branches forward to a specified label in a directive file and continues processing. If $GOTO reaches the end of the file without finding the specified label, TSM terminates processing of the directive file.

$GOTO is valid only from interactive and batch directive files.

Syntax

$GOTO *label*

label	is a 1- to 8-character alphanumeric string defined by a $DEFNAME or *%label* directive. The label marks a point to go to in the file to continue processing.

Examples

The following example demonstrates the $GOTO directive used to branch forward to the code labeled `START`.

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

1.51 $IFA and $IFP Directives

The $IFA/$IFP directives branch to a label in a directive file if a specified argument is absent ($IFA) or present ($IFP) at run time.

The GOTO and GOBACK options enable TSM to branch forward or backward to the nearest occurrence of a label after testing for the argument's presence or absence. If GOTO or GOBACK is not specified, TSM branches forward.

$IFA and $IFP are valid only from interactive and batch directive files.

Syntax

{$IFA|$IFP} *%par* [**GOTO**|**GOBA**CK] *label*

%par is a parameter defined by the $DEFM directive or created dynamically by the $SET or $SETI directives.

[**GOTO**|**GOBA**CK]
specifies direction

GOTO	branches forward to the next occurrence of *label* (default)
GOBACK	branches backward to the previous occurrence of *label*

label is a 1- to 8-character alphanumeric string defined by the $DEFNAME or *%label* directive. The label marks a point to go to in the directive file to continue processing.

Errors

Directive file processing terminates after an error is generated. Error messages display on the terminal in interactive mode or to SLO in batch mode. $IFA or $IFP can generate the following error messages.

If the specified parameter does not exist, the directive generates the following message and displays the line where the error occurred:

```
INVALID COMMAND SYNTAX, THE FOLLOWING RECORD IGNORED:
```

If GOBACK is in effect and TSM reaches the beginning of file without locating the specified label, the directive generates the following message:

`LABEL` *label* `NOT FOUND`

label is the label searched for

If GOTO is in effect and TSM reaches the end of file without locating the specified label, TSM terminates processing of the directive file.

Examples

In this example, `OP` is the parameter for selecting an Assembler option. It is the fourth argument 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
```

At the `TSM>` prompt, the user enters the following:

```
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` argument 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`."

1.52 $IFF Directive

The $IFF directive branches to a label if a specified condition is false in a directive file. If a condition is true, TSM processes the next directive in the file. The following conditions can be tested:

- comparison results
- flag setting
- task abort
- file existence
- error occurrence from nested directive files

Valid comparisons can be made between integers and type integer parameters, strings and type string parameters, or two parameters. If two parameters are compared, both must exist and be of the same type.

GOTO and GOBACK options enable TSM to branch forward or backward to the nearest occurrence of a label after testing for a false condition. If GOTO or GOBACK is not specified, TSM branches forward.

$IFF is valid only from interactive and batch directive files.

Syntax

$IFF *condition* [**GOTO** | **GOBA**CK] *label*

condition specifies a condition to be verified as false. Supply one of the following:

ABORT indicates branch if the preceding task did not abort

ERROR indicates branch if the called directive file did not generate an error

FILE *filename*
filename is a 1- to 8-character name of a user or system file. If the file is not in the system directory or the current working directory, TSM branches.

flagno is a numeric flag number from 1 to 31. Only one flag number can be specified.

PATH *pathname*
pathname specifies the pathname of a file. If the file does not exist, TSM branches.

%par oper {*string* | *int* | *%par*}
or
{*string* | *int* | *%par*} *oper %par*

%par is a parameter defined by the $DEFM directive, or created dynamically by the $SET or $SETI directive, or defined by the $DEFM directive and converted to an integer with the $SETI directive. *%par*'s type must match the type being compared to; i.e., an integer compared to an integer parameter, a string compared to a string parameter, or two parameters of the same type.

oper is one of the following valid relational operators:

EQ	equal	NE	not equal
GT	greater than	GE	greater than or equal
LT	less than	LE	less than or equal

string is a 1- to 72-character alphanumeric string. Strings with embedded delimiters must be enclosed in single quotes. Strings can also result from concatenation of multiple parameters or strings. A concatenated string cannot exceed 72 characters.

Note: Do not use a keyword (ABORT, FILE, PATH, or ERROR) or its valid abbreviation for *string*.

int is any 1- to 7-character integer, optionally preceded by a plus (+) or minus (-) sign. If a sign is specified, it must be the first character of the integer and cannot be delimited by blanks.

[**GOTO** | **GOBA**CK]
specifies direction

GOTO branches forward to the next occurrence of *label* (default)

GOBACK branches backward to the previous occurrence of *label*

label is a 1- to 8-character alphanumeric string defined by a $DEFNAME or *%label* directive which marks a point to go to in the file.

Note: Do not use a keyword (ABORT, FILE, PATH, or ERROR) or its valid abbreviation for *label*.

Errors

Directive file processing terminates after an error is generated. Error messages display to the terminal in interactive mode or to SLO in batch mode. $IFF can generate the following errors.

If an attempt is made to compare an integer or a type integer parameter to a string or a type string parameter, $IFF generates the following message and displays the line where the error occurred:

```
INVALID COMPARISON, THE FOLLOWING RECORD IGNORED:
```

If a concatenated string exceeds 72 characters, $IFF generates the following message and displays the line where the error occurred:

```
EXCESSIVE FIELD SIZE, THE FOLLOWING RECORD IGNORED:
```

If GOTO is in effect (implicitly or explicitly) and TSM reaches the end of file without locating the specified label, TSM terminates directive file processing.

If GOBACK is in effect and TSM reaches the beginning of file without locating the specified label, $IFF generates the following message:

LABEL *label* NOT FOUND

label	is the label searched for

Examples

In the following example, `$IFF` controls the repetition of a processing loop. `$IFF` tests whether `%P1` is less than or equal to zero. If the condition is false (if `%P1`>0), TSM branches back to `LOOP1` and repeats its directives. If the test result is true, TSM processes the directive following `$IFF`.

```
$SETI %P1 = 10
$DEFN LOOP1
    .
    .
    .
$SETI %P1 = %P1 - 1
$IFF %P1 LE 0 GOBACK LOOP1
    .
    .
    .
```

1.53 $IFT Directive

The $IFT directive branches to a label if a specified condition is true in a directive file. If a condition is false, TSM processes the next directive in the file. The following conditions can be tested:

- comparison results
- flag setting
- task abort
- file existence
- error occurrence from nested directive files

Valid comparisons can be made between integers and type integer parameters, strings and type string parameters, or two parameters. If two parameters are compared, both must exist and be of the same type.

GOTO and GOBACK options enable TSM to branch forward or backward to the nearest occurrence of a label after testing for a true condition. If GOTO or GOBACK is not specified, TSM branches forward.

$IFT is valid only from interactive and batch directive files.

Syntax

$IFT *condition* [**GOTO** | **GOBA**CK] *label*

condition specifies a condition to be verified as true. Supply one of the following:

ABORT indicates branch if the preceding task aborted

ERROR indicates branch if the called directive file generated an error

FILE *filename*
filename is a 1- to 8-character name of a user or system file. If the file is in the system directory or the current working directory, TSM branches.

flagno is a numeric flag number from 1 to 31. Only one flag number can be specified.

PATH *pathname*
pathname specifies the pathname of a file. If the file exists, TSM branches.

%par oper {*string* | *int* | *%par*}
or
{*string* | *int* | *%par*} *oper %par*

%par is a parameter defined by the $DEFM directive, or created dynamically by the $SET or $SETI directive, or defined by the $DEFM directive and converted to an integer by the $SETI directive. *%par*'s type must match the type being compared to; i.e., an integer compared to an integer parameter, a string compared to a string parameter, or two parameters of the same type.

oper is one of the following valid relational operators:

EQ	equal	NE	not equal
GT	greater than	GE	greater than or equal
LT	less than	LE	less than or equal

string is a 1- to 72-character alphanumeric string. Strings with embedded delimiters must be enclosed in single quotes. Strings can also result from concatenation of multiple parameters or strings. A concatenated string cannot exceed 72 characters.

Note: Do not use a keyword (ABORT, FILE, PATH, or ERROR) or its valid abbreviation for *string*.

int is any 1- to 7-character integer, optionally preceded by a plus (+) or minus (-) sign. If a sign is specified, it must be the first character of the integer and cannot be delimited by blanks.

[**GOTO** | **GOBA**CK]
specifies direction

GOTO branches forward to the next occurrence of *label* (default)

GOBACK branches backward to the previous occurrence of *label*

label is a 1- to 8-character alphanumeric string defined by a $DEFNAME or *%label* directive which marks a point to go to in the file.

Note: Do not use a keyword (ABORT, FILE, PATH, or ERROR) or its valid abbreviation for *label.*

Errors

Directive file processing terminates after an error is generated. Error messages display to the terminal in interactive mode or to SLO in batch mode. $IFT can generate the following errors.

If an attempt is made to compare an integer or a type integer parameter to a string or a type string parameter, $IFT generates the following message and displays the line where the error occurred:

```
    INVALID COMPARISON, THE FOLLOWING RECORD IGNORED:
```

If a concatenated string exceeds 72 characters, $IFT generates the following message and displays the line where the error occurred:

```
    EXCESSIVE FIELD SIZE, THE FOLLOWING RECORD IGNORED:
```

If GOTO is in effect (implicitly or explicitly) and TSM reaches the end of file without locating the specified label, TSM terminates directive file processing.

If GOBACK is in effect and TSM reaches the beginning of file without locating the specified label, $IFT generates the following message:

    LABEL *label* NOT FOUND

label	is the label searched for

Examples

In the following example, `$IFT` controls the repetition of a processing loop. `$IFT` tests whether `%P1` is greater than zero. If the condition is true (if `%P1`>0), TSM branches back to `LOOP1` and repeats its directives. If the condition is false, TSM processes the directive following `$IFT`.

```
    $SETI %P1 = 10
    $DEFN LOOP1
        .
        .
        .
    $SETI %P1 = %P1 - 1
    $IFT %P1 GT 0 GOBACK LOOP1
        .
        .
        .
```

1.54 $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 must be used when SGO is required or when output spooling options are specified.

$JOB is ignored in nested directive files.

Syntax

$JOB *jobname* [*ownername,key*] **[S]** **[SBO=***size***]** **[SBOD=***devmnc***]** **[SBOF=***file***]**
[SGO=*size***]** **[SLO=***size***]** **[SLOD=***devmnc***]** **[SLOF=***file***]**

jobname is a 1- to 8-character job identifier. If more than 8 characters are specified, only the first 8 characters are used.

[*ownername,key*]

ownername is an owner name. The default is the owner name used to log on. If the job is run in batch mode and the owner name is associated with a key, the default is SYSTEM.

key is the key associated with the owner name in the M.KEY file.

[S] specifies sequential execution is required for jobs run by a $BATCH directive under the same owner name. A sequential job runs when all previously entered sequential jobs under that owner name are completed. This parameter is ignored in the interactive mode or if the job was activated by a TSM $SUBMIT directive.

[SBO=*size***]**

size is the initial number of 192-word blocks of disk space to allocate for the job's SBO file. The default is 32 blocks. The actual number of blocks allocated may be greater than the number specified depending on the allocation unit of the disk. If EOM is encountered, the file is extended automatically.

[SBOD=*devmnc***]**

specifies that any SBO files generated by the job are output to the device specified by *devmnc*. If this field is omitted, the final destination is automatically selected from eligible devices specified by the SPOOL parameter of SYSGEN DEVICE directives.

devmnc is a 6-character device mnemonic that consists of a 2-character device code followed by a 4-character hexadecimal device address. (See Appendix A.) The device address consists of a 2-character device channel number followed by a 2-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.

[SBOF=*file*]

specifies that any SBO files generated by the job are accumulated on *file*. If this field is omitted, the final destination is automatically selected from eligible devices specified by the SPOOL parameter of SYSGEN DEVICE directives.

file	is the name of a permanent disk file to contain the job's SBO. If *file* does not exist, it is dynamically created.

[SGO=*size*]

size is the initial number of 192-word blocks of disk space to allocate for the job's SGO file. 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 disk. If EOM is encountered, the file is extended automatically.

[SLO=*size*]

size is the initial number of 192-word blocks of disk space to allocate for the job's SLO file. The default is 32 blocks. The actual number of blocks allocated may be greater than the number specified depending on the allocation unit of the disk. If EOM is encountered, the file is extended automatically.

[SLOD=*devmnc*]

specifies that any SLO files generated by the job are output to the device specified by *devmnc*. If this field is omitted, the final destination is automatically selected from eligible devices specified by the SPOOL parameter of SYSGEN DEVICE directives.

devmnc	is a 6-character device mnemonic that consists of a 2-character device code followed by a 4-character hexadecimal device address. (See Appendix A.) The device address consists of a 2-character device channel number followed by a 2-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.

[SLOF=*file*]

specifies that any SLO files generated by the job are accumulated on *file*. If this field is omitted, the final destination is automatically selected from eligible devices specified by the SPOOL parameter of SYSGEN DEVICE directives.

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

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 EOJ processing.

When 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 and password are verified in the M.KEY file. The M.KEY file contains information such as the key (if applicable), access restrictions, the project group, and the working directory. The following exceptions apply to the $JOB directive:

- An owner name and a key and/or password do not have to be specified. $JOB uses the submitter's owner name by default.
- For SLOF/SBOF, the current working directory is assumed in the interactive mode. Only a 1- to 8-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.

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 EOJ 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 EOJ, 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 is not 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.

Examples

The following example establishes `COPYTAPE` as the job to be processed. The owner name of the job is `USER1`. Listed output is written to the file `TESTLIST` and output to the magnetic tape mounted at channel `10`, subaddress `00`.

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

1.55 $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 on hold 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. The default is the number specified with the SYSGEN DEVICE directive. See the SYC and Terminal I/O section in this chapter for line size effects on reads.

Examples

The following example establishes a screen width of 125 characters.

```
$LINE 125
```

1.56 $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

Examples

The following example displays the file `INDEX` in the current working volume and directory.

```
    $LIST INDEX
```

The following example displays the file `INDEX` in directory `USER2`, volume `ATLAS04`.

```
    $LIST @ATLAS04(USER2)INDEX
```

1.57 $MAPOUT Directive

The $MAPOUT directive dynamically overrides the execution mode specified at SYSGEN if the task has been cataloged using the SYSMAP keyword on the ENVIRONMENT directive. Tasks run mapped out when this directive is used. This directive is ignored if:

- the task is shared
- the current MPX-32 image is not in mapped out execution mode
- the load module was not cataloged with the SYSMAP keyword

This mode is available only on CONCEPT 32/2000 CPU types.

Syntax

$MAPOUT

1.58 $MOUNT Directive

The $MOUNT directive requests the physical mount of a volume to a device and the logical mount of a nonpublic volume to the current TSM environment. A volume can be physically mounted as public or nonpublic.

For nonpublic volumes, the physical mount request also causes an implicit logical mount of the volume to the current TSM environment. If the requested nonpublic volume is already physically mounted to a device, only a logical mount is performed. A volume must be physically mounted for a logical mount to complete.

No operator interaction occurs if the NOMSG option is specified or if the system-wide SNOP or OPCOM SIMM options are enabled.

If the SYSGEN CMIMM option was specified at system initialization, a device must be specified in all mount requests.

All messages generated by the $MOUNT directive appear only on the terminal from which the $MOUNT command was issued, even if the user is logged on to other terminals.

Syntax

$MOUNT *volname* [**ON** *devmnc*] [**OPT**ION=[**PUB**LIC] [**NOM**SG]] [**SYS**ID=*id*]

volname is the volume to be mounted. TSM does not support the wildcard character * when mounting a nonpublic volume. To use the * character, mount your volume with the OPCOM MOUNT directive.

[**ON** *devmnc*]
devmnc specifies which device to mount the volume on. See Appendix A. Physical mount requests must specify a device.

[**OPT**ION=[**PUB**LIC] [**NOM**SG]]

PUBLIC	if specified, mounts the volume for public use (provided the caller has the SA attribute). The default is nonpublic.
NOMSG	if specified, no message displays to the system console when the disk drive is ready. No volume clean-up is performed.

[**SYS**ID=*id*]
id specifies a 3-character port identifier for a multiport volume only. Must be MP0, MP1...MPF. For compatibility, DP0 or DP1 can be entered for MP0 and MP1.

Response

Physical Mount

When a user requests a physical mount, the following message displays on the terminal:

```
    WAITING FOR OPERATOR TO PHYSICALLY MOUNT VOLUME volname
```

At the same time, the following message displays on the system console:

```
    MOUNT VOLUME volname ON devmnc.
    REPLY R, H, A, OR DEVICE:
```

The operator must respond to the prompt.

Once the volume is physically mounted, the following message displays on the terminal:

```
    PHYSICAL MOUNT OF VOLUME volname COMPLETE
```

Logical Mount

If the volume is already physically mounted, the $MOUNT directive causes a logical mount to be performed and displays the following message:

```
    LOGICAL MOUNT COMPLETE
```

When mounting a multiprocessor volume that was not previously dismounted, TSM prompts the operator for volume clean-up. The operator replies (yes) Y or (no) N.

Note: Because volume clean-up deletes temporary files and resets multiprocessor access information and all resource descriptor locks, do NOT perform it if the volume is currently mounted on another system.

If the operator denies the $MOUNT request, TSM terminates batch jobs whose implicit or explicit mounts failed, and prompts on-line users for alternative action.

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

```
    FILE OVERLAP HAS OCCURRED IN RDnnn
    RD TYPE nnn
    FILENAME IS name
    SECTORS sss THROUGH sss
```

nnn	is a hexadecimal number of the resource descriptor
name	is the 1- to 16-character file name
sss	is a hexadecimal number of a sector

If file overlap is detected in the DMAP/SMAP deallocation file descriptor area, the following message is displayed on the system console and the volume is not mounted.

```
  J.MOUNT - ERROR - FILE OVERLAP HAS OCCURRED IN THE BAD SMAP
```

or

```
  J.MOUNT - ERROR - FILE OVERLAP HAS OCCURRED IN THE BAD DMAP
```

To mount the volume so that data can be recovered, set control switch zero or seven.

Errors

See the OPCOM MOUNT directive for a complete list of possible error messages.

Examples

The following example physically mounts volume `VOLUMX` on drive `DM0800` for nonpublic use. It also logically mounts the volume to the current TSM environment.

```
$MOUNT VOLUMX ON DM0800
```

The following example logically mounts the nonpublic volume `VOLUMX` to the current TSM environment.

```
$MOUNT VOLUMX
```

1.59 $NOMAPOUT Directive

This directive dynamically overrides the execution mode specified at SYSGEN if the task has been cataloged using the SYSMAP keyword on the ENVIRONMENT directive. Tasks run mapped in when this directive is used. This directive is ignored if:

- the task is shared
- the current MPX-32 image is not in mapped out execution mode
- the load module was not cataloged with the SYSMAP keyword

Syntax

$NOMAPOUT

1.60 $NOTE Directive

The $NOTE directive sends a message to all terminals logged on by the owner of the job. This job is not affected by the TSM NOCOMMAND directive.

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 is 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.

The $NOTE directive is valid in interactive or batch directive files only.

Syntax

$NOTE *message*

message	is the message to be displayed. The message (including $NOTE) is a maximum of 72 characters. Characters in columns 73 through 80 are considered to be a sequence number and are displayed as such.

1.61 $OBJECT Directive

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

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

Syntax

$OBJECT

1.62 $OPTION Directive

The $OPTION directive sets options that control various aspects of the user operating environment for tasks running online 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 and 49 to 64 are task-dependent; options 21 to 48 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.

Options without a valid numeric equivalent are local to TSM only and do not get propagated to the task option word at task activation.

Syntax

$OPTION [*option*]

[*option*] is one of the following. Specify name or number.

21 **PRO**MPT
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.

22 **LOW**ER
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.

23 **TEX**T 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.

24 **DUM**P dumps the task's area of memory to the SLO file if an abort occurs. The SLO file is printed on the LOD device.

25 **CPU**ONLY
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).

26 **IPU**BIAS
specifies that IPU-compatible tasks execute on the IPU

27 **NOC**OMMAND
inhibits echoing JCL directives as they are read from the SYC file. This option remains in effect until reset by the COMMAND option.

28 **NOE**RROR
inhibits the display of abort code descriptions if an abort occurs

29 **NOA**BORT
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.

30 **RET**AIN
forces any cataloged options to be ORed with any user-supplied options during task activation.

31 **CLE**AR
forces all cataloged task defined options from 1 to 20 and 49 to 64 to be cleared during task activation.

32 not used

33 **NOW**RAP
disables terminal line wrap. Not valid in the batch mode.

34 **L/C** allows characters to be read as entered on a directive line

n is a number from 1 to 64 specifying a particular option available for an MPX-32 utility or user task.

ABORT
causes abort messages to be displayed to the terminal, console, or SLO device in the interactive, batch or real-time environments. This is the default.

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.

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.

QUIET
inhibits messages from being displayed asynchronously on a full duplex terminal. This option has no effect when specified from the console. This is the default.

U/C converts all directive line input to upper case. This is the default.

UNQUIET
: causes messages to be displayed asynchronously on a full duplex terminal. This is the default for the console only.

WRAP
: enables terminal line wrap. Not valid in the batch mode. This is the default.

If no parameters are specified, `$OPTION` will zero the TCA option words, but not the cataloged options. Option `CLEAR` clears out cataloged task defined options 1 through 20 and 49 through 64 during task activation.

Examples

The following example will zero out all task defined options.

```
$OPTI
$OPTI CLEAR
```

The following example sets options `17` and `20`.

```
$OPTION 17 20
```

The following examples set option LOWER.

```
$OPTI 22
$OPTI LOW
```

1.63 $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 on hold and is valid in interactive mode only.

A page size of zero can be specified to output lines without the intervening message. 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.

Examples

The following example inhibits `ENTER CR FOR MORE` messages at the terminal.

```
$PAGE 0
```

1.64 $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 on hold.

Syntax

$PRINT *pathname* [**COP**IES=*number*] [**DEV**ICE=*devmnc*] [**FOR**MAT={**Y** | **N**}]

pathname specifies the pathname of the file to be printed

[**COP**IES=*number*]
number is a decimal number specifying the number of copies to be printed. The default is one.

[**DEV**ICE=*devmnc*]
devmnc is a device mnemonic (see Appendix A). The output defaults to the autoselectable device defined with the SYSGEN DEVICE directive.

[**FOR**MAT={**Y** | **N**}]
If Y, the character in column one is interpreted as a carriage control character. If N, the character in column one is interpreted as data. If omitted, the character in column one is interpreted as a carriage control character for SLOF files and as data for all other operations.

Examples

The following example prints two copies of file `INDEX` from the current working volume and directory.

```
$PRINT INDEX COP=2
```

The following example prints one copy of file `INDEX` from volume `ATLAS04` in directory `USER`. Characters in column one are interpreted as data.

```
$PRIN @ATLAS04(USER)INDEX FOR=N
```

1.65 $RECALL Directive

The $RECALL directive retrieves one or more commands from the recall buffer. The commands are numbered and may be edited using the standard recall edit functions (see Chapter 10), then reissued by entering a <ret>. The reissued commands are written to the recall buffer, but the $RECALL directive itself is not.

If a range is selected with the $RECALL directive, the up arrow is not functional while $RECALL is processing the requested range. The $RECALL directive cannot be reissued while the range is processing. To exit the range processing prior to completion, enter the [exit recall] key. To skip one or more commands within the requested range, use the down arrow.

When the recall buffer is displayed using the [display] key, the numbers appear at the left of each displayed command. The $RECALL directive can also be used to display the contents of the recall buffer.

If the $RECALL directive is entered without a number, the last command is recalled. The pound sign (#) can be used instead of the word RECALL.

Syntax

${ RECALL | # }[*n1*[:*n2*] | **SHOW**]

n1	is the number of the first command to be recalled
:	is a required delimeter, if entering a range
n2	is the number of the last command in the range to be recalled
SHOW	displays the contents of the recall buffer **Note:** This command provides the same functionality as the [display] key.

Errors

If no command has been issued (e.g. user just logged on) and a $RECALL directive is issued, the following message is output and no action is taken.

```
*RECALL BUFFER IS EMPTY
```

If a $RECALL directive is issued and the number is invalid (e.g. greater than the number of commands in the buffer) or the range is in error (e.g. second number is less than first number), the following message is output and no action is taken.

```
*INVALID COMMAND LINE RECALL PARAMETER
```

If a $RECALL directive is issued and the recall function is turned off, the following message is output and no action is taken.

```
*COMMAND LINE RECALL & EDIT IS NOT AVAILABLE
```

Examples

The following examples recall command 6.

```
$RECALL 6                RECALL 6

$RECA 6                  RECA 6

$#6                      #6
```

The following examples recall commands 3 through 7.

```
$RECALL 3:7              RECALL 3:7

$RECA 3:7                RECA 3:7

$#3:7                    #3:7
```

The following examples display the contents of the recall buffer.

```
$RECALL SHOW             RECALL SHOW

$RECA S                  RECA S
```

1.66 $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 on hold and is valid in interactive mode only.

Syntax

$REMOVE *jobno*

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

Response

LF (line feed)

Examples

The following example removes job number `700` from the system.

```
$REMOVE 700
```

1.67 $RENAME Directive

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

Syntax

$RENAME *currname newname*

currname	is the pathname of the file to be renamed
newname	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.

Examples

The following example renames the file `TEST1` on the current working volume and directory to `LAB1`.

```
$RENA TEST1 LAB1
```

The following example renames the file `TEST1` on volume `TB00` in directory `USER1` to file `LAB1` on volume `TB00` in directory `USER2`.

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

1.68 $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 $IFF directives within a directive file. $RESETF should precede any $IFF or $IFT directives that check the condition of the specified flags. The $RESETF 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.

Examples

The following example sets flags `3`, `7`, and `9` to false:

```
$RESETF 3 7 9
```

1.69 $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

1.70 $SELECT Directive

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

A $SELECT directive can be used to select a directive file other than the current directive file. TSM terminates processing of the current file and begins processing the selected file. 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* [*arg*] ...

pathname	is the pathname of a permanent file containing directives
[*arg*]	is a parameter to pass for the directive file. Up to 16 arguments 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, TSM 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, TSM reads the next directive from the directive file, if any. Directive file processing terminates when end-of-file, $CLEAR directive, $END directive, or $ENDM directive is encountered.

Errors

If an invalid directive is detected in the directive file, TSM prompts the user for interactive input. The user can correct the error, enter any valid directive, or enter a carriage return to skip the invalid directive and execute 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.

1.71 $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* [**E**|**O**] [**H**|**L**] [**I**|**P**] [**NORE**] [**UNB**LOCKED] [*skips reads*]

devmnc is the 6-character device mnemonic and address of a device. See Appendix A.

reel is a 4-character tape reel identifier required if magnetic tape is specified

[**E**|**O**] specifies the tape parity. This option is only available for 7-track magnetic tapes.

E = even parity (default)
O = odd parity

[**H**|**L**] specifies the tape density. This option is only available for 7-track magnetic tapes.

H = 800 BPI (default)
L = 556 BPI

[**I**|**P**] specifies the tape format. This option is only available for 7-track magnetic tapes.

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

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

[**UNB**LOCKED]
specifies that the magnetic tape is unblocked, i.e., contains one logical record per physical record. If this parameter is omitted, blocked is assumed.

[*skips reads*]
specifies the number of files to skip and read

skips is the number of files to skip prior to reading the magnetic tape. The default is zero. If this parameter is specified, *reads* must also be specified.

reads is the number of files to read from the magnetic tape. The default is one file. EOF marks read from the tape are not written to the SYC file. If *reads* is specified, *skips* must also be specified.

1.72 $SELECTF Directive

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

Syntax

$SELECTF *pathname* [*password*] [**UNB**LOCKED] [*skips reads*]

pathname is the pathname of a permanent file. If a directory 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 associated with a key, 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.

[*password*] is ignored

[**UNB**LOCKED]
specifies the disk file was written in the unblocked mode. The default is blocked.

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

[*skips reads*]
specifies the number of files to skip and read

skips is the number of files to skip prior to batch input data records. The default is zero. If the UNBLOCKED option is requested, *skips* must not be specified or must be equal to zero. If *skips* is specified, *reads* must also be specified.

reads is the number of files to read. The default is one file. EOF marks are not copied to the SYC file. If the UNBLOCKED option is requested, *reads* must not be specified or must be equal to one. If *reads* is specified, *skips* must also be specified.

1.73 $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* **[E | O] [H | L] [I | P] [NORE] [UNBLOCKED]** *header*

devmnc is the 6-character device mnemonic and address of a card reader, paper tape reader, or magnetic tape device. The entry consists of a 2-character device mnemonic (see Appendix A) followed by a 4-character hexadecimal device address, such as CR7800. The device address consists of a 2-character device channel number followed by a 2-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.

reel is a 4-character tape reel identifier. Required for magnetic tape.

[E | O] specifies the tape parity. This option is only available for 7-track magnetic tapes.

E = even parity (default)
O = odd parity

[H | L] specifies the tape density. This option is only available for 7-track magnetic tapes.

H = 800 BPI (default)
L = 556 BPI

[I | P] specifies the tape format. This option is only available for 7-track magnetic tapes.

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

[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 1- to 8-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.

1.74 $SELECTLF Directive

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

This directive is valid in batch directive files only.

Syntax

$SELECTLF *pathname* [**UNB**LOCKED] *header* [*password*]

pathname	is the pathname of a permanent file. If a directory 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.
[**UNB**LOCKED]	specifies the disk file was written in the unblocked mode. The default is blocked.
header	is the 1- to 8-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

1.75 $SELECTS Directive

The $SELECTS directive resets all alternate system input source levels established by previous $SELECT*x* 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

1.76 $SET Directive

The $SET directive assigns a string value to a parameter. The value overrides any default $DEFM value or value passed by a $SELECT or $CALL directive. A parameter not previously defined with $DEFM is created dynamically. Up to 16 parameters can exist in a file.

In an interactive session, a missing value on the $SET directive issues a read to the terminal to obtain a user-specified value. If a prompt to the user is desired, precede the $SET directive with a $NOTE directive.

$SET is valid only from interactive and batch directive files.

Syntax

$SET *%par* [*string* | *%par*]

%par	is a parameter defined by the $DEFM directive or created dynamically by $SET. Any parameter appearing as the second parameter on the $SET directive must already exist.
string	is any 1- to 72-character alphanumeric string. The actual length of the string is limited by the length of existing arguments and the length of the $SET line. The $SET line is truncated after column 72 and strings obtained by a read to the terminal are truncated after 72 characters. Strings with embedded delimiters must be enclosed in single quotes.

Errors

Directive file processing terminates after an error is generated. Error messages display to the terminal in interactive mode or to SLO in batch mode. $SET can generate the following error messages.

If an attempt is made to dynamically create a parameter when 16 parameters already exist, $SET outputs the following message and displays the line where the error occurred:

```
EXCESSIVE NUMBER OF PARAMETERS, THE FOLLOWING RECORD IGNORED:
```

If the total number of characters of all arguments exceeds 256, $SET outputs the following message and displays the line where the error occurred:

```
PARAMETER TABLE SIZE HAS BEEN EXCEEDED, THE FOLLOWING RECORD
IGNORED:
```

Examples

In the following example, the first `$SET` directive sets parameter `%P1` to `StringOne`. The second `$SET` directive sets `%P2` to the value of `%P1` (`StringOne`). The third `$SET` directive dynamically creates `%P3` and sets `%P3` to `String Two`. `String Two` is single-quoted because it contains an embedded blank.

```
$DEFM P1 P2
$SET %P1 StringOne
    .
    .
    .
$SET %P2 %P1
$SET %P3 'String Two'
    .
    .
    .
```

1.77 $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. In an interactive session, flags are reset only at logon time. Flags are not reset when a $JOB or $EOJ is specified.

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

Syntax

$SETF *flagno* [*flagno*] ...

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

Examples

The following example sets flags `3`, `7`, and `9` to true:

```
$SETF 3 7 9
```

1.78 $SETI Directive

The $SETI directive assigns an integer value or the results of a valid arithmetic expression to a parameter. The value overrides any set by the $DEFM directive or passed by a $SELECT or $CALL directive. A parameter not previously defined with $DEFM is created dynamically. Up to 16 parameters can exist in a file.

Operands in an arithmetic expression can be type integer parameters, signed or unsigned integers, or numeric type string parameters. The equal sign (=) and addition (+) and subtraction (-) symbols must be delimited by blanks.

In an interactive session, a missing value on $SETI issues a read to the terminal to obtain a user-specified value. An equal sign must follow the *%par*. If a prompt to the user is desired, precede the $SETI directive with a $NOTE directive.

$SETI is valid only from interactive and batch directive files.

Syntax

$SETI *%par* = [[+|-]{*%par*|*int*}] [{+|-}{*%par*|[%]*int*}] ...

par	is a parameter defined by the $DEFM directive or created dynamically by $SETI. Any parameter to the right of the equal sign must be either a type integer or a numeric type string parameter.
int	is any 1- to 7-character integer, optionally preceded by a plus (+) or minus (-) sign. If a sign is specified, it must be the first character of the integer and cannot be delimited by blanks.

Errors

Directive file processing terminates after an error is generated. Error messages display to the terminal in interactive mode or to SLO in batch mode. $SETI can generate the following error messages.

If an attempt is made to set a parameter to a numeric value less than -9999999 or greater than 9999999, $SETI outputs the following message and displays the line where the error occurred:

```
PARAMETER VALUE OUT OF RANGE, THE FOLLOWING RECORD IGNORED:
```

If an attempt is made to set a parameter to a non-numeric string or a non-numeric string type parameter, $SETI outputs the following message and displays the line where the error occurred:

```
INVALID PARAMETER TYPE, THE FOLLOWING RECORD IGNORED:
```

If an attempt is made to dynamically create a parameter when 16 parameters already exist, $SETI outputs the following message and displays the line where the error occurred:

```
EXCESSIVE NUMBER OF PARAMETERS, THE FOLLOWING RECORD IGNORED:
```

If the total number of characters of all arguments exceeds 256, $SETI outputs the following message and displays the line where the error occurred:

```
PARAMETER TABLE SIZE HAS BEEN EXCEEDED, THE FOLLOWING RECORD IGNORED:
```

Examples

In the following example, the first `$SETI` directive sets the parameter `%P1` to `-10`. The second `$SETI` directive sets `%P2` to the result of an arithmetic equation that includes `%P1`.

```
$DEFM %P1 %P2
$SETI %P1 = -10
    .
    .
    .
$SETI %P2 = 100 + %P1 - 550
    .
    .
    .
```

1.79 $SHADOW Directive

The $SHADOW directive specifies the portions of a task's logical address space to be located in shadow memory. The portions are specified using relative or absolute addresses. Multiple $SHADOW directives have a cumulative effect and allow noncontiguous portions of the task to be located in shadow memory. 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 use of dynamic memory allocation services.

The $CLEAR directive clears $SHADOW directives.

Syntax

$SHADOW {**ALL** | **STA**CK | **TAS**K | **TSA** | *start end* [**A**BSOLUTE | **C**ODE | **D**ATA]}
[**REQ**UIRED]

ALL specifies that the entire task, including the TSA, is to be located in shadow memory

STACK specifies that the task's stack is to be located in shadow memory. This option applies only to base mode tasks and is ignored for nonbase mode tasks.

TASK specifies that the entire task, except the TSA, is to be located in shadow memory. If the task plus its TSA are only one map block, the entire space is shadowed.

TSA specifies only the TSA is to be located in shadow memory. If the task plus its TSA are only one map block, the entire space is shadowed.

start is the hexadecimal logical starting address of the logical address space to be located in shadow memory

end is the hexadecimal logical ending address of the logical address to be located in shadow memory

The following specify that *start* and *end* are:

ABSOLUTE relative to logical address zero

CODE 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 relative to the task's data section origin as opposed to the default assumption that the addresses are relative to the task's start.

[**REQ**UIRED]
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.

1.80 $SHOW Directive

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

Syntax

$SHOW [CPUTIME **|** **JOB**S **|** **USE**RS]

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 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 CPU`, 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 JOB`, the display format is:

jobno ownername jobname priority taskno taskname

jobno	is the job sequence number from 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 from 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 USE`, 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 log on
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

Examples

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

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

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

1.81 $SIGNAL Directive

The $SIGNAL directive sends a message to a logged on user or to all terminals. This directive can be used when a task is on hold. If a signal is sent to an owner who is logged on to more than one terminal, it goes to all terminals logged on by that owner.

Batch users can signal interactive users but cannot receive messages.

Syntax

$SIGNAL [*ownername*]

[*ownername*]
is the owner name of a logged on user. If the user is not logged on, the message is denied. If *ownername* is logged on to more than one terminal, the message is sent to all terminals logged on by *ownername*. If *ownername* is not specified, the message is sent to all logged on terminals.

Response

The following message is displayed:

```
ENTER MESSAGE
```

The message is a maximum of 80 characters and is terminated by a carriage return.

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

1.82 $SPACE Directive

The $SPACE directive expands a task's logical address space. This directive is ignored on the 32/87 computer. 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.

1.83 $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 is valid from interactive and batch directive files and can be used when a task is on hold. See the Batch Stream Memory Pool Interaction section of this chapter 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 $SELECT*x* 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 directive in the directive file, unless the $DEFM directive is specified. If $DEFM is specified, then the $JOB must be the second directive in the directive file.

Syntax

$SUBMIT *pathname* [*arg*] ...

pathname	is the pathname of a permanent file containing TSM directives
[*arg*]	is a parameter to pass to the directive file. Up to 16 arguments are passed.

Response

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

jobno ownername `$JOB` *jobname*

Examples

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

1.84 $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={**SLO**|**UT**}

SLO	spools assignments to SLO. If a $JOB statement is in effect, the SLO is spooled to an extendible file until an $EOJ statement is encountered.
UT	directs all SLO to the user's terminal. This applies to both static and dynamic assignments, and overrides an SLOF or SLOD option specified on a $JOB statement.

1.85 $URGENT Directive

The $URGENT directive changes the priority of a batch job. The job must be queued or active. This directive can be used when a task is on hold 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 from 1 to 9999
priority	is the software priority (55 to 64) at which the job is to run

Response

LF (line feed)

Examples

The following example changes the priority of job number `700` to priority `60`.

```
$URGENT 700,60
```

1.86 $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 1- to 8-character name of a directory on the current working volume. If omitted, all files created are system files and only system files on the current working volume are accessed.

Examples

The following example changes the current working directory to `SOURCE`.

```
$USER SOURCE
```

1.87 $WAIT Directive

The $WAIT directive puts a terminal into a wait state where it can receive messages. For example, $WAIT is used to receive messages about submitted batch jobs before a user continues terminal operation or logs off. 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 wakeup character to return to interactive operation.

If $WAIT is not used, an inactive terminal is in a read and is not interrupted by messages unless the terminal times out. Users must issue carriage returns to exit a read and display a message.

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

This directive can be used when a task is on hold 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

Examples

```
     EDT>COL
     1.  $JOB X OWNER1,G SLOF=SLOF1
     3.  $EXECUTE VOLMGR
     4.  LOG
     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.

1.88 $WHO Directive

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

Syntax

$WHO

Examples

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

1.89 $$ Directive

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

$$

1.90 $$$ Directive

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

$$$

1.91 Examples

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

1.91.1 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
$$
```

1.91.2 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 ?

MPX-32 REVISION 3.5 xxxx TBDSYS99 TERMINAL SERVICES MANAGER
(C) COPYRIGHT 1990, ENCORE COMPUTER CORPORATION, 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
```

1.91.3 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.
                                   (Assembler 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 $SELECTD 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
```

1.91.4 Conditional Batch Processing

Example 1

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

```
$JOB EXAMP5 USERF
$IFT FILE ABC FILEPR
$EXECUTE VOLMGR
CREATE ABC
$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
```

1.91.5 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
```

Examples

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 argument is absent, select options `3` and `4` and continue. If not absent, enable the specified option plus options `3` and `4`.

3 Assign a file named in the first argument to `SI` or use `SI` if the first argument 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 argument with a string.

Assign listed output to the terminal.

4 Execute the assembler.

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

6 If anything other than `XA` is entered as the second argument, exit. If `XA` is entered as the second argument, 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 LIBFILE
        DELETE DIRFILE
        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, 1 to 16 parameters are supplied, and each parameter name is 1 to 16 characters.

```
$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.

1.92 Batch Stream Memory Pool Interaction

The batch stream memory pool interaction facility monitors the availability of the memory pool. This protects the system from memory pool deadlocks caused by excessive $SUBMIT or $BATCH commands. When the miscellaneous 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 batch stream 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 <ret> 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.
```

Batch Stream Memory Pool Interaction

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 because run requests are sent while it is waiting to use the memory pool.

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.

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 held task or device
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	deletes the current SBO file from the system output queue
DISABLE	disables an interrupt at a specified priority level
DISCONNECT	disconnects a task from an indirectly connected interrupt level defined at SYSGEN

Continued on next page

Directive Summary

Directive	Function
DISMOUNT	requests the physical dismount of a user 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 a CONTINUE directive
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
MOUNT	requests the physical mount of a user volume to 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	repunches the current SBO file
REQUEST	generates a request interrupt (RI) instruction for a specified interrupt priority level
RESUME	resumes a suspended 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

Continued on next page

Directive	Function
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
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

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 (`??`) indicates that OPCOM is ready to accept directive input. OPCOM issues this prompt after processing an OPCOM directive.

Alternatively, an OPCOM directive can be issued at the TSM prompt with immediate return to TSM rather than remaining in OPCOM. To do this, precede 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, J.TSM, 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.

task number	is an 8-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.
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.
owner name	for a task activated from a terminal is the name specified at logon. It cannot be changed except by logging on again. Usually, 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 lists all tasks of the specified name, their task number, and other information that allows 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, users who issue an [EXECUTE] OPCOM directive from their terminal can control only those tasks that they activated. If users issue a control directive for a task they do 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 (<ret>). 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 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 carriage return/line feed. It then reissues the OPCOM or TSM prompt at the terminal.

Any typing error on the directive line can be corrected by using the command line edit functions. See the Command Line Edit section in Chapter 10. If command line edit is disabled or no MPX.PRO file exists, then pressing <ctrl>H or the backspace key deletes a character and pressing the rub or delete key erases the entire command line.

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

taskno
: is the 8-digit task number assigned at activation

Response

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 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 is issued when a task is already in an abort condition) kills outstanding I/O and deletes the task.

Examples

The following example aborts the task `PGMTEST` if it is a single copy load module with no outstanding allocation requirements or outstanding I/O.

```
ABORT T,PGMTEST
```

The following example aborts task number `02000001` if it has no outstanding allocation requirements or outstanding I/O.

```
ABORT 02000001
```

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. An 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 1- to 8-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

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

If activation is not successful (e.g., an assigned device is not configured in the system), an abort code and message are displayed on the system console.

Examples

The following example activates the permanent file `PGMTEST`.

```
ACTIVATE 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*]

[**D,** *devmnc* [, *density*, *parity*]

D, *devmnc*	*devmnc* is the mnemonic of the device to be specified. It 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

If no parameters are specified, the job file is read from the SID.

Response

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.

Examples

The following example reads batch jobs from the current SID.

```
BATCH
```

The following example reads batch jobs from the card reader on channel `7A`, subaddress `00`.

```
BATCH D,CR7A00
```

The following example reads batch jobs from the file `PGMTEST` on the system volume and directory.

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

2.13 BREAK Directive

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

Syntax

BREAK {**T**, *taskname* | *taskno*}

T, *taskname*	*taskname* is the task name. The task must be a unique copy.
taskno	is the task number assigned at activation

Response

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

Examples

In the following example, if the task `PGMTEST` is a unique copy with a break receiver, the break receiver is entered.

```
BREAK T,PGMTEST
```

In the following example, if task number `02000001` has a break receiver, the break receiver is entered.

```
BREAK 02000001
```

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

intlevel	is the 2-character hexadecimal interrupt priority level
taskno	is the 8-digit task number assigned at activation

Response

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.

Examples

The following example 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 T,PGMTEST,2F
```

The following example connects task number `02000001` to interrupt level `2F`.

```
CONNECT 02000001,2F
```

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 {[**PRI**NT | **PUN**CH | **REA**D] [, *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*	*taskname* is the task name. If *taskname* is used to continue a task, the task must be a unique copy.
taskno	is the 8-digit task number assigned at activation

Response

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.

Examples

The following example continues output to the current system LOD.

```
CONTINUE PRINT
```

The following example continues input from the card reader on channel `78`, subaddress `01`.

```
CONT READ,CR7801
```

The following example continues the task `PGMTEST` if it is a unique copy task.

```
CONT T,PGMTEST
```

The following example continues task number `02000001`.

```
CONT 02000001
```

2.16 DEBUG Directive

The DEBUG directive transfers control (via branch and link) to the system debugger (H.DBUG).

Syntax

DEBUG

Response

Although the DEBUG directive can be issued from any terminal, once the system debugger gains control, its prompt (>>) displays on the system console. It accepts directives only from that device. The debugger routes listings to the printer configured as LP7E.

2.17 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 using the SETTIMER directive.

Syntax

DELETETIMER *timer*

timer	is the 2-character ASCII name of the timer to be deleted

2.18 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 [**D,***devmnc* | **J,***jobno* | **T,**{*taskname* | *jobname*} | *taskno*]

D, *devmnc*	*devmnc* is a device mnemonic used to delete the SLO file currently being output on a device other than the SYSGEN-defined (listed output device) LOD
J, *jobno*	*jobno* is the job sequence number assigned when the job was queued. All SLO files for the job are deleted.
T,{*taskname* | *jobname*}	*taskname* is the task name. All SLO files for the task are deleted. *jobname* is the job name as specified in the $JOB statement. All SLO files for the job are deleted.
taskno	is the 8-digit task number assigned at activation. All SLO files for the task are deleted.

If no parameters are specified, the SLO file currently being output to the system LOD is deleted.

Response

The system output task producing the SLO file deletes the file from the system output queue. If a specified SLO file is being printed, printing stops.

Examples

The following example deletes all SLO files for job number `700`.

```
DEPRINT J,700
```

The following example deletes all SLO files for the task `PGMTEST`.

```
DEPR T,PGMTEST
```

The following example deletes all SLO files for task number `02000001`.

```
DEPR 02000001
```

The following example deletes the SLO file currently printing on the LOD.

```
DEPR
```

The following example deletes the SLO file currently being output to the printer on channel `7A`, subaddress `00`.

```
DEPRINT D,LP7A00
```

2.19 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 [**D**,*devmnc* | **J**,*jobno* | **T**,{*taskname* | *jobname*} | *taskno*]

D,*devmnc*	*devmnc* is a device mnemonic used to delete the SBO file currently being output on a device other than the SYSGEN-defined punched output device (POD)
J,*jobno*	*jobno* is the job sequence number assigned when the job was queued. All SBO files for the job are deleted.
T,{*taskname* | *jobname*}	*taskname* is the task name. All SBO files for the task are deleted. *jobname* is the job name as specified in the $JOB statement. All SBO files for the job are deleted.
taskno	is the 8-digit task number assigned at activation. All SBO files for the task are deleted.

If no parameters are specified, the SBO file currently being output to the system POD is deleted.

Response

If a specified SBO file is being output, output stops.

Examples

The following example deletes all SBO files for job number `700`.

```
DEPUNCH J,700
```

The following example deletes all SBO files for the task `PGMTEST`.

```
DEPU T,PGMTEST
```

The following example deletes all SBO files for task number `02000001`.

```
DEPU 02000001
```

The following example deletes the SBO file currently being punched on the current system POD.

```
DEPU
```

2.20 DISABLE Directive

The DISABLE directive executes a disable channel interrupt instruction, (DCI for 'F' class devices or DI for 'E' class devices) 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 or EI) instruction must be executed. See the ENABLE directive.

Syntax

DISABLE *intlevel*

intlevel	is the 2-character hexadecimal interrupt priority from 00 to 7F

Response

If the request is unsuccessful, the following message is displayed:

```
REQUEST NOT EXECUTED
```

Examples

The following example disables priority interrupt `01` from its associated channel.

```
DISA 01
```

2.21 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*	*taskname* is the task name. If *taskname* is used to disconnect a task, the task must be a unique copy.
taskno	is the 8-digit task number assigned at activation

Examples

The following example disconnects task `PGMTEST` from its indirectly connected interrupt level if the task is a unique copy task.

```
DISCONNECT T,PGMTEST
```

The following example disconnects task number `02000001` from its indirectly connected interrupt level.

```
DISC 02000001
```

2.22 DISMOUNT Directive

The DISMOUNT directive requests the physical dismount of a volume from a device.

If users are currently active on the volume requested for dismount, the volume is placed in a state of pending dismount. Subsequent requests for logical or physical mount of the volume are denied.

When no users are active on the volume, the physical dismount is performed. When the dismount completes, a prompt issues to the system console to inform the operator that the volume can be physically removed from the drive. The operator must respond. If the NOMSG option is specified or the system-wide SNOP or OPCOM SIMM options are enabled, the dismount completes without operator interaction.

If the SYSGEN CMPMM option was specified at system initialization, public volumes cannot be dismounted from the running system. If the SYSGEN CMIMM option was specified, OPCOM DISMOUNT dismounts only volumes physically mounted through OPCOM.

OPCOM uses the M.DMOUNT service (or M_DISMOUNT in base mode) to perform the dismount.

Messages generated by the DISMOUNT directive in interactive mode go to all terminals logged on with the same owner name as the owner who issues the DISMOUNT command.

A public volume dismount request is valid only from the system administrator (SA).

Syntax

DISMOUNT *volname* [**FROM** *devmnc*] [**OPT**ION=[**NOM**SG] [**PUB**LIC]]

volname is the name of the volume to be dismounted

[**FROM** *devmnc*]
specifies the device where the volume is mounted (see Appendix A)

[**OPT**ION=[**NOM**SG] [**PUB**LIC]]

NOMSG	specifies dismount without operator interaction. This option is ignored if the NOMSG option is specified in the MOUNT directive.
PUBLIC	specifies the volume to be dismounted is a public volume. Request is only valid from the system administrator (SA).

Response

When a physical dismount of a public volume is requested, the system console displays the following warning message:

```
WARNING - ATTEMPTING DISMOUNT OF A PUBLIC VOLUME
```

If a user is attached to the volume, physical dismount is postponed and the following message displays to the caller's terminal:

```
DISMOUNT PENDING DUE TO OUTSTANDING ASSIGNMENTS
```

When there are no users active on the volume, the physical dismount completes. The following message displays to the system console to prompt the operator to physically remove the dismounted volume from the drive, if removable:

```
CONFIRM PHYSICAL DISMOUNT OF VOLUME  volname
FROM device
REPLY R TO RESUME:
```

The operator must respond with a character to confirm the dismount. Confirmation is required in order for further mount or dismount activity to continue in the system.

When the physical dismount completes, a message displays to the terminal and the system console:

```
PHYSICAL DISMOUNT OF VOLUME  volname
FROM device COMPLETE.
```

If the volume is currently mounted to another processor, the following message displays to the requestor's terminal:

```
VOLUME  volname CURRENTLY MOUNTED ON FOLLOWING PORTS: [port id,...]
```

The DISMOUNT directive can be used to deallocate a volume when the volume is not present on the device.

Examples

The following example causes volume VOL1 to be dismounted from disk device DM0800 when the volume is inactive without intervening dismount messages.

```
DISM VOL1 FROM DM0800 OPT=NOMSG
```

2.23 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 ASCII-coded hexadecimal side-by-side 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

DUMP {*start,end* | **ON** | **OFF**}

start,end	specifies the starting and ending hexadecimal physical word addresses
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

If an SLO file cannot be dynamically allocated when requested, the following message is displayed:

```
DUMP NOT PERFORMED - TRY AGAIN LATER
```

Examples

The following example dumps the contents of physical memory between logical address `3000` and logical address `3FFF` to the SLO file.

```
DUM 3000,3FFF
```

2.24 ENABLE Directive

The ENABLE directive executes an enable channel interrupt instruction, (ECI for 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 2-character hexadecimal interrupt priority level from 00 to 7F

Response

If the request is unsuccessful, the following message is displayed:

```
REQUEST NOT EXECUTED
```

Examples

The following example enables priority interrupt level `01`.

```
ENA 01
```

2.25 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* | *dd-mo-yy* | *ddmonyy*} *hh:mm:ss* } [,[**D**] [,**TZ=***num*]]

{*mo/dd/yy* | *dd-mo-yy* | *ddmonyy*}
specifies the date

mo	is the 2-digit decimal month
mon	is the 3-character ASCII month abbreviation
dd	is the 2-digit decimal day
yy	is the 2-digit decimal year

hh:mm:ss specifies the time

hh	is the 2-digit decimal hour
mm	is the 2-digit decimal minute
ss	is the 2-digit decimal second

[**D**] indicates daylight savings time is in effect

[,**TZ=***num*] *num* is the number of hours to bias the internal binary time. This can be a negative or positive number.

Examples

The following example 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
```

The following example indicates daylight savings time is in effect.

```
ENTER 09/07/58 12:30:59,D
```

The following example specifies a positive three-hour time bias. The double comma indicates daylight savings time is not in effect.

```
ENTER 07-09-58 12:30:59,,TZ=3
```

The following example specifies daylight savings time and a negative one-hour time bias.

```
ENTER 07SEP58 12:30:59,D,TZ=-1
```

The following example sets the system date to September 7, 1989 and the system time to 12:30:00 p.m.

```
ENTER 09/07/89 1230
```

2.26 ESTABLISH Directive

The ESTABLISH directive activates a task and suspends it at the end of the activation sequence. This task remains inactive until resumed 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, the task is brought into execution at its base priority (cataloged or linked).

This directive enables a user task to activate and suspend until resumed 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

The task is activated, then suspended.

Examples

The following example establishes the permanent load module or executable image file `PGMTEST` with the logon owner name.

```
EST PGMTEST
```

2.27 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

Examples

The following example removes the resident shared image `SHARE1` on the system volume in directory `DIR1` from memory.

```
EXCLUDE SYSTEM(DIR1)SHARE1
```

The following example removes the resident shared image `SHARE1` in the user's current working volume and directory from memory.

```
EXC SHARE1
```

2.28 EXIT Directive

The EXIT directive terminates OPCOM and returns control to TSM.

Syntax

EXIT

2.29 HOLD Directive

The HOLD directive places a system output task, system input task, or a specified user or system task on hold.

Syntax

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

PRINT	if no device mnemonic is specified, holds the system task controlling SLO output to the system listed output device (LOD)
PUNCH	if no device mnemonic is specified, holds the system task controlling SBO output to the system punched output device (POD)
READ	if no device mnemonic is specified, holds the system task reading input from the system input device (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*	*taskname* is the task name. If *taskname* is used to hold a task, the task must be a unique copy.
taskno	is the 8-digit task number assigned at activation

Response

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.

Examples

The following example holds output to the current SLO device.

```
HOLD PRINT
```

The following example holds input from the card reader device on channel `78`, subaddress `01`.

```
HOLD READ,CR7801
```

The following example holds the task `PGMTEST` if it is a unique copy.

```
HOLD T,PGMTEST
```

The following example holds task number `02000001`.

```
HOLD 02000001
```

2.30 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

Examples

The following example loads the shared image `SHARE1` on the system volume in directory `TEST` into memory.

```
INCLUDE @ SYSTEM(TEST)SHARE1
```

The following example loads the shared image `SHARE1` from the user's current working volume and directory into memory.

```
INC SHARE1
```

2.31 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 EOF marks are not written.

Syntax

KILL {T, *taskname* | *taskno*}

T, *taskname*
taskname is the name of a single-copy task

taskno is the task number assigned at activation

Response

Processing is not deferred for outstanding I/O or run requests. All outstanding I/O is terminated. The DQE for the task is deleted.

Examples

The following example kills the task `PGMTEST` if it is a unique copy.

```
KILL T,PGMTEST
```

The following example kills task number `02000001.`

```
KIL 02000001
```

2.32 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 [**ACCO**UNT [*options*] | **EXE**CUTION | **IPU** | **PAT**CHES | **PRI**NT | **PUN**CH | [*taskname*] [,[*ownername*] [,*pseudonym*]]]

ACCOUNT [*options*]
copies the job accounting file contents to an SLO file. The following options can be specified in any combination to limit the display. Commas can optionally separate the entries.

OWNE=*name*
name is an owner name

PROJ=*proj*
proj is a project name

DATE=*date*
date is the job's latest execution date

ORIG={**TSM**[.*nnnn*] | **BAT**CH | **JOB**.*nnnn*}
specifies the mode of operation. TSM indicates origination from TSM interactively and can optionally include a 4-digit terminal number. BATCH or JOB indicates origination as a batch job. Follow JOB with a 4-digit job number. The question mark can be used as a single character wild card within the device or job number.

EXECUTION
lists all entries in the system dispatch queue. This is the default.

IPU lists the last twenty events which caused a trap in the IPU

PATCHES
copies the system patch file contents to an SLO file

PRINT lists entries in the SLO file output queue

PUNCH lists entries in the SBO file output queue

[*taskname*] [,[*ownername*] [,*pseudonym*]]
lists status of tasks matching the specified parameters

taskname	is the task name. If not specified, all tasks with the specified owner name and/or pseudonym are listed.
ownername	specifies the owner name for a particular task or all tasks belonging to the owner. If *taskname* is not specified, the comma is still required. For example: `LIST,`*ownername* If *ownername* 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 a 4-digit terminal number, and job control uses a device mnemonic (see Appendix A). The pseudonym can also 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 *taskname* and *ownername* 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) in the format shown for LIST EXECUTION below. 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 in the format shown for LIST EXECUTION below.

To LIST EXECUTION or tasks selected by task identifiers for all tasks on the 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 from 1 to 64
state	is a 4-character identifier corresponding to the state of the task. See the STATUS directive in this chapter (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

To LIST PRINT or LIST PUNCH, the following is displayed:

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, the current contents of the accounting file is written 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.

The following is the format of the display:

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*	*time* is the amount of CPU time used
`IPU` *time*	*time* is the amount of IPU time used

To LIST PATCHES:

The contents of the system patch file is written to an SLO file for printing.

To LIST IPU:

The last 20 events that caused a trap in the IPU are displayed. If less than 20 traps occurred, only those that occurred are displayed.

The following is the format of the display:

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.

Examples

The following example lists all entries in the dispatch queue with the task name PGMTEST.

```
LIST PGMTEST
```

The following example lists all entries in the dispatch queue with the task name PGMTEST and owner name USER1.

```
LIST PGMTEST,USER1
```

The following example lists all entries in the dispatch queue with the owner name USER2.

```
LIST ,USER2
```

The following example lists statistics on all jobs with owner name USER1 run on April 16, 1981 on any TSM device 20*xx*.

```
LIST ACCOUNT,OWNE=USER1, DATE=04/16/81,ORIG=TSM.20??
```

The following example displays statistics on all jobs with owner name USER2 and project number 120543.

```
LIST ACCOUNT,OWNE=USER2, PROJ=120543
```

2.33 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 with multivolume magnetic tape operations (for example, when mount message is displayed).
- Inhibit operator intervention — suppresses all prompts normally displayed on the system console
- Real time accounting — turns real-time accounting on or off

Syntax

MODE {**OFR**A | **ONR**A | **RCB**T | **RIB**P | **RIM**M | **RNO**P | **SCB**T | **SIB**T | **SIM**M | **SNO**P}

OFRA	turns real-time accounting off
ONRA	turns real-time accounting on
RCBT	resets continuous batch mode
RIBP	resets inhibit banner page
RIMM	resets inhibit mount messages
RNOP	resets inhibit operator intervention
SCBT	sets continuous batch mode
SIBP	sets inhibit banner page
SIMM	sets inhibit mount messages for disk and nonmultivolume magnetic tape operations
SNOP	sets inhibit operator intervention

2.34 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 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.

Examples

The following example stores the hexadecimal value `52535253` at physical location `3000`.

```
MODIFY 3000,52535253
```

The following example stores the hexadecimal value `52535253` at physical location `12000`.

```
MODI 12000,52535253,00000000
```

The following example changes the upper halfword of physical location `3004` to `5253`. The lower halfword is unchanged.

```
MODIFY 3004,52530000, 0000FFFF
```

The following example 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.

```
MODI 12004,00530055,FF00FFFF
```

2.35 MOUNT Directive

The MOUNT directive requests the physical mount of a volume to a device.

OPCOM uses the M.MOUNT service to perform the mount. The service interacts with the operator through the system console to complete the mount, unless the volume is currently mounted. If the NOMSG option is specified or the system-wide SNOP or OPCOM SIMM options are enabled, no operator interaction occurs.

Control returns to OPCOM after the physical mount completes. If the physical mount fails, the reason for failure displays to the terminal.

All messages generated by the MOUNT directive appear only on the terminal from which the MOUNT command was issued, even if the user is logged on to other terminals.

A public volume mount request is valid only from the system administrator.

Syntax

MOUNT *volname* **ON** *devmnc* [**OPT**IONS=[**NOM**SG] [**PUB**LIC]] [**SYS**ID=*id*]

volname is the 1- to 16-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.

[**OPT**IONS=[**NOM**SG] [**PUB**LIC]]

NOMSG specifies mount without operator interaction.

PUBLIC specifies the volume to be mounted is for public use (provided the caller has the SA attribute). The default is nonpublic.

[**SYS**ID=*id*]

id specifies a 3-character port identifier for a multiport volume; MP0, MP1...MPF. For compatibility, DP0 and DP1 can be used in place of MP0 and MP1.

Response

The following messages are issued to the system console:

```
MOUNT VOLUME volname ON DEVICE devmnc
REPLY R,H,A OR DEVICE:
```

To indicate that the volume and drive specified in the message are ready and to proceed with the mount, enter `R` (resume). To hold the mount, enter `H` (hold). To abort the mount, enter `A` (abort). To change the specified device, enter a new device mnemonic.

If the volume is mounted successfully, the following message is displayed on the operator's console:

```
VOLUME MOUNT SUCCESSFUL
```

At the same time the following message displays to the user terminal:

PHYSICAL MOUNT OF VOLUME *volname* COMPLETE

If an error occurs and the mount fails, one of the following error messages displays on the system console:

```
INVALID MOUNT DEVICE SPECIFIED
INVALID PORT ID SPECIFIED, ENTER 'DP0', 'DP1', OR 'MP0'...'MPF'
INVALID VOLUME NAME
J.MOUNT RUN REQUEST FAILED
MOUNT DEVICE NOT IN SYSTEM
MOUNT DEVICE UNAVAILABLE
MVT SPACE UNAVAILABLE
UNABLE TO ACTIVATE MULTIPROCESSOR RECOVERY PROGRAM
UNRECOVERABLE I/O ERROR TO VOLUME
VAT SPACE UNAVAILABLE
VOLUME ALREADY MOUNTED
```

WARNING - VOLUME SHOWS PORT DESIGNATOR MP(DP)*n* ALREADY
ALLOCATED... REPLY C TO CONTINUE OR A TO ABORT

The last message displays if the volume was not previously dismounted from the designated port and no volume clean-up was performed or if another port is currently mounted with the same port ID. 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 RD*num*
RD TYPE *num*
FILENAME IS *name*
SECTORS *num* THROUGH *num*

num	is a hexadecimal number
name	is the 1- to 16-character file name

If file overlap is detected in the DMAP/SMAP deallocation file descriptor area, the following message is displayed on the system console and the volume is not mounted.

```
J.MOUNT - ERROR - FILE OVERLAP HAS OCCURRED IN THE BAD SMAP
```

or

```
J.MOUNT - ERROR - FILE OVERLAP HAS OCCURRED IN THE BAD DMAP
```

To mount the disk, set control switch zero or seven.

Examples

The following example mounts the volume `ANEWVOL` on disk device `DM0800`. Mount messages display to the console unless the SYSGEN SNOP or the OPCOM SIMM option is enabled.

```
MOUNT ANEWVOL ON DM0800
```

2.36 OFFLINE Directive

The OFFLINE directive inhibits a specified device from all further allocation, inhibits using the IPU for task execution, or causes J.HLP to deallocate all help files. The OFFLINE directive is invalid for shadowed intelligent peripheral system (IPS) devices; use the J.SHDW task. The device, IPU, or help task are put back online with the ONLINE directive.

Syntax

OFFLINE {*devmnc* | **HELP** | **IPU**} [,**DEA**L]

devmnc	is the device mnemonic of the device to be taken offline (see Appendix A)
HELP	deallocates all help files allocated to J.HLP
IPU	inhibits all task execution on the IPU
[,**DEA**L]	indicates memory for the memory disk is deallocated after the device is marked offline. DEAL is ignored if *devmnc* does not specify a memory disk.

Response

For *devmnc*:

Any task that has assigned a specific device that is offline 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 HELP:

While help files are offline, online help is not available to users. Online help becomes available when placed online.

For IPU:

While the IPU is offline, 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.

Examples

The following example takes the line printer on channel `7A`, subaddress `00` offline.

```
OFFLINE LP7A
```

The following example disables task execution on the `IPU` configured in the system.

```
OFFLINE IPU
```

The following example takes the memory disk on channel `00`, subaddress `02` offline and deallocates its memory.

```
OFFLINE DM0002,DEAL
```

2.37 ONLINE Directive

The ONLINE directive makes a specified device available for allocation, makes the IPU available for task execution, or makes online help available for use.

Syntax

ONLINE {*devmnc* | **HELP** | **IPU**}

devmnc	is the device mnemonic of the device to be put online (see Appendix A)
HELP	allocates all help files to J.HLP
IPU	specifies that the IPU can be used for task execution

Examples

The following example places the line printer on channel `7A`, subaddress `00` online.

```
ONLINE LP7A
```

The following example enables the `IPU` configured in the system to be used for task execution and resumes IPU task scheduling.

```
ONLINE IPU
```

2.38 PURGEAC Directive

The PURGEAC directive deletes the current contents of the M.ACCNT accounting file.

Syntax

PURGEAC

2.39 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 {**D,***devmnc* | **J,***jobno* | **T,**{*jobname* | *taskname*} | *taskno*} [,*devmnc*]

D,*devmnc*	*devmnc* is the device where output is currently being processed
J,*jobno*	*jobno* is the job's sequence number
T,{*jobname* | *taskname*}	*jobname* is the job name as specified in the $JOB statement. *taskname* is the task name.
taskno	is the 8-digit task number assigned at activation
[,*devmnc*]	is the device where output is to be redirected. If not specified, output is directed to the SYSGEN-defined listed output device (LOD) or punched output device (POD).

Examples

The following example directs the SLO files for job number `700` to the line printer on channel `7E`, subaddress `F8`.

```
REDIRECT J,700,LP7EF8
```

The following example redirects the SBO files for job number `710` to the POD device.

```
REDIRECT J,710
```

The following example redirects the SLO file currently being output to the line printer on channel `7E`, subaddress `F9` to the LOD device.

```
RED D,LP7EF9
```

2.40 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 or the directive is ignored. Printing always begins at the beginning of the SLO file unless overridden by the PAGE parameter.

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.

Syntax

REPRINT [**D,***devmnc* | **J,***jobno* | **T,**{*jobname* | *taskname*} | *taskno*] [,**PAG**E, *start* [, *end*] | , *times*]

D, *devmnc*	*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.
J, *jobno*	*jobno* is the job number used to reprint all SLO files for a particular job
T,{*jobname* | *taskname*}	*jobname* is the job name as specified on the $JOB statement. *taskname* is the task name.
taskno	is the 8-digit task number assigned at activation
PAGE, *start* [, *end*]	specifies a page number on the current listing where reprinting should *start* and optionally *end*. If *end* is not specified, the default is EOF. 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. The default is one.

If no parameters are specified, the current SLO file being output to the SYSGEN-defined LOD is reprinted.

Examples

The following example resumes printing the SLO file on the LOD device on channel `7E`, subaddress `F8`, after interruption on another LOD device, beginning at page `5` and continues to the end-of file.

```
REPRINT D,LP7EF8,PAGE,5
```

The following example prints two additional copies of the SLO file currently being output to the LOD.

```
REPRINT,2
```

The following example reprints the SLO files for job `54` from the beginning of the job if the LOD malfunctions while printing the original copy.

```
REPRINT J,54
```

2.41 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 [**D,***devmnc* | **J,***jobno* | **T,**{*jobname* | *taskname*} | *taskno*]

D,*devmnc*	*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.
J,*jobno*	*jobno* is the job number used to repunch all SBO files for a particular job
T,{*jobname* | *taskname*}	*jobname* is the job name as specified in the $JOB statement. *taskname* is the task name.
taskno	is the 8-digit task number assigned at activation

If no parameters are specified, the current SBO file being output to the SYSGEN-defined POD is repunched.

Examples

The following example repunches the SBO file that was interrupted if the POD malfunctions in the middle of a job.

```
REPUNCH
```

The following example repunches the SBO files for job `54` from the beginning of the job if the POD malfunctions while processing that job.

```
REPUNCH J,54
```

2.42 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 2-character hexadecimal interrupt priority level

2.43 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*
taskname is the task name. If *taskname* is used to resume a task, the task must be a unique copy.

taskno	is the task number assigned at activation

Examples

The following example resumes the task `PGMTEST` if it is a unique copy.

```
RESUME T,PGMTEST
```

The following example resumes task number `02000001`.

```
RESUME 02000001
```

2.44 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 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. The default is a mask of binary ones.

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

Examples

The following example displays all words between physical locations `3000` and `3FFF` with the value `52535253` and their locations.

```
SEARCH 3000,3FFF,52535253
```

The following example displays all words between physical locations `12000` and `12FFF` with the value `52535253` and their locations.

```
SEARCH 12000,12FFF,52535253,FFFFFFFF
```

The following example displays all words between physical locations `12000` and `12FFF` with the value `53` in byte one and their locations.

```
SEARCH 12000,12FFF,00530000,00FF0000
```

2.45 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*
: *taskname* is the task name. If *taskname* sends a message to a task, the receiving task must be a unique copy.

taskno
: *taskno* is the task number assigned at activation

[, *message*]
: is the message to be sent to the specified task. If *message* is entered on the same line as the directive, no response is given. If more space is needed for a message, omit *message* from the directive line and enter a return. The following prompt will be displayed:

```
"ENTER MESSAGE"
??
```

Type the message in response to the `??` prompt, followed by a return.

Examples

The following example sends the message `THIS IS A TEST` to the receiver buffer address of the task `PGMTEST` if the task is a unique copy and has a message receiver.

```
??SEND T, PGMTEST
"ENTER MESSAGE"
??THIS IS A TEST
```

The following example sends the message `THIS IS A TEST` to the receiver buffer address of task number `02000001` if the task has a message receiver. The message is short enough to be included on the directive line.

```
SEND 02000001,THIS IS A TEST
```

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 in the following example.

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
```

Users permitted to access tasks with different owner names can also use this capability.

2.46 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* | **RQI**, *intlevel* | **RST**, *taskno*}

timer	is the 2-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. *t2* will reset the timer to resume a task that was activated at *t1* if the ACP parameter is specified.

ACP,*filename*
indicates the time-out event is the activation of the specified task. *filename* is the name of the system file containing the task.

RQI, *intlevel*
indicates the time-out event is the execution of a request interrupt for the specified interrupt level. *intlevel* is the 2-character hexadecimal interrupt priority level.

RST, *taskno*
indicates the time-out event is the resumption of the specified task. *taskno* is the 8-digit task number assigned at activation.

Examples

The following example activates the file PGMTEST when timer AB expires. The time out is set to occur only once.

```
SETTIMER AB,2,0,ACP,PGMTEST
```

The following example resumes task number 02000001 when timer AB expires. Timer AB is reset to 3 upon each following time event.

```
SETTIMER AB,2,3,RST,02000001
```

The following example issues an RI instruction for interrupt level 2F when timer AB expires. The time out is set to occur only once.

```
SET AB,2,0,RQI,2F
```

2.47 SNAP Directive

The SNAP directive dumps word locations specified by starting and ending physical addresses to the requesting terminal.

The task debuggers 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

Examples

The following example displays the contents between physical locations `3000` and `3FFF`.

```
SNA 3000,3FFF
```

2.48 STATUS Directive

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 IPU directive returns information about the IPU.

The STATUS M directive displays a current memory utilization map.

The STATUS T, *taskname* and STATUS *taskno* directives return the current status of a task and its significant attributes.

The STATUS VOLU directive returns information about 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 status are displayed.

Syntax

STATUS {**CHAN,** *address* | **CONT,** *devmnc* | **D,** *devmnc* | **IPU** | **M**[, *class*] | **T,** *taskname* | *taskno* | **VOLU**[,*volname*]}

CHAN, *address*	*address* is a 2-character hexadecimal channel number or GPMC device address from 80 to FF
CONT, *devmnc*	*devmnc* is a device mnemonic for the device controller (see Appendix A)
D, *devmnc*	*devmnc* is a device mnemonic for the device
IPU	returns online or offline status and notes which task, if any, is running
M[, *class*]	*class* is E (below 128KW), H (above 128KW), or S (slow speed above 128KW) corresponding to the classes of memory. When *class* is specified, the display is limited to only the specified class. If *class* is not specified, physical address boundaries and other information for each memory class is displayed.

T, *taskname*
: *taskname* is the task name

taskno is the task number assigned at activation

VOLU[,*volname*]
: *volname* is the 1- to 16-character name of a mounted volume where status is to be displayed. If *volname* is not specified, the status of all mounted volumes is displayed.

Response

To STATUS CHAN,*address*:

chaddr numcont `CONTROLLERS  CHANNEL [NOT] INITIALIZED`

devchan `CONTROLLER` *addr* `UNITS` *units* [`I/O` *num*] [`CONTR'L MALF`] [`CONTROLLER [NOT] INITIALIZED`]

(the two controller lines are repeated for each controller assigned to the channel)

DEV *devaddr* {`ON|OFF`}{`ALOC|FREE|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

addr is the controller address

units is the number of units on the controller

num 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. This message is displayed only for class F devices.

`CONTROLLER NOT INITIALIZED`	is displayed if extended I/O initialization has not been performed for this controller. This message is displayed only for class F devices.
devaddr	is the device address and device type
`ON`	indicates the device is online
`OFF`	indicates the device is offline
`ALOC`	indicates the device is allocated
`FREE`	indicates the device is free
`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 allocated to the device. 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 CONT,*devmnc*:

devchan `CONTROLLER` *addr* `UNITS` *units* [I/O *num*] [`CONTR'L MALF`]
    [`CONTROLLER` [`NOT`] `INITIALIZED`]

DEV *devaddr* {`ON`|`OFF`} {`ALOC`|`FREE`|`SHAR`|`TSM`} [`IOQ` *devqueue*] [`DEV MALF`] [*taskno*] [*dest*]

(the device line is repeated for each device on the controller)

devchan	is the device type and channel address for the controller
addr	is the controller address
units	is the number of units on the controller

num	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. This message is displayed only for class F devices.
`CONTROLLER NOT INITIALIZED`	is displayed if extended I/O initialization is not performed for this controller. This message is displayed only for class F devices.
devaddr	is the device address and device type
`ON`	indicates the device is online
`OFF`	indicates the device is offline
`ALOC`	indicates the device is allocated
`FREE`	indicates the device is free
`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 allocated to the device. 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*:

DEV *devaddr* {ON | OFF} {ALOC | FREE | SHAR | TSM} [IOQ *num*] [DEV MALF] [*taskno*] [*dest*]

devaddr	is the device address and the device type
ON	indicates the device is online
OFF	indicates the device is offline
ALOC	indicates the device is allocated
FREE	indicates the device is free
SHAR	indicates the device is shared
TSM	indicates the device is in use by TSM
num	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 allocated to the device. 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 IPU:

```
    {ON|OFF} TASK=taskname OWNER=ownername TASK#=taskno
```

ON	indicates the IPU is online
OFF	indicates the IPU is offline
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 M[*,class*]:

```
    MEM low -- high {ALOC|FREE|SHAR} class [MALFUNC] [NONPRES]
```

low	is the low physical memory address boundary
high	is the high physical memory address boundary
ALOC	indicates allocated memory
FREE	indicates free memory
SHAR	indicates shared memory
class	is one of the following: E memory below 128KW H high speed memory above 128KW S slow speed memory above 128KW
MALFUNC	indicates malfunctioning memory. This field is blank if memory is operational.
NONPRES	indicates nonpresent memory. This field is blank if the specified memory is configured.

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=*message*]

taskname:
    message

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 is one of the following:

RT	real time, priority 1-54
TD	time distribution, priority 55-64

origin is one of the following:

B	batch
T	terminal
ƀ	independent

class is one of the following:

E	memory below 128KW
H	high speed memory above 128KW
S	slow speed memory above 128KW

mapblocks is the number of 2K swappable, nonshared map blocks used by the task

priv is one of the following:

U	unprivileged
P	privileged

res is one of the following:

N	nonresident
R	resident

state is a 4-character identifier corresponding to the state of the task as follows:

CURR	current task in execution
SQRT	real-time priority - ready-to-run
SQ55-SQ64	time distribution levels - ready-to-run
CIPU	current IPU task in execution
RIPU	requesting IPU task
SWTI	wait interactive

SWIO	waiting for wait I/O
SWSM	wait, sending message
SWSR	wait, sending run request
SWLO	wait, low speed output
SUSP	wait, suspended for message interrupt, time out, or resume
RUNW	wait for wait run request or time out
HOLD	wait, operator hold
ANYW	wait for any no-wait I/O, no-wait run request, or any message interrupt or break
SWDC	wait, disk space
SWDV	wait, peripheral
MRQ	wait, memory
SWMP	wait, memory pool
PREA	preactivation phase
FREE	dispatch queue available for allocation by a task
SWGQ	general wait queue - resourcemark

NWIO=*requests*
requests is the number of no-wait I/O requests outstanding

RRCT=*requests*
requests is the number of run receiver requests outstanding

MRCT=*requests*
requests is the number of message receiver requests outstanding

DEV=*dev* *dev* is the device. This is displayed if the state is SWDV or SWDC. A channel and subaddress of zero means any device of the specified type.

GQID=*message*
indicates the task is in the general wait queue. This is displayed if the state is SWGQ. *message* indicates the reason the task is waiting in the general wait queue. 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
QUEUED FOR SUSP DUAL-PORT LOCK
QUEUED FOR DEBUG WAIT
QUEUED FOR REMOTE MESSAGE AREA
QUEUED FOR REMOTE MESSAGE EVENT
QUEUED FOR REMOTE ALLOCATE AREA
QUEUED FOR REMOTE DEALLOCATE AREA
QUEUED FOR REMOTE ABORT AREA
QUEUED FOR REMOTE ON/OFF AREA
```

message is any of the following messages describing characteristics of the task:

`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

To STATUS VOLU[, *volname*]:

volname devmnc [*users*] *total avail*
mounted
volmsg

volname	is the 1- to 16-character left-justified, blank-filled name of the mounted volume
devmnc	is the device the volume is mounted on (see Appendix A)
users	is the number of current users for a nonpublic volume, or the number of resources allocated for a public volume, if a physical dismount is pending
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 consists of 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)
```

Examples

The following example displays the current status of volume `TBRD`.

```
STATUS VOLU,TBRD
```

The following example displays the current memory utilization map for `E` class memory.

```
STATUS M,E
```

The following example displays the status of channel `7E`, the controllers assigned to the channel, and all the devices assigned to the controllers.

```
STATUS CHAN,7E
```

The following example displays the status of the line printer device on channel `7E`, subaddress `F8`.

```
STATUS D,LP7EF8
```

The following example displays the current status of all tasks named `PGMTEST`.

```
STATUS T,PGMTEST
```

The following example displays the status of task number `02000001`.

```
STATUS 02000001
```

The following example displays the status of controller `0` assigned to `DM08` and all devices assigned to the controller.

```
STATUS CONT,DM0800
```

2.49 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 {{**ON**|**OFF**},*devmnc* ,{**L**|**P**}, {**R**|**B**} | **SID**,*devmnc* [,*density*,*parity*]}

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
SID,*devmnc*	*devmnc* is the device to be assigned as the system input device
density	is H (high density) or L (low density) if *devmnc* is a 7-track magnetic tape. Otherwise, omit this parameter.
parity	is E (even parity) or O (odd parity) if *devmnc* is a 7-track magnetic tape. Otherwise, omit this parameter.

Examples

The following example makes the line printer on channel `7E`, subaddress `00` available for automatic selection for spooled real-time SLO output.

```
SYSASSIGN ON,LP7E,L,R
```

The following example changes the default SID to the card reader on channel `78`, subaddress `01`.

```
SYSASSIGN SID,CR7801
```

2.50 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* | *dd-mo-yy* | *ddmonyy*} *hh:mm:ss*

mo	is the 2-digit decimal month
mon	is the 3-character ASCII month abbreviation
dd	is the 2-digit decimal day
yy	is the 2-digit decimal year
hh	is the 2-digit decimal hour (24-hour time)
mm	is the 2-digit decimal minute
ss	is the 2-digit decimal second

Examples

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
```

2.51 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 2-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 *time*. The task is reactivated at each time out. If *reset* is not specified, the comma denoting the field must still be specified, and the task is activated or resumed only once. (**Note:** Task must be in a suspended state in order for *reset* to have any effect.)
timerid	is the 2-character ASCII name of the timer that is created

2.52 UNLOCK Directive

The UNLOCK directive recovers multiprocessor resources when a system or disk 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 offline 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.

Note: 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)
```

The last message is displayed when the port ID is in use with any mounted volume on this processor.

Note: The operator should verify that the resource is not in use on any other processor.

2.53 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. A return must be entered 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

Examples

```
  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 { `$JOB` | `$EOF` } *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.

3 Volume Manager (VOLMGR)

3.1 Introduction

The Volume Manager (VOLMGR) creates or deletes permanent disk 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 interactive or batch mode. In interactive mode, VOLMGR accepts directives from either a terminal or 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 disk.

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 (EOT) (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 are a multiple of the disk block size (192 words) since saves to unformatted disk devices are possible. The smallest record is one block, the largest record is 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 disk 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 disk I/O error (for example, no data transfer or a partial transfer) terminates the save and writes an appropriate message. If any tape I/O error occurs during a save, VOLMGR backspaces the tape to the end of the last saved file, sets up a special resource descriptor tape record (RDTR), builds a logical EOF, and exits with a VO06 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 disk are 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.

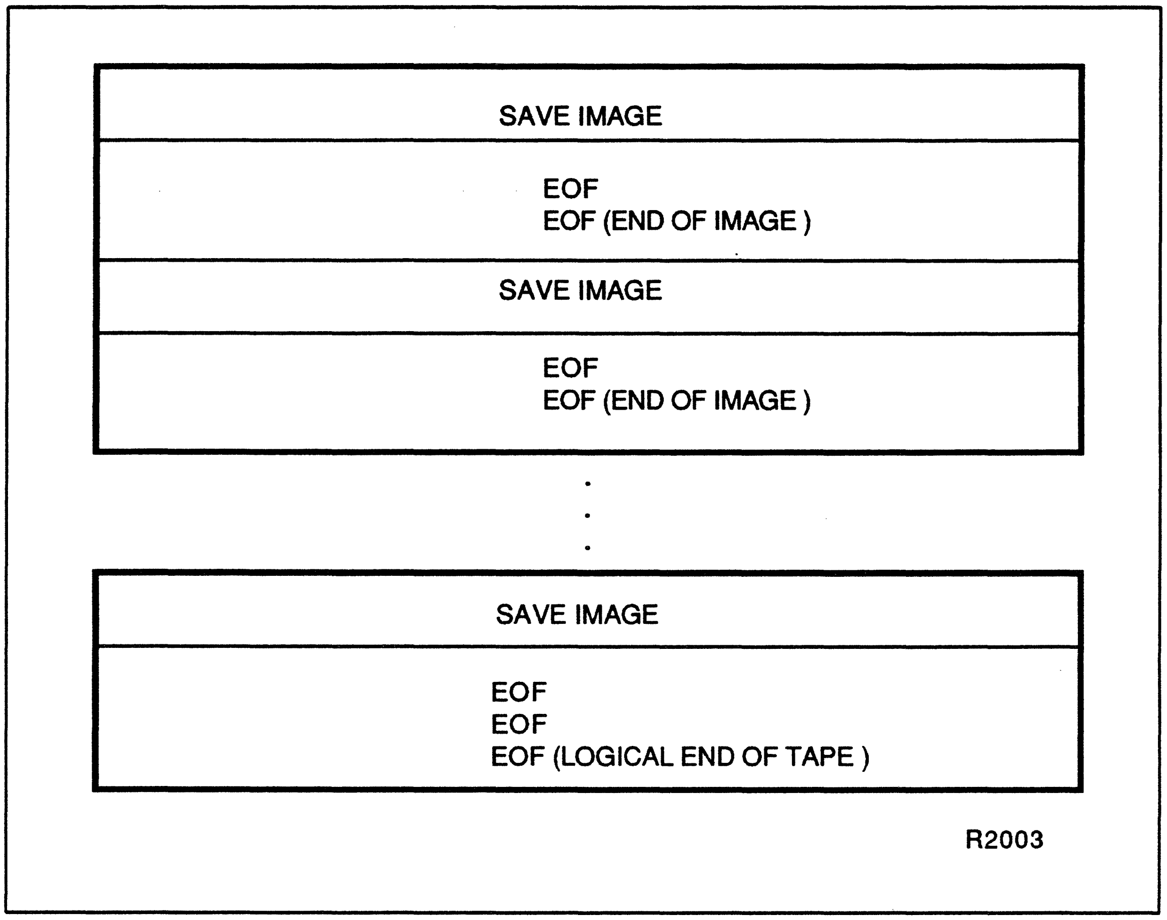


Figure 3-1
Save Tape Structure

Files are restored/copied to temporary disk 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, an 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.

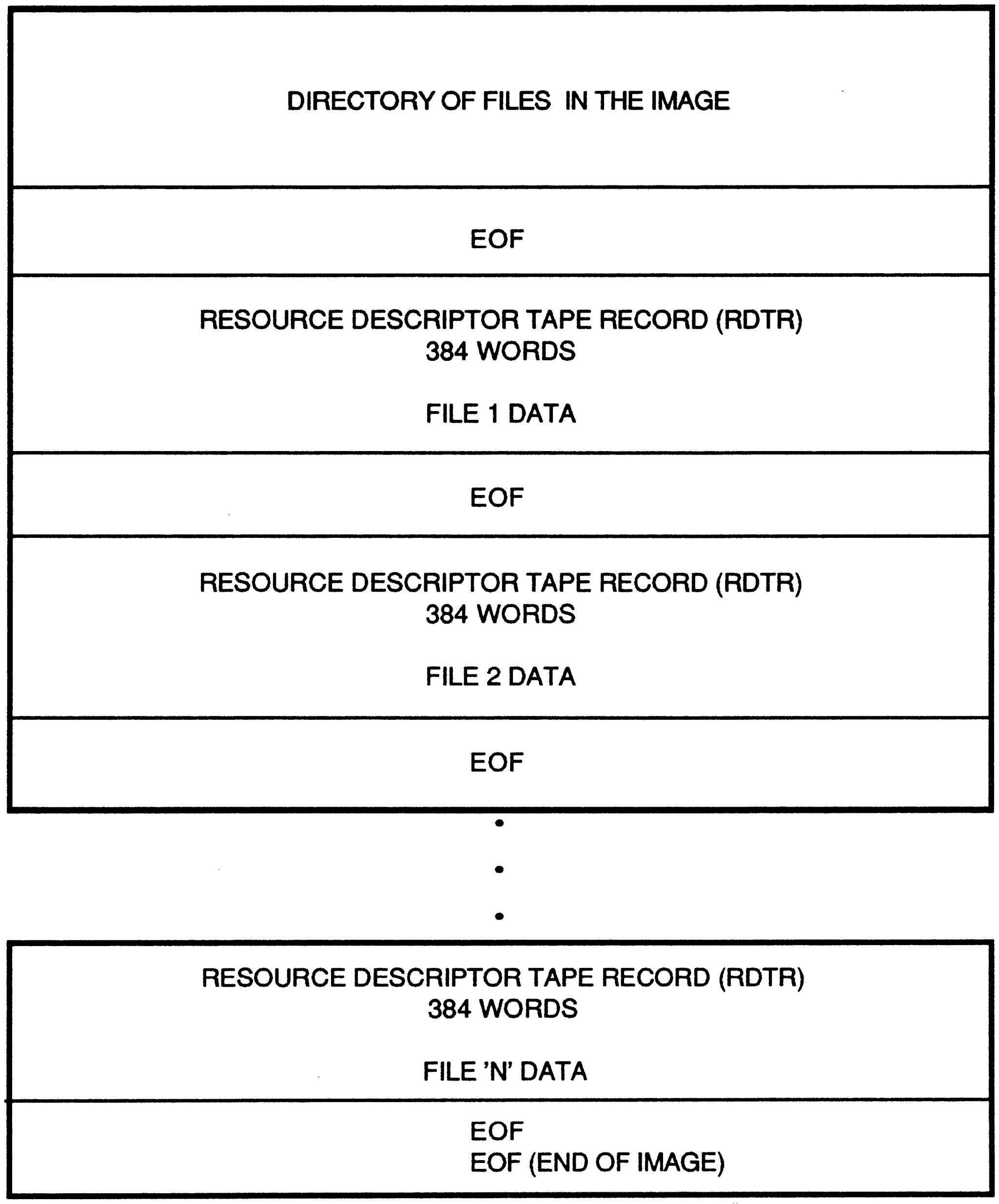

DIRECTORY OF FILES IN THE IMAGE
EOF
RESOURCE DESCRIPTOR TAPE RECORD (RDTR)
384 WORDS
FILE 1 DATA
EOF
RESOURCE DESCRIPTOR TAPE RECORD (RDTR)
384 WORDS
FILE 2 DATA
EOF
RESOURCE DESCRIPTOR TAPE RECORD (RDTR)
384 WORDS
FILE 'N' DATA
EOF
EOF (END OF IMAGE)
R2004

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). A copy of this directory is maintained on a temporary disk file for the actual file lookup and data transfer phases of the save operation. 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 file may only appear once in a save image. Multiple references to a file in a single SAVE directive are ignored.

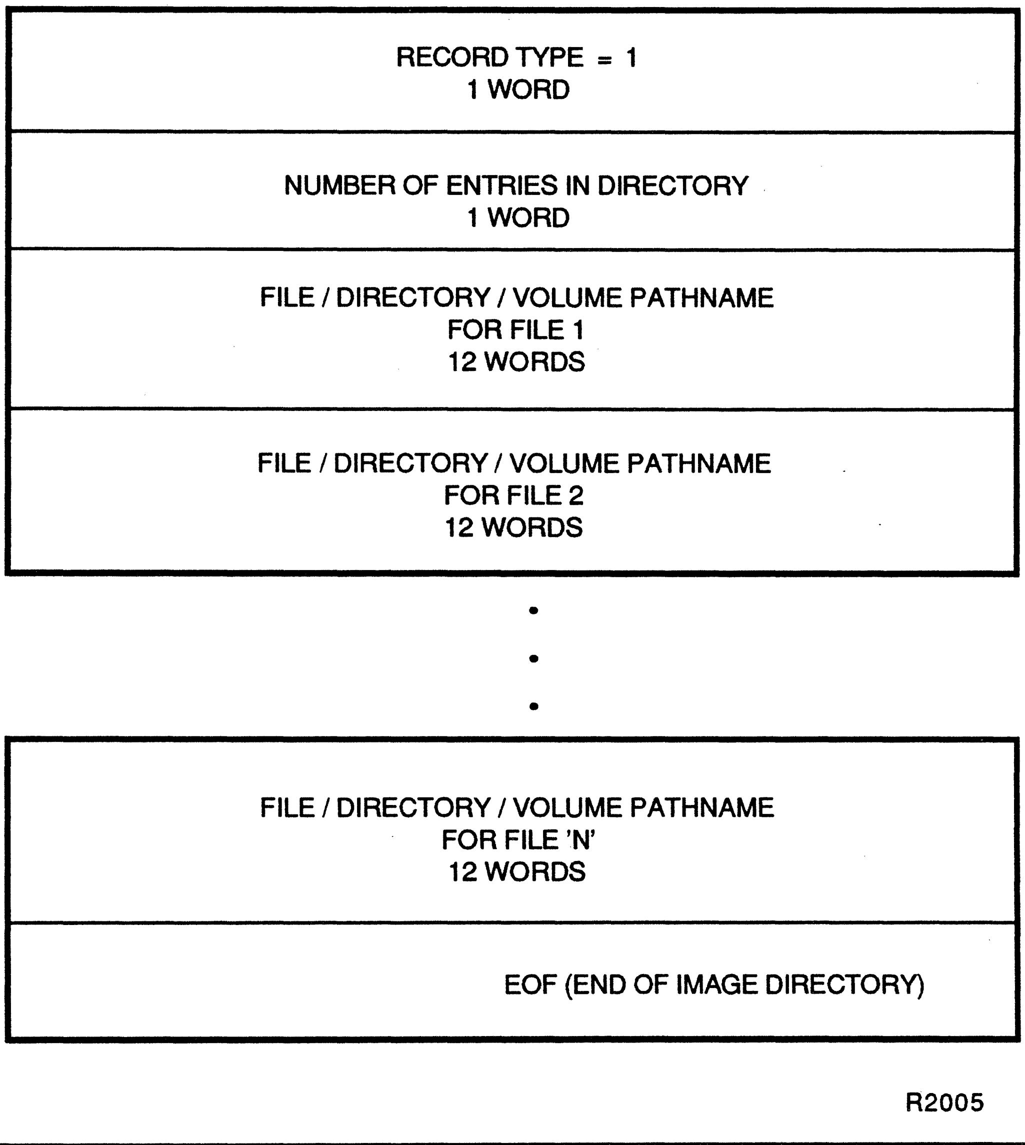

RECORD TYPE = 1
1 WORD
NUMBER OF ENTRIES IN DIRECTORY
1 WORD
FILE / DIRECTORY / VOLUME PATHNAME
FOR FILE 1
12 WORDS
FILE / DIRECTORY / VOLUME PATHNAME
FOR FILE 2
12 WORDS
FILE / DIRECTORY / VOLUME PATHNAME
FOR FILE 'N'
12 WORDS
EOF (END OF IMAGE DIRECTORY)
R2005

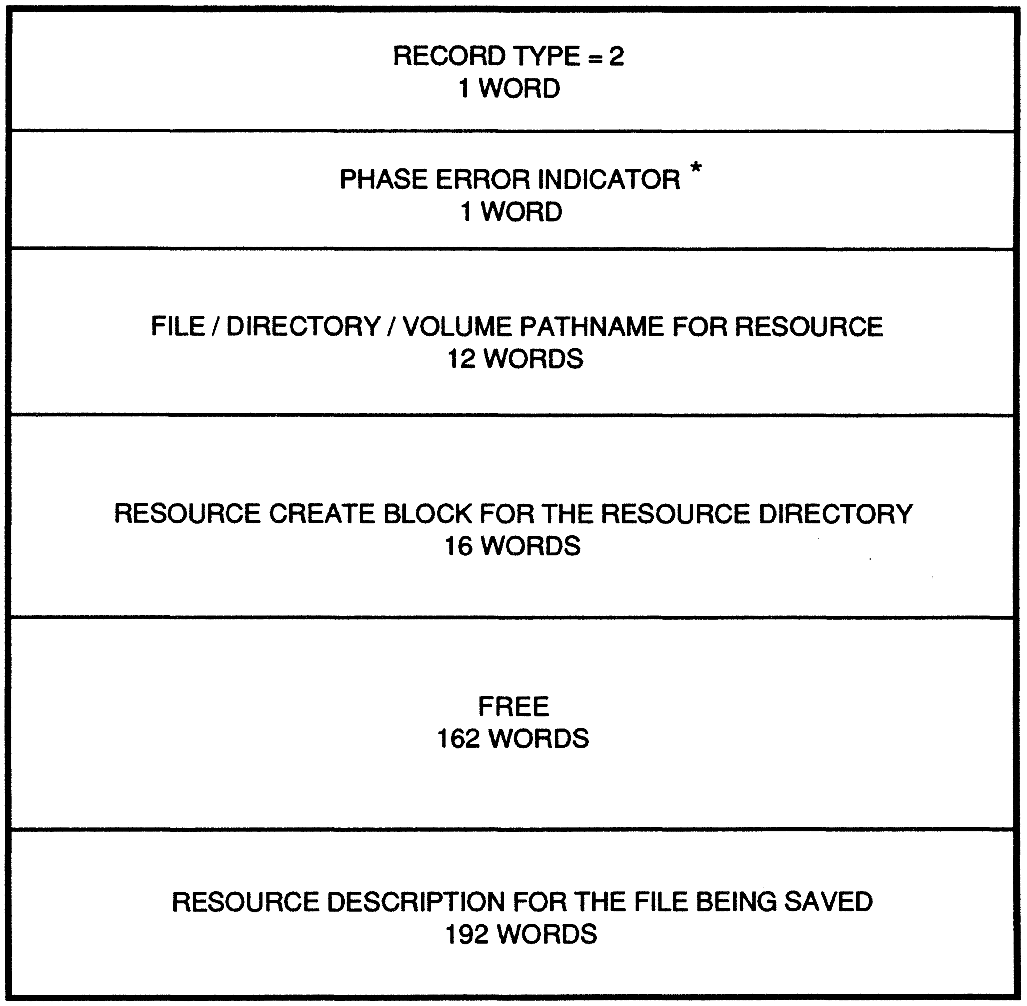
Figure 3-3
Save Image Directory Structure

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 disk. 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 disk volume. Figure 3-4 illustrates the format of the RDTR.


RECORD TYPE = 2
1 WORD
PHASE ERROR INDICATOR *
1 WORD
FILE / DIRECTORY / VOLUME PATHNAME FOR RESOURCE
12 WORDS
RESOURCE CREATE BLOCK FOR THE RESOURCE DIRECTORY
16 WORDS
FREE
162 WORDS
RESOURCE DESCRIPTION FOR THE FILE BEING SAVED
192 WORDS

* Bit 0 set, no resource descriptor or file data on the tape
Bit 1 set, no file data on the tape
Bit 2 set, tape I/O error occurred during a save operation

R2006

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. Minimum valid abbreviations are shown in boldface.

Directive	Description
BACKSPACE FILE	rewinds the tape a specified number of EOF marks
BACKSPACE IMAGE	rewinds the tape a specified number of save images
CLEAR	clears default option values established with the SET directive
CONVERT	restores post-RTM 6.0 and all MPX-32 File Manager save tapes to MPX-32 Revision 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 disk
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 space allocated to an existing file
HELP	displays a short description of each VOLMGR directive
LOG FILE	produces a listing of specified disk 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 a file
RESTORE DIRECTORY	restores files from tape to disk
RESTORE POSITION	restores one file from tape to disk
REWIND	rewinds a tape to its beginning
SAVE	saves files to tape
SAVE INCREMENTAL	saves files created or modified since the last save was performed

Continued on next page

Directive	Description
SDT	produces a system distribution tape (SDT)
SDT MASTER	produces a master system distribution tape
SET	establishes global default option values
SKIP END1	advances the tape to its logical end-of-tape
SKIP FILE	advances the tape a specified number of EOF marks
SKIP IMAGE	advances the tape a specified number of save images
TRUNCATE	decreases the amount of space allocated to an existing file

3.5.1 BRIEF Option

VOLMGR directives that create or delete files, directories, or memory partitions have the BRIEF option to control the amount and type of output that the operation generates. This section describes the output formats 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.

3.5.1.1 BRIEF Option with Files

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 executed
volname	is the volume affected by the directive
dirname	is the directory affected by the directive
filename	is the file 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 ACCESSED BY:                ON:
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 executed
pathname	is the volume, directory, and file name 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: ON:	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. Y specifies blocked. N specifies 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 ACCESSED BY: ON:	specifies the ownername which last accessed the file and lists the date and time it was accessed. Saves to tape are not recorded in this field.
LAST CHANGED BY: ON:	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 from 0 to FF
SHARE=	specifies if the file can be shared

SAVE=	specifies if the file can be saved. If SAVE=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. N specifies that end-of-file (EOF) equals end-of-medium. This attribute is required to be false (N or F) 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 disk I/Os.
ZERO=	specifies if the file is to be zeroed on creation and extension
SIZE=	specifies the size of the file in blocks
EXTEND=	specifies if the file is extendible

AUTO	specifies that the file is only automatically extendible
MANU	specifies that the file is only manually extendible
BOTH	specifies that the file is both automatically and manually extendible
NO	specifies that the file is not extendible.

MAXINC=	specifies the maximum number of blocks the file can be extended
MININC=	specifies the minimum number of blocks the file can be extended
MAXSIZE=	specifies the maximum block size of the file. Zero specifies the size is limited only by available disk 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.

3.5.1.2 BRIEF Option with Directories

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 executed
volname	is the volume affected by the directive
dirname	is the directory affected by the directive
size	is the size of the directory

- For BRIEF=N or F:

```
directive pathname
RID=
CREATED BY:          ON:
LAST ACCESSED 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 executed

pathname is the volume and directory 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:    ON:`
specifies the owner name under which the directory was created and the date and time it was created

`LAST ACCESSED BY:    ON:`
specifies the ownername which last accessed the directory and the date and time it was accessed. Saves to tape are not recorded in this field.

`LAST CHANGED BY:    ON:`
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 from 0 to FF

`SHARE=` specifies if the directory can be shared

`EOFM=`	specifies the status of EOF management. N specifies that EOF equals end-of-medium. Directories are automatically created when this attribute is false (N or F).
`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.

3.5.1.3 BRIEF Option with Memory Partitions

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 executed
volname	is the volume affected by the directive
dirname	is the directory affected by the directive
partname	is the name of the partition 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 affected by the directive

`RID=` is the resource identifier which specifies the volume name, binary date, binary time of creation, resource descriptor address, and resource type

`CREATED BY:   ON:`
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 from 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 can be accessed in batch or interactive mode in the following ways:

```
[$] [EXECUTE] VOLMGR
```

$RUN VOLMGR is valid 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) on exit.

From TSM, by entering VOLMGR and a VOLMGR directive on the same line, control returns to TSM after the VOLMGR directive is executed and its output is displayed. If this is done in 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 specifying initial block allocation size, the number specified is rounded up to the nearest allocation unit. The allocation unit is determined by the storage capacity of the disk. On an 80MB disk, the minimum allocation size is 2 blocks. On a 300MB disk, the minimum allocation size is 4 blocks. Therefore, if the size specified is SIZE = 1, the rounded up result would be 2 on an 80MB disk and 4 on a 300MB disk.

Similarly, when using the CREATE DIRECTORY directive and specifying the maximum number of files to be listed within the directory, the number specified is rounded up to the nearest allocation unit (maximum of 12 entries per block on all disk types). Therefore, if the number of entries specified is ENTRIES = 10, the rounded up result would be 24 on an 80MB disk (12 per block times 2 blocks) and 48 on a 300MB disk (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 disk 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 identifies the path 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 characters can be used in the pathname by specifying a question mark to replace individual characters or an asterisk to replace a string of characters. The special meaning given to the wild card characters 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 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. Enclosing a file name in quotes 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 the following 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 LFC SYC. The default assignment for SYC is:

```
$AS 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 LFC SLO. The default assignment for SLO is to LFC 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 file to contain VOLMGR listed output. This file must already exist.
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 LFC 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.

LFC TAP is assigned as follows:

```
$ASSIGN TAP TO DEV=devmnc BLO=N
```

devmnc	is the device mnemonic of the device on which to perform the save or restore operation

3.10.4 Wild Card Characters

When a wild card character (*) is specified in any directive, the wild card is converted to actual volume/directory/file names by using temporary files. The LFCs 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 disk 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, noun, 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 can be a comment by placing an exclamation point as the first character of the line.

3.12.1 Verbs and Nouns

The verb is the first item on the directive line. When a verb can perform more than one function, a noun specifies which function to perform. For example:

```
CREATE COMMON
CREATE DIRECTORY
CREATE FILE
```

`CREATE` is the verb and `COMMON`, `DIRECTORY`, and `FILE` are nouns.

Default nouns 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 (noun 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). T and Y are synonymous. F and N 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 command verb (noun 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 FILE1
```

`SIZE=100` is the global option and `FILE1` is the parameter.

To create more than one file of the same size, enter:

```
CREATE FILE SIZE=100 FILE1,FILE2,FILE3
```

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 FILE1 SIZE=50,FILE2,FILE3
```

`SIZE=50` is a local option and `FILE2` and `FILE3` 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 FILE1,FILE2,FILE3
```

All three files are created as `50` blocks in size.

In the following example:

```
SET SIZE=50
CREATE FILE FILE1 SIZE=100,FILE2,FILE3 SIZE=150
```

`FILE1` is created as `100` blocks in size, `FILE2` is created with the default size of 50 blocks, and `FILE3` is created as `150` blocks in size.

3.12.3.3 Time Options

Some VOLMGR directives have an optional time specification. The following rules apply to time specification usage.

- Because 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 (ACCESSED, CREATED, CHANGED, SAVED, or RESTORED) can be specified per partition or pathname.
- Regardless of the date format used in the directive, the date will display in the format the date was entered at IPL.
- Format options for date are:

  mo/dd/yy
  dd-mon-yy

mo	is the 1- or 2-digit decimal month
mon	is the 3-character ASCII month abbreviation
dd	is the 1- or 2-digit decimal day
yy	is the 2-digit decimal year

- Format options for time are:

  hh:mm:ss
  hh:mm
  h

hh	is the 1- or 2-digit decimal hour (24-hour time)
mm	is the 1- or 2-digit decimal minute
ss	is the 1- or 2-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.
- 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 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 a name, keyword, or pathname must appear on a single input line.

3.14 BACKSPACE FILE Directive

The BACKSPACE FILE directive backspaces a magnetic tape a specified number of EOF marks.

Syntax

BACKSPACE **F**ILE [[**FIL**ES=]*eofs*]

[[**FIL**ES=]*eofs*]
: *eofs* is the number of EOF marks to backspace. If not specified, the tape is backspaced one EOF mark.

Examples

The following example backspaces the tape one EOF mark.

```
BAC F
```

Each of the following examples backspace the tape two EOF marks.

```
BAC F FIL=2
BAC F 2
```

3.15 BACKSPACE IMAGE Directive

The BACKSPACE IMAGE directive backspaces a tape a specified number of save images.

Syntax

BACKSPACE [**I**MAGE] [[**IMA**GES=]*images*]

[[**IMA**GES=]*images*]
: *images* is the number of save images to backspace. If not specified, the tape is backspaced one save image.

Examples

The following example backspaces the tape one save image.

```
BAC
```

Each of the following examples backspace the tape two save images.

```
BAC IMA=2
BAC I 2
BAC 2
```

3.16 CLEAR Directive

The CLEAR directive clears all default option values previously established with 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 Revision 2.*x* volume/directory structure format. Wild card characters cannot be used with this directive.

The device where the data to convert resides must be specified by a static assignment in JCL. This directive assumes the user assigned LFC TAP before invoking VOLMGR.

Syntax

CONVERT [[**VOL**UME=]*volname*] [**BRI**EF={**Y**|**N**}]

[[**VOL**UME=]*volname*]
: *volname* is 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.

[**BRI**EF={**Y**|**N**}]
: specifies whether to display complete file status. N displays complete file status. Y displays the file name and size only. The default is Y.

Notes:

Data in the original 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.

   A pre-MPX-32 Revision 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 Revision 2.0 files with the unsupported File Manager (FILEMGR) utility.
2. Directory name – the directory name is the same as the original user name. Directories are created if needed. However, if 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 with 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 with 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 others.

Examples

The following example converts all files in the save image and stores them on volume `TEST`. Their complete file status is displayed.

```
CON VOL=TEST BRI=N
```

The following example converts all files in the save image and stores them on the user's current working volume. Only the file names and sizes are displayed.

```
CON
```

3.18 COPY Directive

The COPY directive creates a file by copying an existing file. A file can be copied to the same volume it is currently located on or to a different volume. 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**=]*sourcefile* [**TO**[=]]*targetfile*
[**ACC**ESS=*usertype*([**R**EAD] [**W**RITE] [**U**PDATE] [**M**ODIFY] [**A**PPEND] [**D**ELETE])]...
[**AUTO**EXT={Y|N}] [**BRI**EF={Y|N}] [**CON**TIGUOUS={Y|N}] [**EOFM**={Y|N}]
[**FAS**T={Y|N}] [**MAN**EXT={Y|N}] [**MAXI**NC=*maxblocks*] [**MAXS**IZE=*totblocks*]
[**MIN**INC=*minblocks*] [**NOS**AVE={Y|N}] [**OWN**ER=*ownername*]
[**PRO**JECTGROUP=*projname*] [**REP**LACE={Y|N}] [**SHA**RED={Y|N}]
[**SI**ZE=*blocks*] [**STA**RT=*blocknum*] [**ZER**O={Y|N}]

[**FRO**M=]*sourcefile*
sourcefile is the pathname of the file to copy

[**TO**[=]]*targetfile*
targetfile is the pathname of the file to create

[**ACC**ESS=*usertype*([**R**EAD] [**W**RITE] [**U**PDATE] [**M**ODIFY] [**A**PPEND] [**D**ELETE])]
specifies a type of user and the kinds of access to *targetfile* that *usertype* is allowed. Repeat this option for each *usertype* desired. If not specified, owner and project group have complete access and others have read only access.

usertype	is one of the following types of users:
	OWNER owner of *targetfile*
	PROJECTGROUP members of *targetfile*'s project group
	OTHERS all other users
READ	permits *usertype* to read the file
WRITE	permits *usertype* to write to the file
UPDATE	permits *usertype* to update the file
MODIFY	permits *usertype* to modify the file
APPEND	permits *usertype* to append to the file
DELETE	permits *usertype* to delete the file

[**AUTO**EXT={Y|N}]
specifies if *targetfile* can be automatically extended. The default is Y.

[**BRI**EF={Y|N}]
specifies whether to display complete file status. N displays complete file status. Y displays only *targetfile*'s name and size. The default is Y.

[CONTIGUOUS={Y|N}]
: specifies if extensions to *targetfile* are to be contiguous when possible. Initial allocation of a file is always contiguous. The default is N.

[EOFM=Y|N}]
: specifies the status of EOF management. If not specified, the system default is used.

[FAST={Y|N}]
: specifies whether to retain the file's resource identifier when the file is copied or restored with VOLMGR. If omitted, the file is copied to any available resource identifier.

[MANEXT={Y|N}]
: specifies if *targetfile* can be manually extended. The default is Y.

[MAXINC=*maxblocks*]
: *maxblocks* is the size of each subsequent automatic increment of the file. The default is 64 blocks.

[MAXSIZE=*totblocks*]
: *totblocks* is the maximum size for an extendible file. If not specified, size is limited only by available disk space.

[MININC=*minblocks*]
: *minblocks* is the minimum acceptable automatic increment size if the MAXINC value cannot be obtained. The default is 32 blocks.

[NOSAVE={Y|N}]
: 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.

[OWNER=*ownername*]
: *ownername* is the owner name to associate with *targetfile*. If not specified, defaults to the owner name associated with the VOLMGR task.

[PROJECTGROUP=*projname*]
: *projname* is the project group name to associate with *targetfile*. If not specified, defaults to the project group associated with the VOLMGR task.

[REPLACE={Y|N}]
: If the file name *targetfile* 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={Y|N}]
: specifies if the file can be used in the shared mode. The default is Y.

[SIZE=*blocks*]
: *blocks* is the initial block allocation size of *targetfile*. If not specified, defaults to the size of *sourcefile*.

[START=*blocknum*]
: *blocknum* is the starting block number of *targetfile*. If *sourcefile* cannot be copied to the specified block number, a denial is given. If not specified, *sourcefile* is copied to any available space on the volume.

[ZERO={Y|N}]
: specifies if *targetfile* should be zeroed when created and extended. The default is N.

The *targetfile* is always (re)created. If *targetfile* already exists, the REPLACE=Y option must be specified for the copy to complete. When REPLACE=Y, VOLMGR attempts to delete the existing *targetfile*. If this delete is unsuccessful, a copy does not take place.

Targetfile's attributes are implicitly the same as *sourcefile*'s attributes, except the owner name and project group name are those associated with the VOLMGR task. Also, *targetfile*'s access rights are the system default. Any of *targetfile*'s attributes can be explicitly stated on the directive line to override the implicit values.

Examples

The following example copies the file `INDEX` from directory `DIR1` to file `INDEX2` in directory `DIR2` on the same volume. `INDEX2`'s project group name is `011657` and any existing version of `INDEX2` is replaced. `USER1` owns `INDEX2` and has complete access privileges.

```
COP FRO=@TEST(DIR1)INDEX TO=@TEST(DIR2)INDEX2-
PRO=011657 REP=Y OWN=USER1-
ACC=OW(R W U M A D)
```

The following example copies file `TEMP` from the current working directory to file `PERM` in the same directory. `PERM` has the same attributes as `TEMP`, except that `PERM`'s owner and project group are those associated with the VOLMGR task. Because the REPLACE option is omitted, `TEMP` is not copied if `PERM` already exists.

```
COP TEMP PERM
```

The following example copies all files in directory `TEST` on volume `ATLAS04` to directory `TEST` on volume `ATLAS02`. If any of these files currently exist on `ATLAS02`, they are replaced. All files are shareable.

```
COP @ATLAS04(TEST)* @ATLAS02(TEST)* REP=Y SHA=Y
```

The following example copies all files in directory `MYDIR` that begin with `SJ.` to directory `YOURDIR` on the same volume. If any of these files currently exist in `YOURDIR`, the file is not copied from `MYDIR`.

```
COP @ATLAS04(MYDIR)SJ.* @ATLAS04(YOURDIR)* REP=N
```

The following example copies all files on volume `SOURCE` to volume `DESTINATION`. The first two characters of each file name are changed to `SJ`. The directories on `DESTINATION` must already exist.

```
COPY @SOURCE(*)* @DESTINATION(*)SJ*
```

The following example copies all files in directory `SYSTEM` on volume `V1BACKUP` to the same directory on volume `V1`. `SJ` is appended to each file name.

```
COPY @V1BACKUP(SYSTEM)* @V1(SYSTEM)*SJ
```

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 **C**OMMON [**PA**TH=]*partition* [**PROT**GRAN=]*grans* [**FIR**STPAGE=]*grannum*
[**ACC**ESS=*usertype*([**R**EAD] [**W**RITE] [**D**ELETE])]... [**BRI**EF={**Y**|**N**}]
[**MEM**CLASS={**E**|**H**|**S**}] [**OWN**ER=*ownername*] [**PROJ**ECTGROUP=*projname*]
[**REP**LACE={**Y**|**N**}]

[**PA**TH=]*partition*
partition is the pathname of the partition being created in one of the following forms:

GLOBAL*nn*
creates a global common partition, where *nn* is 00-99. This partition is physically located in any class of memory (E, H, or S).

DATAPOOL
creates a Datapool partition 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).

extname is a 1- to 8-character name to use for a memory partition in a task's extended address space. This partition can be mapped into memory above the first 128KW logical address space available to a task. Because the partition is in extended memory, certain restrictions apply. Partitions in a task's extended address space are physically located in any class of memory (E, H, or S).

[**PROT**GRAN=]*grans*
grans is 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 are write-protected from all sharing tasks. However, only one dynamic partition can be defined in any one map block.

[FIRSTPAGE=]*grannum*

grannum is the starting protection granule to map *partition* to (pages 0-1020 for the 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 SYSGENed common partitions. In extended address space, the top map block is reserved for MPX-32 use.

[ACCESS=*usertype*([READ] [WRITE] [DELETE])]

specifies a type of user and the kinds of access to *partition* this type is allowed. Repeat this option for each *usertype* desired. If not specified, owner and project group have complete access and others have read only access.

usertype	is one of the following types of users: OWNER owner of *partition* PROJECTGROUP members of *partition*'s project group OTHERS all other users
READ	permits *usertype* to read from the partition
WRITE	permits *usertype* to write to the partition
DELETE	permits *usertype* to delete in the partition

[BRIEF={Y|N}]

specifies whether to display complete partition status. N displays the complete partition status. Y displays only the partition name and size. The default is Y.

[MEMCLASS={E|H|S}]

specifies the class of memory for the partition. The default is class S.

[OWNER=*ownername*]

ownername is the owner name to associate with the partition. If not specified, defaults to the owner name associated with the VOLMGR task.

[PROJECTGROUP=*projname*]

projname is the project group name to associate with the partition. If not specified, defaults to the project group associated with the VOLMGR task.

[REPLACE={Y|N}]

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.

Examples

The following example creates partition GLOBAL55 in the current working directory. This partition is 5 pages in length and begins on page 75 in memory. The project group name associated with it is 082950.

```
CRE C PAT=GLOBAL55 PROT=5 FIR=75 PROJ=082950
```

The following example creates partition DATAPOOL on the current working volume and directory. It is 5 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 DATAPOOL 5 75 OWN=USER1 PROJ=061643-
ACC=OW(R W D)
```

The following example creates partition ATLAS on the SYSTEM volume and directory. It is 5 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 with the same name, it is replaced.

```
CRE C @SYSTEM(SYSTEM)ATLAS 5 75 MEM=H OWN=USER2-
REP=Y ACC=OW(R W D)
```

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=]*dirpath*
[ACCESS=*usertype*([READ] [DELDIR] [DELENT] [ADD] [TRAVERSE])]...
[BRIEF={Y|N}] [ENTRIES=*maxfiles*] [OWNER=*ownername*]
[PROJECTGROUP=*projname*] [SHARED={Y|N}]

[PATH=]*dirpath*
dirpath is the pathname to associate with the directory

[ACCESS=*usertype*([READ] [DELDIR] [DELENT] [ADD] [TRAVERSE])]...
specifies a type of user and the kinds of access to *dirpath* that type is allowed. Repeat this option for each *usertype* desired. If not specified, owner has complete access, project group has all but delete access, and others have read only and traverse access.

usertype is one of the following types of users:

OWNER	owner of *dirpath*
PROJECTGROUP	members of *dirpath*'s project group
OTHERS	all other users

READ	permits *usertype* to read entries in the directory
DELDIR	permits *usertype* to delete the directory
DELENT	permits *usertype* to delete entries in the directory
ADD	permits *usertype* to add entries to the directory
TRAVERSE	permits *usertype* to traverse the directory

[BRIEF={Y|N}]
specifies whether to display complete directory status. N displays complete directory status. Y displays the directory name and size only. The default is Y.

[ENTRIES=*maxfiles*]
maxfiles is the maximum number of files to be created in the directory. If omitted, the volume default is used (determined by disk allocation granularity).

[OWNER=*ownername*]
ownername is the owner name to associate with the directory. If omitted, defaults to the owner name associated with the VOLMGR task.

[PROJECTGROUP=*projname*]
projname is the project group name to associate with the directory. If omitted, defaults to the project group associated with the VOLMGR task.

[SHARED={Y|N}]
specifies if the directory can be used in the shared mode. The default is Y.

Examples

The following example creates directory `TEST` on the current working volume. `TEST`'s owner is `USER1`. This directory can contain `150` files, and all other system defaults apply to the directory.

```
CRE D PAT=TEST OWN=USER1 ENT=150
```

The following example creates directory `TEST2` on the current working volume. The directory's full status is displayed. The project group name is `061643` and the directory is not shareable. The owner is `USER2` and has full access privileges. The project group has read-only access and others have no access privileges.

```
CRE D TEST2 BRI=N PRO=061643 SHA=N OWN=USER2-
ACC=OW(R DELD DELE A T) ACC=PR(R) ACC=OT()
```

3.21 CREATE FILE Directive

The CREATE FILE directive creates a file on disk. Wild card characters cannot be used with this directive.

Syntax

CREATE [FILE] [PATH=]*newfile*
[ACCESS=*usertype*([READ] [WRITE] [UPDATE] [MODIFY] [APPEND] [DELETE])]...
[AUTOEXT={Y|N}] [BRIEF={Y|N}] [CONTIGUOUS={Y|N}] [EOFM={Y|N}]
[FAST={Y|N}] [MANEXT={Y|N}] [MAXINC=*maxblocks*] [MAXSIZE=*totblocks*]
[MININC=*minblocks*] [NOSAVE={Y|N}] [OWNER=*ownername*]
[PROJECTGROUP=*projname*] [REPLACE={Y|N}] [SEGNUM=*maxsegs*]
[SHARED={Y|N}] [SIZE=*blocks*] [START=*blocknum*] [ZERO={Y|N}]

[PATH=]*newfile*
newfile is the pathname of the file to create

[ACCESS=*usertype*([READ] [WRITE] [UPDATE] [MODIFY] [APPEND] [DELETE])]
specifies a type of user and the kinds of access to *newfile* this type is permitted. Repeat this option for each *usertype* desired. If not specified, owner and project group have complete access and others have read only access.

usertype	is one of the following types of users: OWNER owner of *newfile* PROJECTGROUP members of *newfile*'s project group OTHERS all other users
READ	permits *usertype* to read the file
WRITE	permits *usertype* to write to the file
UPDATE	permits *usertype* to update the file
MODIFY	permits *usertype* to modify the file
APPEND	permits *usertype* to append to the file
DELETE	permits *usertype* to delete the file

[AUTOEXT={Y|N}]
specifies if the file can be automatically extended. The default is Y.

[BRIEF={Y|N}]
specifies whether to display complete file status. N displays the complete file status. Y displays the file name and size only. The default is Y.

[CONTIGUOUS={Y|N}]
specifies if extensions to the file are to be contiguous when possible. Initial allocation of a file is always contiguous. The default is N.

[EOFM={Y|N}]
specifies the status of EOF management. The default is Y.

[**FAST**={**Y**|**N**}]
: specifies whether to retain *newfile*'s resource identifier when the file is copied or restored using VOLMGR. If omitted, the file is copied to any available resource identifier.

[**MAN**EXT={**Y**|**N**}]
: specifies if the file can be manually extended. The default is Y.

[**MAXI**NC=*maxblocks*]
: *maxblocks* is the size of each subsequent automatic increment of the file. The default is 64 blocks.

[**MAXS**IZE=*totblocks*]
: *totblocks* is the maximum size for an extendible file. If omitted, the size is limited only by available disk space.

[**MIN**INC=*minblocks*]
: *minblocks* is the minimum acceptable automatic increment size if *maxblocks* cannot be obtained. The default is 32 blocks.

[**NOS**AVE={**Y**|**N**}]
: specifies if the VOLMGR no-save attribute applies to the file. If omitted, the file is savable in response to the VOLMGR SAVE directive.

[**OWN**ER=*ownername*]
: *ownername* is the owner name to associate with *newfile*. If omitted, defaults to the owner name associated with the VOLMGR task.

[**PRO**JECTGROUP=*projname*]
: *projname* is the project group name to associate with *newfile*. If omitted, defaults to the project group associated with the VOLMGR task.

[**REP**LACE={**Y**|**N**}]
: if the file name *newfile* already exists, specifies whether it should be replaced (deleted). If omitted, a delete is not attempted on the existing *newfile*, and nothing is created.

[**SEG**NUM=*maxsegs*]
: *maxsegs* is the maximum number of segments permitted at creation if one contiguous space is not available. If SIZE=*blocks* is greater than the available space on the disk, VOLMGR posts a VM11 abort code. If the space requested with *blocks* is available but not in the number of segments requested by *maxsegs*, VOLMGR creates the file with one segment of the largest contiguous space available. If this occurs, VOLMGR generates no error messages, even though the file is not the requested size.

[**SHA**RED={**Y**|**N**}]
: specifies if *newfile* can be used in the shared mode. The default is Y.

[**SIZ**E=*blocks*]
: *blocks* is the initial block allocation size of *newfile*. The default is 16 blocks.

[START=*blocknum*]

blocknum is the starting block number of *newfile*. If the file cannot be created at the specified block number, a denial is given. If omitted, the file is created in any available space on the volume.

[ZERO={Y|N}]

specifies if the file should be zeroed on creation and extension. The default is N.

Note: Because the keyword FILE is optional for this directive, an error may result when trying to create a file named FILE. Avoid using this file name or create the file as shown in one of the following sample directives:

```
CREATE FILE PATH=@WORKING(USER1)FILE
CREATE PATH=FILE
CREATE F FILE
```

Examples

The following example 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 the file is automatically extendible.

```
CRE PAT=PROG1 SIZ=25 FAS=Y AUT=Y
```

The following example creates file PROG1 on the current working volume and directory. The file's complete status is displayed, its owner is USER2, and the owner has full access privileges.

```
CRE PROG1 BRI=N OWN=USER2 ACC=OW(R W U M A D)
```

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=**]*partition* [**BRIEF=**{**Y**|**N**}] [**CONFIRM=**{**Y**|**N**}]
[**CREATED=**{**BEFORE**|**SINCE**} {*date*|*time*|*date*,time}]

[**PATH=**]*partition*
: *partition* is the pathname of the partition to delete

[**BRIEF=**{**Y**|**N**}]
: specifies whether to display complete partition status. N displays complete partition status. Y displays only the partition name and size. The default is N.

[**CONFIRM=**{**Y**|**N**}]
: specifies if confirmation is required before deleting the partition. If Y, the user must respond to a confirmation prompt. If N, no prompt is issued. The default is N. This option is valid in the interactive mode only.

[**CREATED=**{**BEFORE**|**SINCE**} {*date*|*time*|*date*,time}]
: deletes common areas that match the specified options. If this option is omitted, creation date and time are ignored.

BEFORE	indicates before the specified date and time
SINCE	indicates since the specified date and time
date	is the date to compare *partition*'s date to. This is specified in the format *mo/dd/yy* or *dd-mon-yy*.
time	is the time to compare *partition*'s time to. This is specified in the format *hh*[:*mm*[:*ss*]].

Examples

The following example deletes partition `TEST2` on the current working volume and directory if it was created since March 31, 1988 at 8:00 a.m.

```
DEL C TEST2 CRE=S 3/31/88,08
```

The following example deletes partition `CASE1` on the current working volume and directory if it was created before midnight on April 23, 1988.

```
DEL C CASE1 CRE=B 23-APR-88
```

The following example deletes partition `JOBA` in directory `TEST` on the `SYSTEM` volume if it was created before 8:00 a.m. today. Confirmation is required before the partition is deleted and the complete status of the partition is displayed.

```
DEL C PAT=SYSTEM(TEST)JOBA BRI=N CON=Y CRE=B 08:00:00
```

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=]*dirpath* [BRIEF={Y|N}]
[{ACCESSED|CHANGED|CREATED}={BEFORE|SINCE} {*date*|*time*|*date,time*}]
[CONFIRM={Y|N}]

[PATH=]*dirpath*
: *dirpath* is the pathname of the directory to be deleted

[BRIEF={Y|N}]
: specifies whether to display complete directory status. N displays complete directory status. Y displays only the directory name and size. The default is Y.

[{ACCESSED|CHANGED|CREATED}={BEFORE|SINCE} {*date*|*time*|*date,time*}]
: deletes directories that match the specified keyword and options

ACCESSED	deletes directories accessed before or since a specified date/time. If omitted, accessed date and time are ignored.
CHANGED	deletes directories changed before or since a specified date/time. If omitted, changed date and time are ignored.
CREATED	deletes directories created before or since a specified date/time. If omitted, creation date and time are ignored.
BEFORE	indicates before the date/time
SINCE	indicates since the date/time
date	is the date to compare *dirpath*'s date to. This is specified in the format *mo/dd/yy* or *dd-mon-yy*.
time	is the time to compare *dirpath*'s time to. This is specified in the format *hh*[:*mm*[:*ss*]].

[CONFIRM={Y|N}]
: specifies if confirmation is required before deleting the directory. If Y, the user must respond to a confirmation prompt. If N, no prompt is issued. The default is N. This option is valid in the interactive mode only.

Examples

The following example deletes directory `APPJ` from the current working volume and displays its complete status.

```
DEL D APPJ BRI=N
```

The following example deletes directory `GLOSS` from the current working volume if it was created since April 29, 1988 at 2:30 p.m.

```
DEL D GLOSS CRE=S 04/29/88,14:30
```

The following example prompts for confirmation then deletes directory `GLOSS` from volume `VOL1`.

```
DEL D PAT=@VOL1(GLOSS) CON=Y
```

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 user must have delete access to the file and delete entry access to the directory where it is located. Wild card characters are valid, but require the user to have traverse access to the directory.

Syntax

DELETE [FILE] [PATH=]*delfile* [BRIEF={Y|N}] [CONFIRM={Y|N}]
[{ACCESSED|CHANGED|CREATED|RESTORED|SAVED}={BEFORE|SINCE}{*date*|*time*|*date,time*}]

[PATH=]*delfile*
delfile is the pathname of the file to delete

[BRIEF={Y|N}]
specifies whether to display complete file status. N displays the complete status. Y displays the file name and size only. The default is Y.

[CONFIRM={Y|N}]
specifies if confirmation is required before deleting the file. If Y, operator must respond to a confirmation prompt. If N, no prompt is issued. The default is N. This option is valid in the interactive mode only.

[{ACCESSED|CHANGED|CREATED|RESTORED|SAVED}={BEFORE|SINCE}{*date*|*time*|*date,time*}]
deletes files that match the specified keywords and options

ACCESSED
deletes files accessed before or since a specified date/time. If omitted, accessed date and time are ignored.

CHANGED
deletes files changed before or since a specified date/time. If omitted, changed date and time are ignored.

CREATED
deletes files created before or since a specified date/time. If omitted, creation date and time are ignored.

RESTORED	deletes files restored before or since a specified date/time. If omitted, restored date and time are ignored.
SAVED	deletes files saved before or since a specified date/time. If omitted, saved date and time are ignored.
BEFORE	indicates before the specified date/time
SINCE	indicates since the specified date/time
date	is the date to compare *delfile*'s date to. This is specified in the format *mo/dd/yy* or *dd-mon-yy*.
time	is the time to compare *delfile*'s time to. This is specified in the format *hh*[:*mm*[:*ss*]].

Examples

The following example prompts for confirmation and then deletes file `TEMP` from the current working volume and directory if it was restored since midnight, May 31, 1988.

```
DEL PAT=TEMP CON=Y RES=S 5/31/88
```

The following example deletes all files with 4-character file names that begin with `S` on the current working volume and directory. A confirmation prompt is issued for each file before it is deleted.

```
DEL S??? CON=Y
```

The following example deletes all files ending in `WRKFL` on volume `USER1` in directory `USER1`. A confirmation prompt is issued for each file before it is deleted.

```
DEL @USER1(USER1)*WRKFL CON=Y
```

3.25 EXIT Directive

The EXIT directive exits 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 remain the same. Wild card characters can be used with this directive.

Syntax

EXTEND [PATH=]*extfile* [BRIEF={Y|N}] [EXTSIZE=*blocks*]

[PATH=]*extfile*
: *extfile* is the pathname of the file to extend

[BRIEF={Y|N}]
: specifies whether to display complete file status. N displays complete file status. Y displays the file name and size only. The default is Y.

[EXTSIZE=*blocks*]
: *blocks* is the number of blocks to extend the file. If *blocks* cannot be obtained or if the option is omitted, it defaults to the MAXINC size specified in the resource descriptor. If the MAXINC value cannot be obtained, VOLMGR uses the MININC size specified in the resource descriptor.

Examples

The following example extends by 2 blocks all files with 8-character file names beginning with TC on the current working volume and directory and created as manually extendible.

```
EXT TC?????? EXT=2
```

The following example extends by the default increment size all files created as manually extendible in directory NEWPROG on volume TEST.

```
EXT @TEST(NEWPROG)*
```

3.27 HELP Directive

The HELP directive displays a short explanation of each VOLMGR directive.

Syntax

HELP [*directive*]

[*directive*] is the VOLMGR directive to display. If omitted, VOLMGR displays all directives and descriptions.

Examples

The following example displays an explanation of the EXTEND directive.

```
HEL EXTEND
```

3.28 LOG FILE Directive

The LOG FILE directive generates a formatted listing of the specified files. Wild card characters can be used with this directive.

Syntax

LOG FILE [LISTING=*listfile*] [PATH=]*logfile* [BRIEF={Y | N}]
[{ACCESSED | CHANGED | CREATED | RESTORED | SAVED}={BEFORE | SINCE}{*date* | *time* | *date,time*}]

[LISTING=*listfile*]
listfile is the pathname of a file to hold the listing. The default is SLO. This option can only be specified globally.

[PATH=]*logfile*
logfile is the pathname of the files to log

[BRIEF={Y | N}]
specifies whether to display complete file status. N displays complete file status. Y displays the file name and size only. The default is Y.

[{ACCESSED | CHANGED | CREATED | RESTORED | SAVED}={BEFORE | SINCE}{*date* | *time* | *date,time*}]
logs the files that match the specified keywords and options

ACCESSED	logs files accessed before or since the specified date/time. If omitted, accessed date and time are ignored.
CHANGED	logs files changed before or since the specified date/time. If omitted, changed date and time are ignored.
CREATED	logs files created before or since the specified date/time. If omitted, creation date and time are ignored.
RESTORED	logs files restored before or since the specified date/time. If omitted, restored date and time are ignored.
SAVED	logs files saved before or after the specified date/time. If omitted, saved date and time are ignored.
BEFORE	indicates before the date/time
SINCE	indicates since the date/time
date	is the date to compare *logfile*'s date to. This is specified in the format *mo/dd/yy* or *dd-mon-yy*.
time	is the time to compare *logfile*'s time to. This is specified in the format *hh*[*:mm*[*:ss*]].

Note: If wild card characters are used when specifying *logfile*'s directory, VOLMGR searches only directories that the user has read access to. If wild card characters are not used, the user only needs to have traverse access to the directory where *logfile* is located.

Examples

The following example logs all files with 4-character file names in directory `NEWDIR` on volume `ATLAS02`. Their file names and sizes are displayed.

```
LOG F @ATLAS02(NEWDIR)????
```

The following example logs all files on all public volumes that have been changed since 10:30 a.m. today. Their complete status is displayed. The log is directed to file `PROG`.

```
LOG F LIS=PROG PAT=@*(*)* BRI=N CHA=S 10:30
```

The following example logs file `GLOSS` in the current working directory and displays only its file name and size.

```
LOG F GLOSS
```

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 [**LIS**TING=*listfile*]

[**LIS**TING=*listfile*]
listfile is the pathname of a file to contain the listing. The default is SLO.

Examples

The following example logs all saved files in the current save image.

```
LOG I
```

The following example logs all saved files in the current save image in file `SAVELIST`.

```
LOG I LIS=SAVELIST
```

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=*listfile*] [[PATH=]*resource*] [BRIEF={Y | N}]
[ROOT={Y | N}]
[{ACCESSED | CHANGED | CREATED | RESTORED | SAVED}={BEFORE | SINCE}{*date* | *time* | *date,time*}]

[LISTING=*listfile*]
listfile is the pathname of a file to hold the listing. The default is SLO. This option can only be specified globally.

[[PATH=]*resource*]
resource is the pathname of the resource to log. If omitted, VOLMGR logs files from the current working directory.

[BRIEF={Y | N}]
specifies whether to display complete resource status. N displays complete resource status. Y displays only the resource name and size. The default is Y.

[ROOT={Y | N}]
specifies if the root directory is needed. When *resource* is a root directory and ROOT=Y is specified, the volume root directory information is logged. The default is ROOT=N.

[{ACCESSED | CHANGED | CREATED | RESTORED | SAVED}={BEFORE | SINCE}{*date* | *time* | *date,time*}]
logs resources that match the specified keyword and options

ACCESSED	logs resources accessed before or since a specified date/time. If omitted, accessed date and time are ignored.
CHANGED	logs resources changed before or since a specified date/time. If omitted, changed date and time are ignored.
CREATED	logs resources created before or since a specified date/time. If omitted, creation date and time are ignored.
RESTORED	logs resources restored before or since a specified date/time. If omitted, restored date and time are ignored.
SAVED	logs resources saved before or since a specified date/time. If omitted, saved date and time are ignored.
BEFORE	indicates before the specified date/time
SINCE	indicates since the specified date/time
date	is the date to compare *resource*'s date to. This is specified in the format *mo/dd/yy* or *dd-mon-yy*.

time is the time to compare *resource*'s time to. This is specified in the format *hh*[:*mm*[:*ss*]].

Notes:

Because the keyword RESOURCE is optional for this directive, an error may result when trying to log a resource named FILE, IMAGE, or RESOURCE since these are VOLMGR keywords. Avoid using these names for resources or log the resource as shown in the following sample directives:

1.
   ```
   LOG RESOURCE PATH=@VOL1(DIR1)FILE
   LOG RESOURCE PATH=@VOL1(DIR1)IMAGE
   LOG RESOURCE PATH=@VOL1(DIR1)RESOURCE
   ```

2.
   ```
   LOG PATH=FILE
   LOG PATH=IMAGE
   LOG PATH=RESOURCE
   ```

3.
   ```
   LOG R FILE
   LOG R IMAGE
   LOG R RESOURCE
   ```

Examples

The following example logs all files in the current working directory and displays only their file names and sizes.

```
LOG
```

The following example logs all files in directory `TEST` on the `SYSTEM` volume which were created since 2:00 p.m. today. It displays only their file names and sizes. The listing is written to file `LIST`.

```
LOG LIS=LIST @SYSTEM(TEST) CRE=S 14:00
```

The following example logs all files in the current working directory which were saved before midnight on April 29, 1988. It displays only their file names and sizes. The listing is written to file `OUTPUT` in directory `LOGOUT` on volume `ANYVOL`.

```
LOG R LIS=@ANYVOL(LOGOUT)OUTPUT SAV=B 29-APR-88
```

The following example logs all directory entries in the root directory of the current working volume.

```
LOG ^
```

The following example logs all files on directory `NEWCODE` on volume `TEST`.

```
LOG ^@TEST(NEWCODE)
```

The following example logs the root directory information on the current working volume.

```
LOG ^ROOT=Y BRI=N
```

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 [**LIS**TING=*listfile*] [**BRI**EF={**Y** | **N**}]

[**LIS**TING=*listfile*]
listfile is the pathname of a file to hold the listing. The default is SLO.

[**BRI**EF={**Y** | **N**}]
specifies whether to display complete file status. N displays complete file status. Y displays only file name and size. The default is Y.

Examples

The following example logs one file on the save tape and displays its file name and size.

```
LOG S
```

The following example logs one file on the save tape on the current working volume and directory and displays its full status. The listing is written to file `STAT`.

```
LOG S LIS=STAT BRI=N
```

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=]*currname* **[TO=]***newname* **[BRIEF={Y|N}]**

[FROM=]*currname*
currname is the current pathname of the file to rename

[TO=]*newname*
newname is the new pathname to give the file

[BRIEF={Y|N}]
specifies whether to display complete file status. N displays complete file status. Y displays file name and size only. The default is Y.

Notes:

If the *newname* file name already exists, it is left unchanged and the following message is displayed:

```
UNABLE TO RENAME
```

Currname and *newname* must have the same volume name.

Examples

The following example renames file `TEMP` to `PERM` in the current working directory and displays its full status.

```
REN TEMP PERM BRI=N
```

The following example renames file `TEMP` to `PERM` in directory `ATLAS04`. It displays only its file name and size.

```
REN FROM=^(ATLAS04)TEMP TO=^(ATLAS04)PERM
```

The following example renames all files in the current working directory that begin with `AA.`, changing the first three characters of each file name to `ZZ.`.

```
REN AA.* ZZ.*
```

The following example renames all files in the current working directory that begin with `BB`, appending `XX` to each file name.

```
REN BB* *XX
```

3.33 RESTORE DIRECTORY Directive

The RESTORE DIRECTORY directive restores specified files from a save tape to disk. Wild card characters can be used with this directive.

Syntax

RESTORE [DIRECTORY] [VOLUME=*volname*] [[FROM=]*sourcefile* [[TO=]*targetfile*]]
[SEGNUM=*maxsegs*]
[{ACCESSED | CHANGED | CREATED | RESTORED | SAVED}={BEFORE | SINCE}{*date* | *time* | *date,time*}]
[CONFIRM={Y | N}] [NEWEST={Y | N}] [BRIEF={Y | N}]

[VOLUME=*volname*]
: is the volume to restore the file to if different from the volume where the file was saved. If this volume does not exist, the restore does not take place. Do not specify this option if *targetfile* is specified.

[[FROM=]*sourcefile*
: is the pathname the file to restore was saved under

[[TO=]*targetfile*]]
: is the pathname to restore the file to. *targetfile* cannot be specified without *sourcefile*.

If *sourcefile* is specified but *targetfile* and *volname* are not, all files are restored to *sourcefile*.

If neither *sourcefile*, *targetfile*, or *volname* is specified, all files in the save image are restored to the volume and directory they were saved from.

[SEGNUM=*maxsegs*]
: *maxsegs* is the maximum number of segments permitted if one contiguous space is not available at restoration. If SIZE=*n* is greater than the available space on the disk, VOLMGR posts a VM11 abort code. If the space requested by the SIZE=*n* parameter is available but not in the number of segments requested with *maxsegs*, VOLMGR restores the file with one segment of the largest contiguous space available. If this occurs, VOLMGR generates no error messages, even though the file is not the requested size. If SEGNUM is specified, the file's extension attributes are temporarily ignored.

[{ACCESSED | CHANGED | CREATED | RESTORED | SAVED}={BEFORE | SINCE}{*date* | *time* | *date,time*}]
: restores files that match the specified keyword and options

ACCESSED
: restores files accessed before or since a specified date/time. If omitted, accessed date and time are ignored.

CHANGED
: restores files changed before or since a specified date/time. If omitted, changed date and time are ignored.

CREATED
: restores files created before or since a specified date/time. If omitted, creation date and time are ignored.

RESTORED	restores files restored before or since a specified date/time. If omitted, restored date and time are ignored.
SAVED	restores files saved before or since a specified date/time. If omitted, saved date and time are ignored.
BEFORE	indicates before the specified date/time
SINCE	indicates since the specified date/time
date	is the date to compare *sourcefile*'s date to. This is specified in the format *mo/dd/yy* or *dd-mon-yy*.
time	is the time to compare *sourcefile*'s time to. This is specified in the format *hh*[*:mm*[*:ss*]].

[CONFIRM={Y|N}]
specifies if confirmation is required before restoring the file. If Y, the user must respond to a confirmation prompt. If N, a prompt is not issued. The default is N. This option is valid in the interactive mode only.

[NEWEST={Y|N}]
if the file name *targetfile* already exists, specifies whether the newest version of it should be kept. If this option is omitted, the file on tape is restored if it meets specifications described in the notes section below.

[BRIEF={Y|N}]
specifies whether to display complete file status. N displays the complete file status. Y displays only the file name and size. The default is Y.

Notes:

Three aspects of RESTORE must be considered:

- the eligibility of a file to be restored due to date constraints supplied as options on the directive line
- the owner name, project group, and access attributes of *targetfile*
- the values of the created, changed, saved, and restored dates 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. ACCESSED, CREATED, CHANGED, SAVED, RESTORED

      The file is eligible if the date in the file's resource descriptor satisfies the comparison condition (SINCE/BEFORE) and date/time specified in the directive.

   b. CONFIRM

      Confirmation is performed before any restoring is performed unless CONFIRM=N is specified.

   c. NEWEST

      If the file does not already exist on disk, it is automatically eligible to be restored.

      If the file already exists on disk, the tape version is eligible if:

      - the create dates match but the changed date on tape is newer than the changed date on disk
      - the create date on tape is newer than the create date on disk

2. Attributes of the restored file are handled as follows:
   a. The disk destination is implicit (*targetfile* 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 neither the system administrator or the owner of the file(s) being restored, an error is generated.

   b. The disk destination is explicit (*targetfile* specified)

      This is a copy of the tape file rather than a true restore. If the file already exists on disk, it is overwritten. If the file does not already exist on disk, it is created.

      If the new directory name does not exist on disk an error is generated. The directory must already exist since VOLMGR cannot make any reasonable defaults.

If the new directory name does exist on disk:

- When the user running the job is not the system administrator (SA), 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 SA, 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. ACCESSED date

      Set to the last accessed date from the tape.

   b. CREATED date

      Set to the created date from the tape.

   c. CHANGED date

      Set to the changed date from the tape.

   d. SAVED date

      Set to the saved date on the tape if the SA is running the restore. The saved date remains unchanged if it is not the SA running the restore and the file already exists on disk. 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.

Examples

The following example restores all files in the save image that were saved from directory `TEMP` on volume `ATLAS04` to the same directory and volume. It requests confirmation for each file before it is restored.

```
RES FRO=@ATLAS04(TEMP)* CON=Y
```

The following example restores all files in the save image created since 5:00 a.m. on August 29, 1988 to volume `TESTCASE` in the directory where they were saved. It requests confirmation before each file and displays only the file name and size of each.

```
RES VOL=TESTCASE CON=Y CRE=S 29-AUG-88,05:00
```

The following example restores all files in the save image which were changed since midnight on April 1, 1988 to the volume and directory where they were saved. If any file on the tape still exists on disk, the copy of the file on tape replaces the copy of the file on disk according to eligibility rules stated in 1c above.

```
RES CHA=S 04/01/88 NEW=Y
```

The following example restores all files in the save image to the same volume, directory, and file name where they were saved.

```
RESTORE
```

The following example restores all files in the save image to `@SYSTEM(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 in `@SYSTEM(SYSTEM)`.

```
RES FRO=@*(*)* TO=@SYSTEM(SYSTEM)*
```

The following example restores all files beginning with `F66` in the save image which were saved from directory `SYSTEM` on volume `V1` to directory `F77` on volume `VF77`. The first three characters of each file name are changed to `F77`.

```
RES @V1(SYSTEM)F66* @VF77(F77)F77*
```

The following example restores all files in the save image which were saved from directory `SYSTEM` on volume `V1` to directory `F77` on volume `VF77`. The characters `F77` are appended to each file name.

```
RES @V1(SYSTEM)* @VF77(F77)*F77
```

3.34 RESTORE POSITION Directive

The RESTORE POSITION directive restores one file from a magnetic tape to disk. 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 **P**OSITION [[TO=]*targetfile*]

[[TO=]*targetfile*]
: *targetfile* is the pathname of the file to restore the data to. If omitted, the file is restored to the pathname that it was saved under.

Examples

The following example 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
```

The following example restores the file on tape to file `SS*WRKFL` in directory `SOURCE` on volume `TEST`.

```
RES P @TEST(SOURCE)'SS*WRKFL'
```

3.35 REWIND Directive

The REWIND directive positions a magnetic tape to its loading point, beginning of tape (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.

Note: If an I/O error occurs while the system administrator is saving files, all files processed before the error are saved on the tape. All files after the error can be processed with another SAVE directive on another tape.

Syntax

SAVE [PATH=]*savefiles* [BRIEF={Y|N}]
[{ACCESSED|CHANGED|CREATED|RESTORED|SAVED}={BEFORE|SINCE}{*date*|*time*|*date,time*}]
[CONFIRM={Y|N}] [SAVN={Y|N}]

[PATH=]*savefiles*
savefiles is the pathname of the files to save

[BRIEF={Y|N}]
specifies whether to display complete file status. N displays complete file status. Y displays only the file name and size. The default is Y.

[{ACCESSED|CHANGED|CREATED|RESTORED|SAVED}={BEFORE|SINCE}{*date*|*time*|*date,time*}]
saves files that match the specified keyword and options

ACCESSED	saves files accessed before or since a specified date/time. If omitted, accessed date and time are ignored.
CHANGED	saves files changed before or since a specified date/time. If omitted, changed date and time are ignored.
CREATED	saves files created before or since a specified date/time. If omitted, creation date and time are ignored.
RESTORED	saves files restored before or since a specified date/time. If omitted, restored date and time are ignored.
SAVED	saves files saved before or since a specified date/time. If omitted, saved date and time are ignored.
BEFORE	indicates before the specified date/time
SINCE	indicates since the specified date/time
date	is the date to compare a file's date to. This is specified in the format *mo/dd/yy* or *dd-mon-yy*.
time	is the time to compare a file's time to. This is specified in the format *hh*[:*mm*[:*ss*]].

[CONFIRM={Y|N}]
specifies if confirmation is required before saving each file. If Y, the user must respond to a confirmation prompt. If N, no prompt is issued. The default is N. This option is valid in the interactive mode only.

[SAVN={Y|N}]
specifies if the VOLMGR no-save attribute should be overridden. If omitted, files created with the no-save attribute enabled are not saved.

Notes:

Four aspects of SAVE must be considered:

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 accessed, 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. ACCESSED, CREATED, CHANGED, SAVED, RESTORED

      The file is eligible if the relevant date in the resource descriptor of the file on disk 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 CONFIRM=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. ACCESSED, CREATED, CHANGED, and RESTORED dates

      These dates remain unchanged on disk and tape.
   b. SAVED date

      On tape, the date always reflects the date of the save.

      On disk, 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.

Examples

The following example saves all files in the current working directory.

```
SAVE *
```

The following example saves all files in all volumes and all directories.

```
SAVE @*(*)*
```

The following example saves all files in all directories on the SYSTEM volume and displays full status for each file.

```
SAV @SYSTEM(*)* BRI=N
```

The following example saves all files in directory TEMP on volume TEST which 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.

```
SAV PATH=@TEST(TEMP)XX.???? RES=B 15:30
```

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 tape, the saved date always reflects the date of the save. On disk, the saved date only updates if the system administrator executes the save. The created, changed, and restored dates remain unchanged on disk and tape.

Wild card characters can be used with this directive.

Note: If an I/O error occurs while the system administrator is using the SAVE INCREMENTAL directive, all files processed before the error are saved on the tape. All files after the error can be processed with another SAVE I directive on another tape.

Syntax

SAVE INCREMENTAL [**PA**TH=]*savefile*

[**PA**TH=]*savefile*
savefile is the pathname of the file to save

Examples

The following example saves all files in all directories on volume `NEWVOL` which have been created or changed since the last SAVE was performed.

```
SAVE I @NEWVOL(*)*
```

The following example saves file `SOURCE` in the current working directory if it has been created or changed since the last SAVE was performed.

```
SAV I SOURCE
```

3.38 SDT Directive

The SDT directive is most commonly used for creating a system distribution tape (SDT). An SDT 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 R3. 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=]*imagefile* [WEOF={Y|N}] [,[PATH=]*imagefile* [WEOF={Y|N}]]...

[PATH=]*imagefile*
imagefile is the pathname of the file containing the system image. More than one pathname can be specified (see Notes below).

[WEOF={Y|N}]
specifies whether to write an end-of-file after the associated *imagefile*. This option can only be specified locally. If omitted, an end-of-file is not written.

Notes:

For a user tape or floppy disk SDT, one PATH=*imagefile* is required. When no other *imagefiles* are specified, the default set of load modules and format is as follows:

bootstrap loader (standard)
system image
J.VFMT
end-of-file (EOF) mark
J.MOUNT
J.SWAPR
VOLMGR
EOF mark
EOF mark

For a master floppy disk SDT, the following is required:

```
PATH=@SYSTEM(SYSTEM)FLOP.SYS
```

This format 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 EOF marks. Refer to the Examples section.

The LFC 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 disk.

Examples

The following example generates a user SDT that matches the default.

Note: To properly generate the SDT using this format, you must specify WEOF=Y for VOLMGR's WEOF parameter.

```
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
```

The following example generates a user SDT with the system image located in file `IMAGEFILE` in `SYSTEM(SYSTEM)`.

```
SDT PATH=@SYSTEM(SYSTEM)IMAGEFILE
```

Following is the format of a custom-built user SDT and the example that generates it:

bootstrap loader (standard)
IMAGEFILE
LOADMOD1
LOADMOD2
EOF mark
EOF mark

```
SDT PATH=@SYSTEM(SYSTEM)IMAGEFILE WEOF=N-
,PATH=@SYSTEM(SYSTEM)LOADMOD1 WEOF=N-
,PATH=@SYSTEM(SYSTEM)LOADMOD2 WEOF=Y
```

The following example performs master SDT processing when the system is booted.

```
SDT PATH=@SYSTEM(SYSTEM)FLOP.SYS
```

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

Notes:

The tape format is as follows:

bootstrap loader
MSTRALL
EOF mark
MSTREXT
EOF mark
MSTROUT
EOF mark
J.VFMT
EOF mark
J.MOUNT
J.SWAPR
VOLMGR
EOF mark
EOF mark

LFC TAP is assumed to be assigned before invoking VOLMGR. The device assignments accepted by the SDT MASTER directive must be for magnetic tape or floppy disk.

Before issuing this directive, the system image files MSTRALL, MSTREXT, and MSTROUT must exist and be located on the system volume in the system directory. 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 disk SDT, use the SDT directive and specify the following:

```
PATH=@SYSTEM(SYSTEM)FLOP.SYS
```

To generate the FLOP.SYS image, use the directive file DIR.27F or DIR.87F provided with the release copy of MPX-32. The special file name FLOP.SYS causes master SDT processing when booting the system.

For more information about master SDTs, refer to the MPX-32 Reference Manual, Volume III, Chapter 2.

3.40 SET Directive

The SET directive establishes global default option values for VOLMGR directives where the specific options are applicable.

Syntax

SET [AUTOEXT{Y|N}] [BRIEF={Y|N}]
[{ACCESSED|CHANGED|CREATED|RESTORED|SAVED}={BEFORE|SINCE}{*date*|*time*|*date,time*}]
[CONFIRM={Y|N}] [CONTIGUOUS={Y|N}] [EOFM={Y|N}] [FAST={Y|N}]
[LISTING=*listfile*] [MANEXT={Y|N}] [MAXINC=*maxblocks*]
[MAXSIZE=*totblocks*] [MEMCLASS={E|H|S}] [MININC=*minblocks*]
[NEWEST={Y|N}] [NOSAVE={Y|N}] [OWNER=*ownername*]
[PROJECTGROUP=*projname*] [REPLACE={Y|N}] [SHARED={Y|N}]
[SIZE=*blocks*] [START=*blocknum*] [ZERO={Y|N}]

[AUTOEXT={Y|N}]
specifies if the file can be automatically extended. The default is Y.

[BRIEF={Y|N}]
specifies whether to display complete file status. N displays the complete status. Y displays the file name and size only. The default is Y.

[{ACCESSED|CHANGED|CREATED|RESTORED|SAVED}={BEFORE|SINCE}{*date*|*time*|*date,time*}]
processes resources that match the specified keyword and options

ACCESSED	processes resources accessed before or since a specified date/time. If omitted, accessed date and time are ignored.
CHANGED	processes resources changed before or since a specified date/time. If omitted, changed date and time are ignored.
CREATED	processes resources created before or since a specified date/time. If omitted, creation date and time are ignored.
RESTORED	processes resources restored before or since a specified date/time. If omitted, restored date and time are ignored.
SAVED	processes resources saved before or since a specified date/time. If omitted, saved date and time are ignored.
BEFORE	indicates before the specified date/time
SINCE	indicates since the specified date/time
date	is the date to compare a resource's date to. This is specified in the format *mo/dd/yy* or *dd-mon-yy.*
time	is the time to compare a resource's time to. This is specified in the format *hh*[:*mm*[:*ss*]].

[CONFIRM={Y | N}]
: specifies if confirmation is required before processing a directive. If Y, the user must respond to a confirmation prompt. If N, no prompt is issued. The default is N. This option is valid in the interactive mode only.

[CONTIGUOUS={Y | N}]
: specifies if extensions to the file are to be contiguous when possible. Initial allocation of a file is always contiguous. The default is N.

[EOFM={Y | N}]
: specifies the status of EOF management. If omitted, the system default is used.

[FAST={Y | N}]
: specifies if a file's resource identifier is to remain unchanged when the file is copied or restored using VOLMGR. If omitted, the file is placed in any available resource identifier.

[LISTING=*listfile*]
: *listfile* is the pathname of a file to hold the listings. The default is SLO.

[MANEXT={Y | N}]
: specifies if the file can be manually extended. The default is Y.

[MAXINC=*maxblocks*]
: *maxblocks* is a size for subsequent automatic increments. The default is 64 blocks.

[MAXSIZE=*totblocks*]
: *totblocks* is a maximum size for an extendible file. If omitted, the size is limited only by available disk space.

[MEMCLASS={E | H | S}]
: specifies the class of memory. The default is class S.

[MININC=*minblocks*]
: *minblocks* is a minimum acceptable automatic increment size if *maxblocks* cannot be obtained. The default is 32 blocks.

[NEWEST={Y | N}]
: if a resource already exists, indicates whether the most recent version of it is saved.

[NOSAVE={Y | N}]
: specifies if the VOLMGR no-save attribute applies to a resource. If omitted, the resource is savable in response to the VOLMGR SAVE directive.

[OWNER=*ownername*]
: *ownername* is an owner name. If omitted, defaults to the owner name associated with the VOLMGR task.

[**PRO**JECTGROUP=*projname*]
projname is a project group name. If omitted, defaults to the project group name associated with the VOLMGR task.

[**REP**LACE={**Y**|**N**}]
if a resource currently exists, indicates whether it should be replaced (deleted). If omitted, the existing resource remains unchanged.

[**SHA**RED={**Y**|**N**}]
specifies if a resource is to be used in the shared mode. The default is Y.

[**SIZ**E=*blocks*]
blocks is a block size for initial allocation. The default is 16 blocks.

[**STA**RT=*blocknum*]
blocknum is a starting block number. If the specified block number cannot be used, a denial is given. If omitted, any available space on the volume is used.

[**ZER**O={**Y**|**N**}]
specifies if a resource is to be zeroed on creation and extension. The default is N.

Examples

The following example sets the default project group name to `REL2.0`, the default starting block number to `25`, and the default initial allocation size to `5` blocks. It specifies to retain the resource identifier for files, to not issue confirmation prompts with VOLMGR directives, and to display file name and size only. The example specifies that files can be extended automatically and manually, and that extensions should be contiguous. These default option values apply unless overridden on the directive line.

```
SET PRO=REL2.0 STA=25 SIZ=5 FAS=Y CONF=N-
BRI=Y AUT=Y MAN=Y CONT=Y
```

3.41 SKIP END Directive

The SKIP END directive positions a save tape at its logical EOT.

Syntax

SKIP END

3.42 SKIP FILE Directive

The SKIP FILE directive advances a save tape 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 to its logical EOT.

Syntax

SKIP FILE [[**FILE=**]*eofs*]

[[**FILE=**]*eofs*]
eofs is the number of EOF marks to skip. The default is one EOF mark.

Examples

The following example advances the tape one EOF mark.

```
SKI F
```

Each of the following examples advances the tape two EOF marks.

```
SKI F FIL=2
SKI F 2
```

3.43 SKIP IMAGE Directive

The SKIP IMAGE directive advances a save tape a specified number of save images.

Syntax

SKIP [IMAGE] [[**IMA**GES=]*saveimages*]

[[**IMA**GES=]*saveimages*]
saveimages is the number of save images to skip. The default is one save image.

Examples

The following example advances the tape one save image.

```
SKI
```

Each of the following examples advances the tape two save images.

```
SKI IMA=2
SKI 2
```

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 [**PA**TH=]*truncfile* [**BR**IEF={**Y**|**N**}]

[**PA**TH=]*truncfile*
truncfile is the pathname of a file whose space is to be decreased

[**BR**IEF={**Y**|**N**}]
specifies whether to display complete file status. N displays complete file status. Y displays the file name and size only. The default is Y.

Examples

The following example truncates file `INDEX` in directory `NEWDIR` on volume `NEWVOL`.

```
TRU @NEWVOL(NEWDIR)INDEX
```

The following example truncates all files on the current working volume and directory.

```
TRU *
```

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.

The following error message is generated when an I/O error occurs.

```
I/O ERRORS(S) DEV=devmnc LFC=lfc OPER=op FCB STATUS -
FCB.SFLG=value FCB.ISTI=value FCB.IST2=value
```

devmnc	is the device mnemonic of a configured peripheral device where I/O was attempted
lfc	is the 1- to 3-character logical file code used by VOLMGR
op	is the type of I/O operation requested when the error occurred
value	is an 8-digit hexadecimal value contained in the specified FCB word
`FCB.SFLG`	represents word 3 of the FCB
`FCB.1ST1`	represents word 11 of the FCB
`FCB.1ST2`	represents word 12 of the FCB

Abort codes and their messages are described in Appendix C.

4 COMPRESS

4.1 Introduction

COMPRESS collects object modules into one file which can be cataloged and executed. For information about using COMPRESS to build an object file for system generation, see MPX-32 Reference Manual, Volume III.

4.1.1 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.

4.2 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).

4.2.1 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.

The default assignment for IN is to the system file JH.32:

```
$ASSIGN IN TO JH.32 BLO=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

4.2.2 Output File (OT)

The output file contains the concatenated object modules output by COMPRESS. The output file is assigned to logical file code OT.

The default assignment for OT is to the system file OH.32:

```
$ASSIGN OT TO OH.32 BLO=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

4.2.3 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.

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

DEV=*devmnc*
devmnc is the mnemonic of a device to contain the COMPRESS audit trail

4.2.4 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*

4.3 Error Messages

When one of the following error messages is displayed to the LO 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.

4.4 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 is an example of listed output 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
```

5 Rapid File Allocation Utility (J.MDTI)

5.1 Overview

The Rapid File Allocation Utility (J.MDTI) allows copies of resource descriptors (RDs) for selected permanent files to reside in the memory resident descriptor table (MDT). This eliminates the disk access normally necessary to allocate these files. 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 other disk allocations. To avoid this overhead, refer to the Programming Considerations section in this chapter.

The Rapid File Allocation utility contains:

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 and 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.

For instructions for setting up an input file, see the Input Files section of this chapter.

The following sections describe accessing J.MDTI from TSM or user tasks to append the MDT.

5.2 Accessing J.MDTI

At system initialization, J.MDTI is 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 given in the Input Files section in this chapter. 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
```

Access to J.MDTI can be restricted for owner names by setting the appropriate bit in the M.KEY file.

5.3 Logical File Code Assignments

There are five logical file codes (LFCs) 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 LFC 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 LFC 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 LFC RPN.

The default assignment for RPN is to LFC 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 LFC SLO.

The default assignment for SLO is to LFC 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 Assignments
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 returns to TSM.

5.5 Input Files

Input files contain the pathnames and 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 in a file name other than M.MDTF.

Input files are composed of:

- 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 can be entered in any order.

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 disk 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.

Notes:

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.

5.5.1 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 the following 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: To run these examples, replace the RIDs listed here with actual RIDs.

5.6 Programming Considerations

The MDT entries are duplicates of the RDs which reside on disk. 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 disk structures directly, not using the H.VOMM services, can destroy the MDT's integrity; direct changes made to RDs 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 disk.

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.

5.7.1 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.
```

5.7.2 Listed Output

All of the following messages are written to LFC SLO and form the audit trail.

The following 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 LFC 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 disk.

6 Shadow Utility (J.SHAD)

6.1 Introduction

The Shadow Utility (J.SHAD) specifies the portions of a task's logical address space that are to be located in shadow memory (shadowed). The specified portions are shadowed each time the task is activated until the shadowed area is redefined by J.SHAD. Specification alters the load module's RRS by adding type 11 RRSs (assign to shadow memory) to the load module's RRS block. 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

If accessed interactively, J.SHAD displays the prompt `J.S>`. Enter the name of the task to be shadowed:

`J.S>` *taskname*

Enter J.SHAD directives at the further prompts. The J.SHAD 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 J.SHAD: 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 LFC SYC.

In the interactive mode, the default assignment for SYC is to LFC 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 LFC UT.

In interactive mode, the default assignment for UT is to LFC UT. In 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 LFC INT.

There are no default or optional assignments for INT.

6.3.4 LFC Summary

Table 6-1 shows the LFCs used by J.SHAD and their default assignments. There are no optional assignments.

Table 6-1
J.SHAD LFC Summary

LFC	Mode	Default Assignment
SYC	interactive	UT
	batch	SYC
UT	interactive	UT
	batch	SLO
INT	--	None

6.4 Shadow Utility Directives

6.4.1 EXIT Directive

The EXIT directive exits J.SHAD.

Syntax

EXIT

Note: J.SHAD 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

$SHADOW {**ALL**|**STA**CK|**TAS**K|**TSA**|*start end* [**A**BSOLUTE|**C**ODE|**D**ATA]}
[**REQ**UIRED]

ALL	specifies that the entire task, including the TSA, is to be located in shadow memory
STACK	specifies that the task's stack is to be located in shadow memory. This option applies only to base mode tasks and is ignored for nonbase mode tasks.
TASK	specifies that the entire task, except the TSA, is to be shadowed. If the task plus its TSA are only one map block, the entire space is shadowed.
TSA	specifies only the TSA is to be shadowed. If the task plus its TSA are only one map block, the entire space is shadowed.
start	is the hexadecimal logical starting address of the logical address space to be located in shadow memory
end	is the hexadecimal logical ending address of the logical address to be located in shadow memory

The following specify that *start* and *end* are:

ABSOLUTE	relative to logical address zero
CODE	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	relative to the task's data section origin as opposed to the default assumption that the addresses are relative to the task's start.

[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.

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 disk 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

The following example shadows 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>
```

The following example shadows the task without its TSA with shadow memory required.

```
TSM> J.SHAD
J.S> @SYSTEM^(SYSTEM)MYTASK
J.S> SHAD TAS REQ
J.S> EXIT
TSM>
```

The following example shadows the 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>
```

The following example clears shadow RRSs in a task.

```
TSM> J.SHAD
J.S> USERTASK
J.S> EXIT
TSM>
```

The following example shadows all of a task in batch with shadow memory not required.

```
$JOB SHADOW
$J.SHAD
USERTASK
SHADOW ALL
$EOJ
$
```

7 ANSI Labeled Tapes

7.1 General Information

The ANSI labeled tape utilities allow the use of ANSI labeled tapes on MPX-32 Revision 3.3 and later systems. These MPX-32 systems process ANSI labeled tapes according to implementation level 4 of ANSI standard X3.27-1978.

The ANSI labeled tape utilities 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 about the utilities, see the individual sections in this chapter.

ANSI labeled tapes must be used for data transfers to non-MPX-32 systems that require ANSI labeled tapes. The ANSI standard allows 17-character file names. The ANSI labeled tape utilities also read tapes with lowercase letters and underscores in the labels. 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 MPX-32.

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 J.LABEL. The label contents can be logged by AVOLM to ensure that the character strings follow the ANSI standard.

After the tape set is labeled, it is mounted using 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 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 10. 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. 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 I/O operations that can be performed 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 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 the $ASSIGN directive in Chapter 1. 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 filled with circumflexes (^).

7.2.1.1 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.

7.2.1.2 Variable-Length Records

Variable-length records on an ANSI labeled tape set are preceded by a 4-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.

7.2.1.3 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 J.LABEL. User labels are optional and can be read and written as discussed below.

7.2.3.1 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.

7.2.3.2 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

When an AMOUNT is issued, the following message is displayed. The second line of the message is the AMOUNT parameters.

`PREPARE FOR ANSI MOUNT`
lvid=vid[*,vid...*] [`DEV=`*devmnc*]

On the first I/O request from the user, the following 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).

`MOUNT` *vid* `ANSI ON` *devmnc* `REPLY R, A, H, OR DEVICE:`

When a tape change is required, the tape rewinds and the following dismount message is displayed.

`DISMOUNT` *vid* `ANSI FROM` *devmnc*

If the incorrect tape is mounted, the following message is displayed.

`INCORRECT ANSI TAPE` *vid* `MOUNTED FOR` *vidl*

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, the following message is displayed. Respond `Y` or `N` depending on the desired effect.

`SUBSTITUTE ANSI TAPE` *vid* `FOR` *vidl*`? (Y/N)`

When a non-ANSI tape is mounted on an ANSI tape drive, the following message is displayed.

```
THAT TAPE IS NOT ANSI FORMAT
```

When an incorrect tape is mounted or the mounted tape is non-ANSI, the following message is displayed. Respond `Y` or `N` depending on the desired effect.

```
ABORT TASK REQUESTING TAPE (Y/N)?
```

Usage

When an ADMOUNT is issued, the tape rewinds and a normal dismount message is displayed followed the message:

```
FINISHED WITH ANSI LVID=lvid
```

When the tape is not the continuation of the file currently being processed, the following message is displayed. Mount the correct tape.

```
ANSI TAPE vid DOES NOT CONTINUE CURRENT FILE
```

7.2.6 Examples

7.2.6.1 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>
```

7.2.6.2 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 ACC=(A) FOR=F REC=132 BSI=1024-
PRO=0 EXP=86096
$MEDIA
COPY,SI,SO
EXIT
END
$ASSIGN SI TO TEST.DATA.02
$ASSIGN SO TO @ANSITAPE(SAMTAP)FILE2 ACC=(A)
$MEDIA
COPY,SI,SO
EXIT
END
$ADMOUNT SAMTAP
```

7.2.6.3 Tape Reading

The following job copies the contents of two files produced by the previous job to the printer. The format of the files is read 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 ANSI labeled tape utilities are documented (in alphabetical order) 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

Note: Each utility has one logical file code (LFC) called OUT that is assigned to LFC=UT at CATALOG time.

7.3.1 Dismount ANSI Labeled Tape Utility (ADMOUNT)

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 1- to 6-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
```

Error Messages

The following are the ADMOUNT error messages and their explanations:

```
ERROR - LOGICAL VOLUME IDENTIFIER MUST BE SPECIFIED
ERROR - NOT MOUNTED
ERROR - ONLY ONE PARAMETER CAN BE SPECIFIED
ERROR - VID MUST BE 1 THROUGH 6 CHARACTERS IN LENGTH
```

If the specified LVID has not been mounted by this owner, the following message is displayed:

```
ERROR - VOLUME IS ACTIVE
```

When there is an active assignment to a file on the LVID, the following message is displayed:

```
    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.

If there is insufficient memory space for the ANSI labeled tape processing, the following message is displayed.

```
ERROR - NO J.ATAPE TABLE SPACE IS AVAILABLE FOR ANSI PROCESSING
```

Wait for available memory.

7.3.2 Mount ANSI Labeled Tape Utility (AMOUNT)

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=vid [,*vid....*]
: *lvid* is a 1- to 6-character identifier that accesses a set of tapes. *vid* is a 1- to 6-character volume identifier of each tape that specifies the order that the tapes are accessed. Up to 10 volume identifiers can be specified.

[**DEV**=*devmnc*]
: specifies the 2-, 4- or 6-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
```

Error Messages

If J.ATAPE could not be run requested, the following messages are displayed. See M.SRUNR in Volume I, Chapter 6 for information on the returned error status.

```
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
```

If there is insufficient memory to process the ANSI labeled tape, the following message is displayed. Wait for available memory.

```
ERROR - NO J.ATAPE TABLE SPACE IS AVAILABLE FOR ANSI PROCESSING
```

If the specified LVID is already in use, the following message is displayed. Choose a different LVID or wait until the specified LVID is available.

```
ERROR - DUPLICATE LVID
```

7.3.3 Display ANSI Labeled Tape Utility (ASTAT)

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               } List
     OWNER2 VOL2               }
```

Error Messages

If the specified LVID is already in use, the following message is displayed. Choose a different LVID or wait until the specified LVID is available.

```
DUPLICATE LVID
```

If J.ATAPE could not be run requested, the following message is displayed. See M.SRUNR in Volume I, Chapter 6 for information on the returned error status.

```
ERROR - INVALID J.ATAPE START STATUS
RETURNED STATUS IS nn
```

If there is insufficient memory for ANSI labeled tape processing, the following message is displayed. Wait for available memory.

```
ERROR - NO J.ATAPE TABLE SPACE IS AVAILABLE FOR ANSI PROCESSING
```

7.3.4 Log ANSI Labeled Tape Utility (AVOLM)

AVOLM performs the following functions:

- lists format information and volume identifier's file identifier
- lists the contents of an ANSI labeled tape set
- 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* [, *vid*...]]
: *vid* is a 1- to 6-character volume identifier for each tape. Up to ten volume identifiers can be specified. If more than one VID is specified, they must form a tape set.

[**DEV=***devmnc*]
: *devmnc* specifies the 2-, 4-, or 6-character device mnemonic of the tape drive to be used

[BRIEF={Y|N}]

specify BRIEF=N to display each label on the tape and verification of the number of data blocks used. Specify BRIEF=Y to display the following information:

VOL1	the VID, owner, and 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	the 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.

Error Messages

The following are the AVOLM error messages:

```
ERROR - OPTIONS MUST FOLLOW THE VID LIST
ERROR - VID MUST BE ONE 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)

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*] [**OWNE**R=*ownername*] [**PROT**ECT=*p*]

vid is a 1- to 6-character ANSI volume identifier

[**DEV=** *devmnc*]
devmnc specifies the 2-, 4-, or 6-character device mnemonic of the tape drive to be used

[**OWNE**R=*ownername*]
ownername is the owner name for the tape

[**PROT**ECT=*p*]
p is a one-character identifier of the protection to be applied. Zero specifies owner only access. A through Z are reserved by the ANSI specification for installation specific protection. MPX-32 treats A through Z as owner-only protection. If not specified, the default is no protection.

On successful completion, J.LABEL is exited.

Error Messages

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 - 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
ERROR - MAGNETIC TAPE ASSIGNMENT FAILURE
```

The last error is generated if the specified device is off-line.

8 Sort/Merge (FSORT2)

8.1 Introduction

Sort/Merge (FSORT2) sorts the contents of standard MPX-32 files in a user-specified sequence and outputs the results to a file. FSORT2 directives let the user detail the order and sequence rules used for sorting. The TSM $ALLOCATE directive allows expansion of the sort and merge work areas for improved performance.

Subroutines can be called from ASSEMBLE or FORTRAN77+ programs to produce the same sorting for user programs. These subroutines use argument lists that are comparable to the FSORT2 directives. The routines are accessed with the CATALOG assignment to SORT.LIB and SORT.DIR. For more information refer to the CATALOG section in the MPX-32 Utilities 3*.x* Reference Manual.

Note: An end-of-file (EOF) is required on all input files to designate end of data for Sort/Merge.

8.2 Logical File Code (LFC) Assignments

The LFCs for FSORT2 are: control statements (CTL), primary input (IN), secondary inputs (IN1 through IN8), merge inputs (M1 through M8), primary output (OUT), listing file (LO), designated work file device (..1 and ..2) and work file (WK1, WK2, .W1, .W2).

8.2.1 Control Statements (CTL)

Control statements are read from LFC CTL.

The default assignment for CTL is to LFC SYC:

```
    $ASSIGN CTL TO SYC
```

There are two optional assignments for CTL:

$ASSIGN CTL TO {*pathname* | **DEV=***devmnc*}

pathname is the pathname of any valid MPX-32 file

DEV= *devmnc*
devmnc is the mnemonic of any device except the null device

8.2.2 Primary Input (IN)

Data elements to be sorted are read from LFC IN.

There is no default assignment for IN.

There are two optional assignments for IN:

$ASSIGN IN TO {*pathname* | **DEV=***devmnc*}

pathname is the pathname of any valid MPX-32 file

DEV= *devmnc*
devmnc is the mnemonic of any device. If CTL is assigned to LFC SYC, IN cannot be assigned to LFC SYC.

8.2.2.1 Blocking Factor Choices

The MPX-32 blocking format is controlled by the choice of TSM/JCL assignment statement options, specifically the (,U) field, while the sort special blocking and direct access are controlled by the designation of input physical record size and output physical record size in the SORT parameters.

8.2.3 Secondary Inputs (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8)

Additional data elements to be sorted are read from these files after all elements have been read from LFC IN. The data element length is the same as used for the primary input LFC IN. Format choices are the same as for IN. Additionally, the TSM/JCL assignment statement can be used to select a different blocked/unblocked format choice from any other LFC.

There are no default assignments for IN1 through IN8.

8.2.4 Merge Inputs (MG1,MG2,MG3,MG4,MG5,MG6,MG7,MG8)

These LFCs are used to define additional pre-sorted data elements to be merged with the sorted combination of the data elements read from all the IN and IN*n* LFCs. These files can be assigned to a valid file or device. The data element length is the same as for the primary input LFC IN. Format choices are the same as for IN except that special blocking and direct access are not supported. Additionally TSM/JCL statements can be used to select a different blocked/unblocked format choice from any other LFC.

There are no default assignments for MG1 through MG8.

There are two optional assignments:

$ASSIGN MG*n* TO {*pathname* | **DEV=***devmnc*}

n is a number from 1 to 8

pathname is the pathname of any valid MPX-32 file

DEV= *devmnc*
devmnc is the mnemonic of any device

8.2.4.1 Order of Precedence for Merge Inputs

FSORT2 sorts and outputs the elements based on the sort parameters input through CTL. If FSORT2 encounters data elements that are equivalent, the program first outputs, in order, those elements from LFCs MG1 through MG4. FSORT2 then outputs elements from the sorted combination of the IN files, and then from LFCs MG5 through MG8. By selecting MG LFC assignments, users can indicate merge priority for the file's elements.

8.2.5 Primary Output File (OUT)

The sorted and merge selected data elements are written to LFC OUT in the proper order.

The output data elements can be in any of the formats available for input file data elements, and may be in a completely different format than the input format.

If option 1 has been indicated (input and output files the same), a rewind is initiated before the start of sorted output, otherwise the position of OUT is not altered before output starts. After all output is written, an EOF operation is performed.

There is no default assignment for OUT.

There are two optional assignments:

$ASSIGN OUT TO {*pathname* | **DEV=***devmnc*}

pathname is the pathname of any valid MPX-32 file

DEV= *devmnc*
devmnc is the mnemonic of any device

8.2.6 Listing File (LO)

FSORT 2 uses the LO file to log the input control statements that control the sort. FSORT2 also writes error messages, a count of the number of input records, and the number of output records to the LO file. LFC can be assigned to any valid file or device, including the null device.

The user can provide an assignment for LO. User reassignment of this device is optional. FSORT2 writes an EOF when all output to LO is complete.

There are two optional assignments:

$ASSIGN LO TO {*pathname* | **DEV=***devmnc*}

pathname is the pathname of any valid MPX-32 file

DEV= *devmnc*
devmnc is the mnemonic of any device

8.2.7 Designated Work File Devices (..1 and ..2)

These LFCs are dynamically allocated by FSORT2 to provide work space during its execution.

8.2.8 Work Files (WK1, WK2, .W1, .W2)

These LFCs are dynamically allocated by FSORT2 to provide the actual work space requirements for sorting during its execution.

8.3 LFC Summary

The following is a table of LFCs and their default and optional assignments.

Table 8-1
Sort/Merge LFC Summary

LFC	Default Assignment	Optional Assignment
CTL	SYC	Any valid file or device except the null device
IN	N/A	Any valid file or device. Must not be SYC if CTL is assigned to SYC.
IN1 - IN8	N/A	Same as IN
MG1 - MG8	N/A	Any valid file or device
OUT	N/A	Any valid file or device
LO	N/A	Any valid file or device
..1 and ..2	N/A	N/A
Wk1, Wk2, .W1, .W2	N/A	N/A

8.4 Input Files and Input Data Elements

The input data elements can be in one of the following data formats:

- standard blocked
- unblocked
- special standard blocked
- special unblocked
- direct access blocked
- direct access unblocked

8.4.1 Standard Blocked Format

This format provides a single element of sort input per MPX-32 logical record. This format is commonly used for creating data files using the STORE command in EDIT and formatted I/O operations of FORTRAN 77+, ASSEMBLE, etc. It is limited to a logical record length of 254 bytes and an element length of 254 bytes.

8.4.2 Unblocked Format

This format provides a single element of sort input per MPX-32 physical record. This format is commonly used for unblocked physical device access such as magnetic tape and direct card, terminal, and communications input. Record length is limited to 4095 bytes and is further limited by the device characteristics.

8.4.3 Special Standard Blocked Format

This format provides the capability for one or more elements of sort input data to be contained in one logical record. Since the logical record format is limited to 254 bytes, the data element to be sorted is typically limited to 1-100 bytes for any effective packing.

8.4.4 Special Unblocked Format

One or more elements of sort input data are contained in one physical record.

This format is typically used for a non-Encore blocked transfer or large data files on magnetic tape. The physical record length is limited to 4095 bytes and is further limited by the device characteristics.

8.4.5 Direct Access Blocked Format

This format provides the capability for more than one element of sort input data to be contained in one physical record. This format is commonly used for FORTRAN. Physical record size must not exceed 760 bytes and cannot equal the logical record size. Option 13 must be set. Refer to the Usage Notes section of this chapter for further information.

8.4.6 Direct Access Unblocked Format

This format provides a single element of sort input per physical record. This format is commonly used by FORTRAN users. Logical record length and physical record length must be equal. The record length is limited to 764 bytes. Option 13 must be set. Refer to the Usage Notes section of this chapter for further information.

8.5 Options

Options used by FSORT2 include indicating the use of the input file for output also, and the amount of main memory space to be used during execution.

Options are:

Option	Description
1	input and output files are the same. Rewind before output.
2-9	spare
10	variable length records to be output.
11	spare
12	merge only. Do not read from input files.
13	direct access
14-16	spare

8.6 Using Extra Memory With FSORT2

The JCL/TSM $ALLOCATE directive can be used to vary the operating performance of the sorting function. The static memory requirements cataloged with FSORT2 are sufficient to provide reasonable sorting performance for typical 80 byte logical records (average initial string length will be about 50-100 in this case) and minimal performance with larger (600 byte or larger) records. Using $ALLOCATE before invoking FSORT2 provides extra real memory for use in FSORT2 internal tables, buffers, and pointers, and generally increases performance and reduces the sort or merge time. If sufficient space is available to hold the complete input then FSORT2 performs an in-core sort and immediate output if no merge functions have been triggered. Larger logical or physical record sizing can also require the user to increase the JCL/TSM allocation. Merges of more than four files generally require an increased allocation.

Large memory allocation is not recommended for systems that have a small physical memory concurrent with high task activation, swapping, or other thrash-inducing activity.

8.7 Accessing FSORT2

8.7.1 Accessing FSORT2 in Utility Mode

To access FSORT2 as part of a batch job, create a job file using an editor. The job file can be read as SYC and the job activated using MPX-32 batch job techniques.

To activate FSORT2 and run interactively, use the TSM ASSIGN command to make assignments equivalent to those preceding the $FSORT2 command on a jobfile, then issue FSORT2 directives. (SELECT and OBJECT statements are not available when running FSORT2 interactively.)

```
TSM> $ASSIGN n...
              .
              .
              .
TSM> $OPTION n...
TSM> $FSORT2
FSO> directive
FSO> directive
FSO> <ctrl>C    (terminal EOF indication)
```

See the FSORT2 Directives section for the FSORT2 directives.

8.7.2 Accessing FSORT2 through Subroutine Calls

FSORT2 allows users to incorporate subroutines into their own ASSEMBLE or FORTRAN 77+ programs. These programs are usually written to achieve special reformatting or other processing either before or after the sort, to provide "canned", invariant sorts, or to provide sorts within a real-time program, without the necessity of connecting to a utility version.

The controlling program must ensure that all of the required input files are present before sort execution, and must read the sorted file stream produced by the sort.

The user must ensure that the catalog parameters for file and buffer space include Sort/Merge requirements.

There are four subroutine entry points available to the Sort/Merge subroutine package. Two of these entry points must be called in the proper order for successful operation, while the other two entry points are optional, as required, for the particular sort operation to be performed.

Errors detected by the Sort/Merge package are passed back in an error return variable described in one of the statements. Errors that are detected by the operating system cause a return to MPX-32.

8.7.2.1 CALL SORT:PAR (Sort Parameter Call)

This optional call must be made before any other entry point is called. This entry point changes the leading sort parameters from their default, or previously set values. This permits tuning of the Sort/Merge processing. The memory partition size field corresponds to the optional $ALLOCATE statement for utility execution.

Parameter Definition Subroutine — This subroutine is called once per sort to pass optional parameters. If it is called, it must be called before SORT:HDR to be effective.

Calling Sequence

CALL SORT:PAR(*imempart,iminwy,imaxwy,iminsc,imaxsc*)

imempart	is an integer value specifying the minimum memory partition requested
iminwy	is an integer value specifying the minimum number of merge ways
imaxwy	is an integer value specifying the maximum number of merge ways
iminsc	is an integer value specifying the minimum number of sectors per bucket
imaxsc	is an integer value specifying the maximum number of sectors per bucket

Example

The following example provides at least `40,000` bytes of memory pool, at least `2` merge ways but not more than `20`, and at least `4` sectors per intermediate sort bucket, but no more than `20` sectors per bucket for subsequent processing.

```
    CALL SORT:PAR(40000,2,20,4,20)
```

8.7.2.2 CALL SORT:HDR (Sort Header Call)

This required call must precede the SORT:FLD and the SORT:X calls. This entry point indicates record size and formatting to be used for subsequent Sort/Merge processing, and to pass an estimate of records to be input, and alternate sequencing desires. It corresponds directly to the H statement of the utility version. The IOPT field corresponds to the $OPTION statement used before the utility version execution. The meaning of each bit is shown.

Header Definition Subroutine — This subroutine is called once per sort.

Calling Sequence

CALL SORT:HDR(*lrl,irl,iorl,ienr,nsf,nfsum,ncsq,iopt,iecd*)

lrl	is an integer value specifying the logical record length in bytes
irl	is an integer value specifying the physical record length in bytes (input)
iorl	is an integer value specifying the physical record length in bytes (output)
ienr	is an integer value specifying the estimated number of input records
nsf	is an integer value specifying the number of sort fields (not used in FSORT2)
nfsum	is an integer value specifying the total sum of control field lengths (not used in FSORT2)
ncsq	indicates the normal collating sequence (IRA specifies ascending sort; IRD specifies descending sort)
iopt	is an integer value specifying the run option indicator

Option	Decimal No.	Bit No.	Explanation
0	0		no options
1	1	31	rewind before output
2	2	30	
3	4	29	
4	8	28	
5	16	27	
6	32	26	
7	64	25	
8	128	24	
9	256	23	strict sequencing
10	512	22	variable length records
11	1024	21	
12	2048	20	merge only
13	4096	19	direct access
14	8192	18	
15	16384	17	
16	32768	16	

iecd	is an integer value specifying the returned error code (1-99) or 0

Example

The following example specifies a logical record length of `89` bytes, physical record length for input and output as `89` bytes, an estimate of `1000` input records, and ascending sort. Options 9 and 10 are set using decimal indications.

```
OPTN=512+256 !$OPTION 9 10
CALL SORT:HDR(89,89,89,1000,0,0,IRA,OPTN,IERR)
```

8.7.2.3 CALL SORT:FLD (Sort Field Definition Call)

This optional call orders and collates the sequence for each field that is to be used as a key for sorting. This entry point can be called as many times as required. One sort key is passed with one CALL and the order of comparison is from major (first CALL) to minor (last CALL). This CALL corresponds directly to the F statement of the utility version.

Field Definition Subroutine — This subroutine is called as many times as there are fields to be sorted. If called, it must be called after SORT:HDR(ETC).

Calling Sequence

CALL SORT:FLD(*ityp*,*isip*,*ieip*)

ityp	is an integer sort type indicator (IRN, IRC, ETC)
isip	is an integer starting column number for this field
ieip	is an integer ending column number for this field

Example

The following example specifies ascending sort in columns `147` through `150`, descending sort in columns `411` through `418`, and describes binary halfword data in columns `6` and `7`.

```
CALL SORT:FLD (IRN,147,150)
CALL SORT:FLD (IRO,411,418)
CALL SORT:FLD (IRC,6,7)
```

8.7.2.4 CALL SORT:X (Sort Execution Linkage Call)

This required call indicates that all sort parameters and fields have been defined to Sort/Merge, that all files have been mounted and are ready, and that the sort (or merge) is to be performed now. Any errors detected are returned to the variable used in the CALL SORT:HDR statement. Upon successful or unsuccessful completion of the sort/merge, the subroutine returns to the next FORTRAN statement in sequence. This call corresponds exactly to the EOF at the end of LFC CTL in the utility version.

This subroutine is called once per sort.

Calling Sequence

CALL SORT:X

Example

The following example shows the calling sequence for a sort that is implemented as a subroutine. The file to be sorted is placed into LFC TG. When the sort is performed, the result is placed into an LFC OUT and is then routed for later printing.

```
C NOW SORT THE DATA SET THAT WAS GENERATED
C
.
.
.
  CALL SORT:PAR(20000,4,20,4,20)
  IOPTN=0
  CALL SORT:HDR(30,30,120,0,0,0,1RA,IOPTN,IERR)
  IF (IERR .NE.  0) GOTO 800 !ERROR
  CALL SORT:FLD (IRN,1,12)
  IF (IERR .NE.  0) GOTO 800 !ERROR
  CALL SORT:X
  IF (IERR .NE.  0) GOTO 800 !ERROR
  .
  .
  .
```

8.7.2.5 Subroutine Version Differences

- The LFC CTL is not used.
- Dynamic DSECT memory requests are used to obtain all memory space for the sort operational buffers. Therefore, the calling program must leave sufficient DSECT/CSECT logical and physical address space for satisfaction of the DSECT requests.
- Users must open the OUT and LO files.
- The catalog directives for files and buffers must satisfy Sort/Merge requirements.

8.8 FSORT2 Directives

FSORT2 requires at least two directives for proper operation; one header directive record, and at least one field directive.

8.8.1 Header Directive

This directive must be the first directive in the file assigned to CTL. There must be only one header directive per sort. The header directive information must be in the proper columns for correct operation.

8.8.1.1 Header Directive Indicator

Column 1 must contain the letter H.

8.8.1.2 Logical Record Length

Columns 3 through 6 must contain the logical record length for the data element to be sorted. This is a decimal value, coded as four right justified numeric digits, i.e., 0080 not 80.

8.8.1.3 Physical Record Length of the Input

Columns 8 through 11 must contain the maximum physical record length on any input file LFC. This is a decimal value, coded as four right-justified numeric digits. It must be an exact multiple of the logical record length, but must also be less than 4095 bytes. When this value is not equal to the logical record length, special unblocking operations are enabled. For reading single logical record per physical record files, such as the typical MPX-32 blocked format, this field must equal the logical record length.

8.8.1.4 Physical Record Length of Output

Columns 13-16 must contain the maximum physical record length in bytes of the desired output file. This is a decimal value, coded as four numeric digits, right justified. It must be an exact multiple of the logical record length, but less than 4095 bytes. When this value is not equal to the logical record length, special blocking operations are enabled. For writing a single logical record per physical record file, such as the typical MPX-32 blocked format, this field must equal the logical record length.

8.8.1.5 Input Record Count Estimate

Columns 18 through 23 must contain the value of the estimated number of logical records to be read from devices or files whose length FSORT2 cannot estimate. This is a decimal value, coded as six right-justified decimal numeric digits. This field is necessary whenever data elements are read from any type of file:

- magnetic tape
- terminal devices
- SYC,SLO,SBO,SGO files

This field cannot contain any estimates for logical records being read from permanent or temporary disk files, as the program can usually estimate the number of records based upon the size of these files.

For sorts performed from mixed types of devices, this field should contain an estimate of all the records from the inestimable devices and files. FSORT2 adds to this the estimate for the estimable files. The total estimate, from this field and other information, is multiplied by 1.1 for insurance. The total estimated length is used for sizing temporary file space.

In the event of an insufficient work file space abort (SS41), this field can be used to force FSORT2 to pre-allocate space that it might not otherwise supply by assigning one of the alternate input LFCs (IN1 through IN8) to the null device (e.g. `$AS IN5 TO DEV=NU0000`) and placing a higher count in the estimated records field.

8.8.1.6 Spare Fields

Columns 25 and 27-28 are ignored.

8.8.1.7 Normal Basic Collating Sequence

Column 30 must contain an indicator for the basic collating sequence interpretation. The character D in this column indicates that the normal collating sequence as defined on the field directives will be inverted for this sort/merge execution. The character A or any other character in this column indicates that the normal collating sequence is to be used.

8.8.2 Field Directives

One or more field directives must follow the header directive. The order of the field directives determines their priority in the sort comparisons; that is, the first field directive indicates the most major sort field, while the last directive indicates the most minor field.

8.8.2.1 Field Directive Indicator

Column 1 must contain the letter F.

8.8.2.2 Comparison Indicator

Column 3 must contain an alphanumeric character according to the following:

N Normal unsigned ascending sequence. This corresponds to the normal internal collating sequence for ASCII; therefore, upper case Z precedes lower case a, 0 comes before 9, 0 comes before A. See an ASCII code chart. This also handles EBCDIC data in the EBCDIC collating sequence. That is, Z precedes O and lower case z precedes upper case A. See an EBCDIC code chart.

O Reverse of N - that is, Z precedes A, 9 precedes 0, etc.

B Signed ascending sequence. The sign bit of the data in the starting column is interpreted, but is not interpreted for succeeding columns in this field. This type is useful for comparing integer computational or Encore normalized floating point values. For halfword data, the ending column should be 1 more than the starting column; for fullword data, 3 more; for doubleword data, 7 more. In this sequence, -5 precedes -1 precedes 0 precedes 50.

C Reverse of B - that is, +50 precedes 0 and -5.

8.8.2.3 Starting Column Number

Columns 5-8 must contain the character position that is to be checked first in this field. The value must be a right-justified, decimal quantity (0051 not 51) ranging from 0001 through *nnnn*, where *nnnn* is the maximum logical record length defined in the header directive.

8.8.2.4 Ending Column Number

Columns 10-13 must contain the character position that is to be checked last in this field. The value must be a right-justified decimal quantity (0055 not 55) ranging from the value of the starting column number of this field through *nnnn* where *nnnn* is the maximum logical record length defined on the header directive.

8.8.3 Usage Notes

`0005 0003` is not permitted. `0081 0081` is not permitted for 80 character logical records.

Within each field, comparisons proceed from the starting column through the ending column.

There is no need to observe any normal machine boundary requirements in any one field - that is, a doubleword integer can be compared with:

```
F C 0002 0009
```

For multiple signed comparisons, separate field directives must be used. The following example obtains 2 separate signed word comparisons and results are quite different from the previous example.

```
F C 0002 0005
F C 0006 0009
```

Multiple field directives can be entered. FSORT2 uses a dynamic area of free space to hold field information. Over 600 fields with 2000 columns have been tested. Use ($)ALLOCATE directives for larger quantities.

Fields can overlap and can be redefined. The following example is permitted:

```
F N 0006 0095
F O 0006 0095
F C 0009 0010
F N 0010 0095
```

The order of presentation of field directives determines their relative sort precedence. The first field directive is the most major key, while the last field directive is the most minor.

For variable length record options, fields past the last data byte are not compared. When record lengths are different, a shorter record usually precedes a longer one. See the normal collating sequence field in the header directive.

The normal collating sequence indicator (A/D) in the header directive alters the sense of the comparison type in each field directive. This feature can be used to invert the sense of the field comparison type, the variable length record scan shut-off, and the strict sequencing option.

When option 13 is set, direct access mode is in effect. All input files and the OUT file are treated as direct access. Mixing of sequential and direct access files is not permitted. The user must indicate end-of-file by writing X'0F' in the first byte of the record immediately following the last record to be sorted. A special feature for users of the subroutine package only, enables the user to skip *n* records in the IN file. The word variable SM:OFFST is defined with the assembler DEF statement. The user can store the integer number of records to be skipped in that location prior to calling SORT:X.

8.9 Examples

8.9.1 Disk Resident MPX-32 Blocked Format Input Sorting

The following example demonstrates a batch job stream to sort a disk resident, MPX-32 blocked format file with an 80 character logical record length assumed. Output is directed to another named, pre-existing disk file. No listing is provided and the minimum memory size is used. The sort check sequence is column 50, 51, 52, 53, 54, 10, 11, 12

```
$JOB JOBNAME OWNER
$AS IN TO STIN
$AS OUT TO STOUT
$FSORT2
H 0080 0080 0080 000000      A
F N 0050 0054
F O 0010 0012
$EOJ
```

Notes:

For TSM the $ in column 1 need not be typed but the operator must enter a <ctrl>C to indicate EOF.

8.9.2 Tape Resident Unblocked Input with Tape Output

The following example demonstrates a batch stream to sort a tape resident, unblocked format file with a maximum 120 character logical record length with output to another similar tape. Extra memory is used and a listing goes to SLO. An estimated 40,000 records exist.

```
$JOB JOBNAME OWNER
$AS IN TO DEV=M91001 ID=INPUT BLO=N
$AS OUT TO DEV=M91000 ID=OUTS BLO=N
$AS LO TO SLO
$OPTION 10
$ALLOCATE 20000
$FSORT2
H 0120 0120 0120 040000      A
F N 0001 0120
$EOJ
```

8.9.3 Special Unblocking on Input with No Sorting Action

In the following example, the input to sort is an input tape containing specially blocked physical records of 800 bytes each. Each physical record contains 10 80-byte records. This is often referred to as 10:1 blocked 80-byte format in a data processing type environment. The output is a new, unblocked magnetic tape that is in exactly the same order.

```
$JOB JOBNAME OWNER
$AS IN TO DEV=M91000 ID=IBMI BLOC=N
$AS OUT TO DEV=M91001 ID=SVTA BLOC=N
$OPTION 9
$FSORT2
H 0080 0800 0080 010000      A
$EOJ
```

8.9.4 Special Unblocking on Input with Total Reversal

This example is the same as the preceding example except that output is totally reversed from the input. Note that the total reversal is otherwise controlled by the header card.

```
$JOB JOBNAME OWNER
$AS IN TO DEV=M91000 ID=IBM1 BLO=N
$AS OUT TO DEV=M91001 ID=SYTB BLO=N
$OPTION 9
$FSORT2
H 0080 0800 0080 010000      D
$EOJ
```

8.9.5 Direct Access

In the following example, input to sort is an unblocked direct access file. Output is a blocked direct access file containing 11 records.

```
$JOB JOBNAME OWNER
$AS IN TO SRT.IN BLOC=N
$AS OUT TO SRT.OUT
$OPTION 13
$FSORT2
H 0060 0060 0660             A
F N 0033 0037
$EOJ
$$
```

8.9.6 Errors and Aborts

For a list of the FSORT2 and subroutines error and abort codes, see Appendix C.

9 Online Help

9.1 General Description

Online help displays documentation on requested topics on the terminal. A wide range of topics is available to all users.

Online help can be accessed through TSM, or during execution of a task by pressing a predefined help key. Once you invoke online help, further documentation is available for display. When you exit online help, you are returned to your point of interruption.

In addition to the documentation provided, you can modify, add, or delete the documentation to suit your requirements.

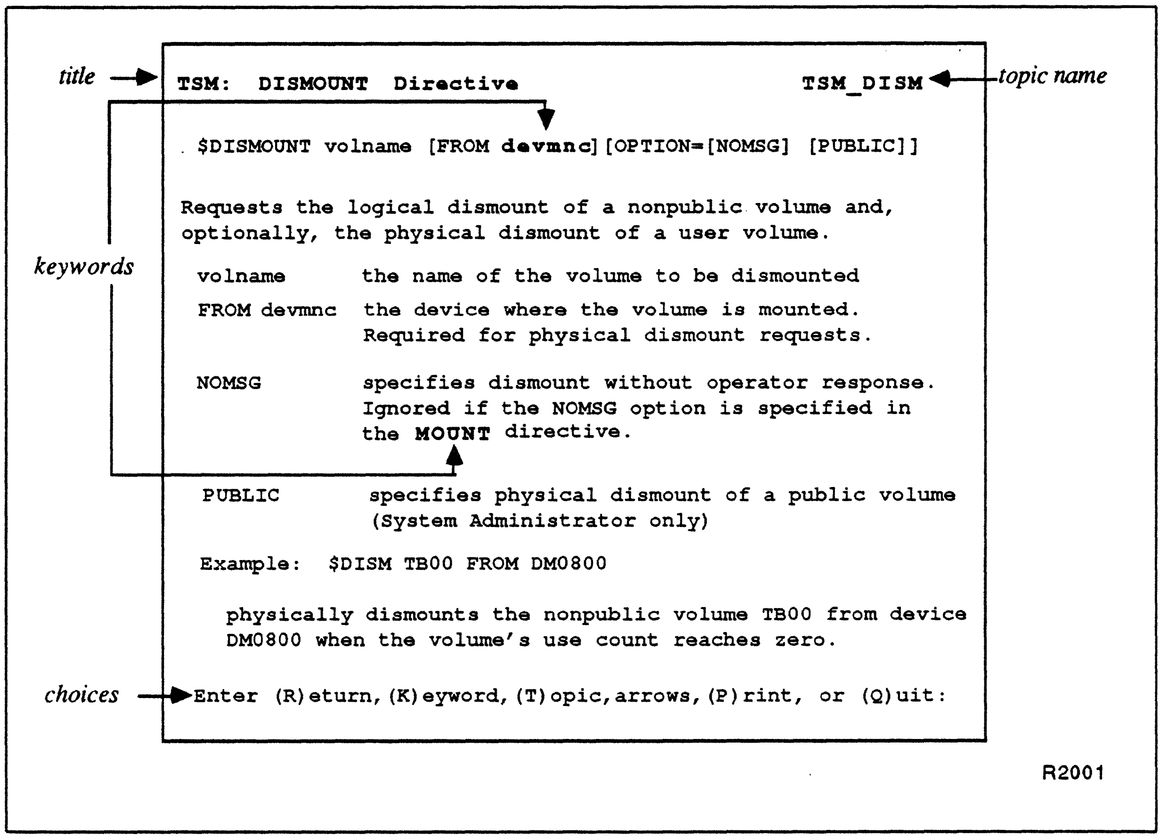
9.2 Components

Online help consists of the following:

HELP	the task used to request information to be displayed on the terminal screen
help files	files containing the documentation that displays when online help is invoked
J.HLP	the task that displays the requested information on the screen
HELPT	the task that translates files created with an ASCII editor into files that are usable by J.HLP

9.3 Display

When online help is invoked, documentation is displayed on the terminal as shown below.



```
TSM:  DISMOUNT  Directive                                TSM_DISM

 $DISMOUNT volname [FROM devmnc][OPTION=[NOMSG] [PUBLIC]]

Requests the logical dismount of a nonpublic volume and,
optionally, the physical dismount of a user volume.

  volname       the name of the volume to be dismounted

  FROM devmnc   the device where the volume is mounted.
                Required for physical dismount requests.

  NOMSG         specifies dismount without operator response.
                Ignored if the NOMSG option is specified in
                the MOUNT directive.

  PUBLIC        specifies physical dismount of a public volume
                (System Administrator only)

  Example:  $DISM TB00 FROM DM0800

    physically dismounts the nonpublic volume TB00 from device
    DM0800 when the volume's use count reaches zero.

 Enter (R)eturn, (K)eyword, (T)opic,arrows, (P)rint, or (Q)uit:
```

Figure 9-1
Sample Online Help Display

Each topic of information that displays at the terminal is called a topic entry. Most topic entries fit within the default screen display size of 23 lines.

Each topic entry has a unique name which displays at the upper right of the terminal. The topic name consists of a general topic prefix, an underscore, and a unique entry suffix. For more information about this naming convention, refer to section 9.5.8.

A topic entry can contain one or more keywords. Keywords provide a way to request additional information related to the topic entry currently displayed. Keywords are highlighted or listed below the documentation on the terminal. Selecting a keyword displays another topic entry.

At the bottom of each display are your choices within online help. See section 9.5 for an explanation.

9.4 Access

Access online help by either of the following methods.

9.4.1 From TSM

Access online help from the TSM prompt by entering `HELP` and optionally a topic name:

    TSM>HELP [*topicname*]

Information on the requested topic is displayed. When no topic name is specified, the topic entry containing the top level menu for online help (TOP_MENU) is displayed. This menu is shown below and offers the keywords (in the left-hand column) that can be selected for further information:

```
Top Level Menu for Online Help                           TOP_MENU

Select one of the following:

MPX             MPX-32 Operating System Information
TOOLS           MPX-32 Tools and Utilities
LANGUAGES       Compilers and Compiler Libraries
COMMUNICATIONS  Communication Products

Enter (R)eturn,(K)eyword,(T)opic,arrows,(P)rint, or (Q)uit:
```

9.4.2 Using the Help Key

Press the predefined help key (<help>) at any time to access online help documentation for the current interactive task. The default help key defined in the system MPX.PRO file is <ctrl>P. See Chapter 10 for information on modifying that assignment.

For example, to display the online help documentation for the TSM ASSIGN directive while in the process of specifying an ASSIGN statement, press <help> anywhere on the command line; such as:

```
TSM>$ASSIGN OUT TO<help>
```

When you exit online help, your ASSIGN statement and cursor are displayed as they were before pressing <help>.

Exception

User tasks issuing data format inhibit (DFI) reads cannot invoke online help through the help key. These tasks must send a message request to J.HLP and provide a message request receiver. J.HLP services the terminal that the message specifies and then returns a message request to the routine when finished. If this is not established, online help is not available while running the task.

The message request is sent using the M.SMSGR service. The message receiver is established using the M.RCVR service. Use the following data structure for the message:

word 0, byte 0	message type (must be the value 0)
word 0, bytes 1-3	reserved
word 1	the right justified, zero filled hex device channel and subaddress of the terminal to be serviced by J.HLP (e.g., 7EA0)
words 2-5	the topic name of the requested information, left justified and blank filled
words 6-9	user's current volume name, left justified and blank filled
words 10-13	user's current directory name, left justified and blank filled
words 14-15	user's owner name, left justified and blank filled
words 16-17	user's project name, left justified and blank filled

After J.HLP has serviced the terminal, a two word status message is returned as follows:

word 0, byte 0	0 - normal return – no errors occurred 1 - invalid message type received 2 - terminal's channel and subaddress were not identified in J.HLP's terminal context area (TCA) 3 - help is offline
word 0, bytes 1-3	reserved
word 1	the hex device channel and subaddress of the terminal serviced by J.HLP, right justified, zero-filled

For more information, refer to the M.SMSGR and M.RCVR services in Chapter 6 of Volume I of the reference manual.

9.5 Choices Within Online Help

The choices displayed at the bottom of the terminal while in online help allow you to display other topic entries, print any of the topic entries, or quit online help. These choices are described in alphabetical order in the following sections.

9.5.1 arrows

Use the keyboard arrows to move the cursor onto a highlighted keyword, then press <ret> to view information presented through that keyword.

Arrow support is available only when the TERMDEF 'cm' function is available and the arrow key strings are defined in the MPX.PRO file by the functions [left], [right], [up], and [down]. If MPX.PRO is not initialized, the defaults are <ctrl>H, <ctrl>L, <ctrl>K, and <ctrl>J, respectively. See Chapter 11 for information on TERMDEF and Chapter 10 for information on MPX.PRO.

If arrow support is not in effect, the keywords will appear at the bottom of the screen. If that is the case, you must use the (K)eyword choice to select a keyword.

9.5.2 (B)ack

Press `B` to display the previous screen of a multiscreen topic entry.

9.5.3 (F)orward

Press `F` to display the next screen of a multiscreen topic entry.

Pressing the space bar also performs this function. Pressing <ret> displays one additional line of the current multiscreen topic entry. These two choices are not listed on the choices line but are provided for compatibility with the original online help release.

9.5.4 (K)eyword

Press `K` to display a prompt requesting a keyword. Type the keyword as it appears on the help screen and press <ret>.

Keywords are local, defined only for the current topic entry. Each is a quick link to an entry related to the current topic. Choosing a keyword adds the current topic entry to the topic stack and displays the requested entry. This topic stack enables you to quickly return to a previous topic entry by selecting the (R)eturn choice discussed in section 9.5.7. The stack holds a maximum of 32 topic entries. When the limit is exceeded, an error message is displayed. See section 9.7.3 for further discussion.

If arrow support is not in effect, the keywords are listed at the bottom of the screen.

9.5.5 (P)rint

Press `P` to display a menu screen enabling you to print a topic entry.

The following menu is displayed:

```
Print Topic Menu for Online Help

   Total Topics Displayed: nnnnn

Defaults:
   OUTPUT sent to: SLO
   NUMBER the lines: No


DISPLAYED_TOPICS    Print Last n Topics Displayed
TOPIC_NAME          Print by Topic Name

Enter (R)eturn,(K)eyword,arrows, or (Q)uit:
```

To print a topic entry, first set the values under `Defaults:` to the desired values before indicating which topics to print.

To modify the defaults, move the cursor to the keyword for the item you wish to change and press <ret>, or select (K)eyword and enter the keyword followed by <ret>.

To send output to a file instead of SLO, select `OUTPUT`. A prompt requesting the name of a permanent file you wish to send the output to is displayed:

```
Pathname?
```

At this prompt, enter the pathname of the file to contain your output followed by <ret>. The entry on the Print Topic Menu changes from `SLO` to your specified pathname. If at the `Pathname?` prompt you decide not to redirect output, press <ret> only. The choices line is redisplayed and no change is made.

If the file you specified exists and you have write access, the following prompt is displayed:

```
Specified file already exists.  Overwrite it (Y/N,[Y])?
```

Enter `Y` or <ret> to overwrite the file. Enter `N` to redisplay the `Pathname?` prompt.

If the file exists and you do not have write access, the following message is displayed:

```
Specified file exists. Cannot overwrite. Any key to continue.
```

The `Pathname?` prompt is redisplayed after pressing any key.

Each time the keyword `NUMBER` is selected, the value toggles between `Yes` and `No`. Specify `Yes` to insert line numbers in columns 1 and 2 of the output; specify `No` to supply output without line numbers.

To specify printing a number of previously displayed topic entries, select the keyword `DISPLAYED_TOPICS`. The following prompt is displayed:

```
    Number of topics to print [1]?
```

Specify a number representing the last *n* topic entries displayed followed by <ret>. Pressing only <ret> indicates the current topic entry only (the default). Entering `0` followed by <ret> redisplays the choices line and no action is taken. If you enter a number greater than the number of topic entries displayed during the current online help session (shown at the `Total Topics Displayed` heading), the following prompt is displayed:

```
    Number exceeds total topics. Print total (Y/N,[N])?
```

Enter `Y` to print the total number of topic entries. Enter `N` or <ret> to redisplay the choices line without selecting a number.

To specify printing a particular topic entry by supplying the topic name, select the keyword `TOPIC_NAME`. The following prompt is displayed:

```
    Topic Name?
```

Enter the name of the topic entry you wish printed followed by <ret>.

Each additional topic entry selected for printing is concatenated to the previously selected topic entries throughout the current online help session. Printing of the selected topic entries occurs when (Q)uit is selected. SLO is spooled for output at the printer or, if a permanent file was specified, the file is closed. Each printed topic entry is separated by a form feed (new page).

If a file is created, the owner and project for the file are the same as the user's current owner name and project name. If an existing file is specified, the owner and project do not change.

The file may be printed using any available print program. To print the file using the LASER or SPRINT programs, specify `Y` to the `Embedded Forms Control` prompt. The Toolkit LPR program may also be used. The $PRINT directive may output incorrect page breaks if sent to a serial printer handled by H.F8XIO with a page size other than zero.

9.5.6 (Q)uit

Press `Q` to exit online help.

If you invoked online help from the TSM prompt, you will return to a TSM prompt. If you used the help key to invoke online help, you will be returned to the command line and cursor position in effect before <help> was pressed.

9.5.7 (R)eturn

Press `R` to redisplay the previous topic entry.

When keywords are selected to display related topic entries, a stack of topic entry names is created and maintained. Each time `R` is selected, the previous topic entry is redisplayed. If the current topic entry is the first one on the stack, pressing `R` displays the top level menu (TOP_MENU) discussed in section 9.4.1.

9.5.8 (T)opic

Press `T` to display a prompt requesting a topic name. Enter a valid topic name followed by <ret>.

Each topic entry possesses a unique topic name. Each topic name has three components: a general topic prefix, an underscore character, and a unique entry suffix. For example, the topic name for the TSM $ASSIGN directive topic entry is TSM_$ASSIGN; TSM is the general topic prefix and $ASSIGN is the unique entry suffix. Topic names are displayed at the top right of the terminal screen.

If you enter a string that does not contain an underscore character, J.HLP assumes this string is an entry suffix and adds to it the current entry's prefix and an underscore. If this new string matches an existing topic name, its topic entry is displayed.

A specific topic entry can be requested from any online help display whether or not it is related to the current display.

When keywords are selected to display related topic entries, a stack of topic entry names is created and maintained. Selecting `T` and providing a valid topic name clears this stack and displays the requested topic entry.

9.6 Modifying Online Help Information

9.6.1 Description

Online help information can be modified by changing existing topic entry text or creating new topic entries.

Topic entries reside within standard ASCII text files. These text files are then processed by the online help translation program, HELPT. HELPT inserts information J.HLP uses to display the proper topic entry on the terminal when requested. After translation, these files are called help files.

To modify the online help information, you can do any of the following:

- modify existing topic entries by making a copy of the help file containing those topic entries and making the changes
- create new topic entries by:
  - making a copy of the help file containing related information and adding new topic entries
  - creating a new file to contain new topic entries

Use any ASCII editor that produces blocked and uncompressed records for lines (e.g., EDIT stored unnumbered or write formats or Programmable Editor (PE)). Follow the conventions described in section 9.6.3 when modifying or creating topic entries. Then translate the file following the steps described in section 9.6.4.

9.6.2 Help Files

Help files have the following characteristics:

- blocked file
- file type FD (assigned by HELPT)
- reside in the system help directory, which by default is @SYSTEM^HELP
- file names ending in `.HLP`

The help files included with MPX-32 and their contents are:

DS.HLP	data structures
GMPX.HLP	general MPX help including online help, command line recall and edit, miscellaneous tables, and error codes
PROC.HLP	information on the processors (TSM, OPCOM, VOLMGR) and their directives
SAT.HLP	system administrator tools
SVC.HLP	system services
TOP.HLP	top level menu topic entries (TOP_MENU)

Additional help files exist and may be installed on your system. To avoid complications, do not duplicate help file names.

9.6.3 Topic Entry Requirements

Topic entries are made up of topic name lines, text lines, keyword lines, and comment lines. These lines (except text lines) begin with a dot command, consisting of a period in column one followed by the command designation. These dot commands are:

.TN	specifies a topic name line
.KW	specifies a keyword line
.<space>	specifies a comment line

Two additional dot commands are inserted in help files when they are translated by the HELPT task. Ignore these when editing help files. They are:

.TL	specifies a topic list
.PI	specifies cursor position information

Any line not beginning with one of the dot commands is a text line.

There must be no lines except comment lines between the last keyword line (.KW) of a topic entry, if provided, and the first topic name line of the next topic entry (.TN). An error message is displayed during translation if this occurs and HELPT aborts. See section 9.7 for a discussion on error messages.

For best results, whenever possible confine the topic entry to 23 lines of text. This is the maximum number of lines that are displayed on the terminal when online help is requested; fewer lines are displayed if the current value for the number of lines of screen display (page size) is less than 23 lines. The display includes the topic name and title which occupy the topmost line on the terminal display. The choices line at the bottom of the terminal display is inserted by J.HLP automatically and is not included in the line count. If a topic entry display exceeds the current page size, the number of lines defined for the page size is displayed and `(F)orward` is included on the choices line. Pressing `F` displays additional lines of the topic entry with `(B)ack` included on the choices line.

Figure 9-2 is a sample topic entry. The sections that follow describe the contents.

```
topic name  .TN TSM_DISM TSM: DISMOUNT Directive
  lines     .TN TSM_DISMOUNT TSM: DISMOUNT Directive
            .TN TSM_$DISM TSM: DISMOUNT Directive
            .TN TSM_$DISMOUNT TSM: DISMOUNT Directive

             $DISMOUNT volname [FROM devmnc~] [OPTION=[NOMSG] [PUBLIC]]

            Requests the logical dismount of a nonpublic volume and,
   text     optionally, the physical dismount of a user volume.
  lines
             volname       the name of the volume to be dismounted

             FROM devmnc   the device where the volume is mounted.
                           Required for physical dismount requests.

             NOMSG         specifies dismount without operator response.
                           Ignored if the NOMSG option is specified in
                           the MOUNT directive.

             PUBLIC        specifies physical dismount of a public volume
                           (System Administrator only)

            Example:  $DISM TB00 FROM DM0800

               physically dismounts the nonpublic volume TB00 from device
               DM0800 when the volume's use count reaches zero.

  keyword   .KW devmnc~ MT_DTC
   lines    .KW MOUNT TSM_MOUNT
  comment   . ***************************
   line
                                                                  R2002
```

Figure 9-2
Sample Topic Entry

9.6.3.1 Topic Name Line (.TN)

The first line of a topic entry is the topic name line. The topic name line provides a unique identifier for the topic entry. When online help is invoked, J.HLP searches the topic name lines and displays the requested topic entry. Each topic entry must have at least one topic name line. A separate topic name line can exist for each request possibility (see Example below). The topic name line is prefaced with `.TN` followed by a space.

Syntax

.TN *topicname* [*title*]

topicname is a string consisting of 1 to 16 nonblank characters identifying the topic entry. Separate .TN and *topicname* with a single space. If the topic entry is to be displayed through use of the help key, *topicname* must contain an underscore separating the general topic prefix from the unique entry suffix. The general topic prefix should be the name of the task at which the help key is pressed. For example, the topic name OPCOM_LIST is designed to display the topic entry for the OPCOM LIST directive when the user presses <help> while in OPCOM attempting to execute the LIST directive. *topicname* appears at the upper right of the topic entry display (see Figure 9-1 for a sample display).

[*title*] is an optional 1- to 60-character title for the topic entry. Separate *title* from *topicname* with a single space or tab. *title* appears at the upper left of the topic entry display (see Figure 9-1 for a sample display).

Example

The following example shows the four topic name lines provided for the topic entry on the TSM CHANGE directive. This causes the topic entry to display when help is requested for `TSM_CHAN`, `TSM_CHANGE`, `TSM_$CHAN`, or `TSM_$CHANGE` or by pressing the help key during execution of the TSM CHANGE directive using any of CHANGE's valid invocations.

```
.TN TSM_CHAN TSM: CHANGE Directive
.TN TSM_CHANGE TSM: CHANGE Directive
.TN TSM_$CHAN TSM: CHANGE Directive
.TN TSM_$CHANGE TSM: CHANGE Directive
```

9.6.3.2 Text Line

Text lines follow topic name lines in a topic entry. Any line within a topic entry not prefaced with one of the dot commands listed in this section is a text line. The text lines constitute the information displayed when the topic entry is requested.

Text lines can be no longer than 79 characters. Use tabs, spaces, and blank lines to position the text for best display on the terminal.

9.6.3.3 Keyword Line (.KW)

Keyword lines follow text lines in a topic entry. The keyword line provides the mechanism for displaying additional information related to the topic entry currently displayed.

A keyword is a character string within the topic entry text indicating additional information is available when the keyword is selected. The keyword line provides J.HLP with the name of the topic entry to display. The keyword line is prefaced with `.KW` followed by a space.

Keywords within a topic entry are optional. They are supplied only when additional related information is available. More than one keyword can be supplied, each with its own keyword line.

Syntax

.KW *keyword topicname*

keyword is a string of 1 to 16 characters embedded within topic entry text and surrounded by any combination of the following delimiters:

blanks		slashes	/
parentheses	( )	backslashes	\
brackets	[ ]	commas	,
braces	{ }	colons	:
double quotes	"	semicolons	;
single quotes	'		

or followed by:

period	.
question mark	?
exclamation	!

Separate .KW and *keyword* with a single space. *keyword* is not case sensitive; therefore, do not supply identical keywords with different capitalization within a single topic entry. If *keyword* occurs more than once within the topic entry text and you wish to highlight only specific occurrences during display, indicate which of those occurrences to highlight by following them with a tilde (~) within the text and on the keyword line. This is illustrated in Figure 9-2. The tilde does not cause text realignment when the topic entry is displayed. If the tilde is not used to specify highlighting of a specific keyword occurrence, all occurrences of *keyword* within this topic entry highlight.

topicname is a string consisting of 1 to 16 nonblank characters. *topicname* is a valid topic name provided on a .TN line for the topic entry to be displayed when *keyword* is selected.

9.6.3.4 Comment Line (.<space>)

A comment line is used to insert text within a help file that does not display when online help is requested (e.g., a separation between topic entries for readability within the file). Comment lines are prefaced with a dot immediately followed by a space. They can appear anywhere within the file.

Syntax

. *comment*

comment is text used for clarification within the help file but not displayed when online help is invoked

9.6.4 Translation

Whenever topic entries have been created or modified, the file they reside in must be translated into a help file.

Follow these steps interactively or in batch mode:

1. Assign input and output files for the HELPT translation task. The input file is the file containing the new or modified topic entries. The output file is the translated help file. For example:

   ```
   TSM>AS INP TO WORKFILE
   TSM>AS OUT TO @SYSTEM(HELP)NEW.HLP
   ```

   In this interactive example, `WORKFILE` is a newly created file or modified copy of an existing help file. Output is directed to a permanent help file in the system help directory, `NEW.HLP`. The help file must have a file name ending in `.HLP`.
2. Execute HELPT:

   ```
   TSM>HELPT
   ```

If an error message is displayed and HELPT aborts, refer to the Error Messages section of this chapter. If HELPT does not abort, continue with step 3.

The following steps require access to the OPCOM OFFLINE and ONLINE functions.

3. Take J.HLP offline:

   ```
   TSM>!OFFLINE HELP
   ```
4. Put J.HLP back online:

   ```
   TSM>!ONLINE HELP
   ```

When these steps are complete, the new information is ready for display.

9.7 Error Messages

9.7.1 Operator Console Messages

When a problem occurs during I/O to the help file or the terminal, the following messages are displayed on the operator's console. These messages are prefaced with the task name J.HLP.

`J.HLP: UNABLE TO USE HELP DIRECTORY.  ERROR VM`*nn*

J.HLP cannot reach the help directory specified at SYSGEN. J.HLP exits. *nn* is an error code returned by H.VOMM (see Appendix C for a description).

`J.HLP: UNABLE TO ASSIGN WITH AUTO OPEN` *filename*`.  M.ASSN ERROR` *nn*

filename is the name of a file in the help directory that cannot be allocated or opened at assign. *nn* is an error code returned by the M.ASSN service. See the M.ASSN system service description in Volume I, Chapter 6 of the MPX-32 Reference Manual for a description of the code.

`J.HLP: UNABLE TO ASSIGN TERMINAL` *tyid*`.  M.ASSN ERROR` *nn*

J.HLP encountered an error while trying to assign terminal *tyid*. Online help is not available to that terminal. *nn* is an error code returned by the M.ASSN service. See the M.ASSN system service description in Volume I, Chapter 6 of the MPX-32 Reference Manual for a description of the code.

`J.HLP: ERROR IN HELP FILE` *filename*`, MARKED OFFLINE`

An I/O error occurred while reading a help file or data corruption is detected. The file is marked offline. *filename* is the help file that J.HLP is trying to access. Retranslate the file to correct the error.

`J.HLP: ERROR DURING I/O TO TERMINAL` *tyid*

An I/O error occurred during a read or write to terminal *tyid*. Online help exits for that terminal and control returns to the point at which help was requested. When the problem causing the I/O error is corrected, J.HLP will work correctly.

`J.HLP: ** WARNING ** DUPLICATE TOPIC NAME:` *topicname*

topicname is a topic name that appears on more than one topic name line. If not corrected, it is unpredictable which topic entry will display when the duplicated topic name is requested. Edit the help file and make all topic names unique.

9.7.2 HELPT Error Messages

The following messages appear when problems are encountered during the translation of a help file. Output is to LFC SLO, which is assigned at catalog time to LFC=UT, or can be dynamically assigned.

`SEQUENCE ERROR: LINE` *n*
`ENCOUNTERED INPUT:` *input*
`EXPECTED INPUT(S):` *input*

HELPT encountered an input sequence that is not allowed. HELPT aborts. *n* is the line number in the file where the sequence error occurs. *input* is one or more of the following:

`.TN`	topic name line
`.KW`	keyword line
`COMM`	comment line
`.TL`	topic list line
`.PI`	position information line
`EOF`	end of file
`TEXT`	none of the above, therefore a text line

Edit the file, correct the problem, and retranslate.

`TOPIC NAME LINE ENCOUNTERED WITHOUT A TOPIC NAME IN LINE` *n*

A topic name line at line number *n* in the file does not contain a topic name. HELPT aborts. Edit the file, supply the topic name, and retranslate.

`KEYWORD LINE ENCOUNTERED WITHOUT A TOPIC NAME OR KEYWORD IN LINE` *n*

A keyword line at line number *n* in the file does not contain a topic name or a keyword. HELPT aborts. Edit the file, supply the missing parameter on the keyword line, and retranslate.

`KEYWORD` *keyword* `ON LINE` *n* `IS NOT EMBEDDED IN TOPIC TEXT`

A keyword line specifies a keyword not found in the topic entry text. *keyword* is the keyword defined in the keyword line on line number *n* in the file. This is a warning message. HELPT does not abort. This situation prevents keyword selection for *keyword* within this topic entry. Edit the file and match the keyword on the keyword line with a character string in topic entry text.

`DUPLICATE TOPIC NAME:` *topicname*

topicname is a topic name that appears on more than one topic name line. This is a warning message. HELPT does not abort. If not corrected, it is unpredictable which topic entry will display when the duplicated topic name is requested. Edit the help file and make all topic names unique.

9.7.3 Usage Messages

The following error messages appear on the terminal when problems occur while requesting or using online help.

`THE HELP FILE` *filename* `THAT CONTAINS` *topicname* `IS OFFLINE`

The user requested a topic entry stored in the help file *filename* that is currently offline. *topicname* is the requested topic name. Retranslate the file and put it online.

`HELP HAS BEEN OFFLINED BY THE SYSTEM OPERATOR`

J.HLP is currently offline. Online help is available when J.HLP is placed online.

`*ONLINE HELP IS NOT AVAILABLE.  J.HLP IS NOT ACTIVE.`

J.HLP has aborted and is currently not active. When J.HLP is reactivated, online help becomes available.

`HELP NOT AVAILABLE FOR TOPIC NAME:` *topicname*. `ANY KEY TO CONTINUE`

Information is not available for the requested topic entry, *topicname*. Respond to the prompt by pressing any key. The choices line is redisplayed.

`TOPIC STACK LEVEL EXCEEDED.  ANY KEY TO CONTINUE`

The topic stack, built by selecting keywords, has exceeded the maximum of 32 topic names. Press any key to redisplay the choices line. Using arrows or (K)eyword to request another topic entry will only redisplay the error message. Choose (T)opic to clear the stack and request a new topic, or (Q)uit to exit online help.

10 Command Line Recall and Edit

10.1 Introduction

Command line recall and edit provides functions for editing the current command line and for recalling previously entered command lines. Each function is assigned to a terminal key and can be invoked by pressing that key. By using edit functions, users can move the cursor across the command line, and insert, delete, or replace its characters. By using recall functions, users can recall up to 23 previous command lines, list the lines available for recall, and re-enter any available line. The edit functions and recall functions can be used independently, or in combination (i.e., to modify a recalled command line).

Command line recall and edit is available only to TSM TY devices.

Recall and edit functions are valid from any task that uses standard formatted task reads, such as TSM, OPCOM, or VOLMGR. The function keys cannot be used from a task in any of the following conditions:

- with the NO ECHO bit set
- with data formatting inhibit set (e.g., the Programmable Editor)
- issuing reads in no-wait mode

All recall and edit functions assume that the linesize defined at SYSGEN is the actual terminal linesize and that the terminal is not set for autowrap.

The recall and edit keys are assigned in the MPX.PRO file and can be reassigned by the system administrator (SA) or user at any time. MPX.PRO is a source file that resides in the system directory and, optionally, in any user directory. It assigns the predefined recall and edit functions to specific terminal keys. The file also enables or disables command line recall and edit, sets the default edit mode, and specifies the maximum number of command lines available for recall.

The default keys and modes are established in the distributed MPX.PRO file that is maintained by the SA. However, users can customize a copy of MPX.PRO and place it in their default working directory to override the system assignments. A user MPX.PRO file can reassign keys, set new default modes, or disable command line recall and edit for a terminal.

Note: When using command line recall and edit, ASCII control characters not defined in the MPX.PRO file are ignored, with the exception of wakeup, break, and <ctrl>C. This means that pressing the control key and any character not defined in the MPX.PRO file has no effect.

10.2 Editing Command Lines

Using edit functions, users can edit the current command line for any task that issues standard formatted reads. Edit functions move the cursor across the command line in increments of a character, word, or the remaining part of a line. Other edit functions delete a character, word, or a part of a line, and change the edit mode.

Users can choose between two modes for editing: modify mode and insert mode. In modify mode, characters that are added to a line overtype any existing characters. In insert mode, any added characters shift the existing characters to the right. A default mode is established at logon, but users can switch modes at any time by using the mode toggle key or the modify or insert keys. The MPX-32 default mode is modify mode.

Command line edit defines a word as one or more characters surrounded by valid MPX-32 delimiters. It treats multiple consecutive blanks as a single delimiter. For a list of the valid MPX-32 delimiters, see the JCL chapter in this manual.

10.2.1 Edit Functions

The edit functions are assigned to keys in the MPX.PRO file. A system MPX.PRO file, maintained by the SA, assigns default edit keys that are available to all users of valid terminals on the system. However, users can reassign the keys to customize command line editing to their own keyboard layout. See the MPX.PRO section for information on changing the key assignments and default edit mode.

To determine the default MPX-32 edit key assignments for a particular terminal, match each ASCII character code listed in Table 10-1 to the terminal key that generates that code (listed in the terminal documentation). The matching key is the default key for the edit function.

Table 10-1
Edit Functions and the ASCII Character Codes

Action		Edit Function	Default Televideo 9220 ASCII Code	Corresponding Televideo 9220 Key Sequence
Move cursor	left 1 character	[left]	X'1B5B44' *	<esc> [D
	right 1 character	[right]	X'1B5B43' *	<esc> [C
	to beginning of word	[back word]	X'11'	<ctrl> Q
	to next word	[next word]	X'17'	<ctrl> W
	to beginning of line	[begin line]	X'02'	<ctrl> B
	to end of line	[end line]	X'05'	<ctrl> E
Delete	character	[delete char]	X'7F'	<del>
	word	[delete word]	X'12'	<ctrl> R
	to end of line	[erase end line]	X'04'	<ctrl> D
	left 1 character	[backspace]	X'08'	<ctrl> H
Insert/Modify	insert/modify toggle	[insert toggle]	X'14'	<ctrl> T
	insert mode	[insert mode]	X'0E'	<ctrl> N
	modify mode	[modify mode]	X'01'	<ctrl> A

* Key assignment must be specified with a TERMDEF 2-character control code.

10.2.1.1 Using the Edit Functions

Any time that a current command line from a valid task is displayed, users can:

- delete all or part of the line. When the cursor is positioned under a character, [delete char] deletes that character. [delete word] deletes the word and right delimiter at the current cursor position. [erase end line] deletes from the cursor to the end of the line. [back space] deletes one character and shifts remaining text left as it backspaces.
- move forward or backward across the line. There are six edit functions that move the cursor. [left] moves the cursor to the previous character. [back word] moves the cursor to the beginning of the current or previous word and [begin line] to the first character of the line. [right] moves the cursor right one character. [next word] moves the cursor to the beginning of the next word and [end line] to the last character of the line. See Table 10-1.
- insert characters in the line. In insert mode, users insert characters in the command line simply by positioning the cursor and typing. The new characters shift the previous characters to the right without destroying them. In this mode, the tab key inserts blanks from the cursor to the next tab position, shifting any characters to the right. [insert toggle] changes between insert mode and modify mode. [insert] establishes insert mode.
- modify characters in the line. In modify mode, users modify the command line by positioning the cursor and typing over characters. The new characters replace any characters that they overstrike. In this mode, the tab key moves the cursor to the next tab position without displacing characters. [insert toggle] changes between insert mode and modify mode. [modify] establishes modify mode.

10.3 Recalling Command Lines

Using recall functions, users can display and, optionally, re-enter one or more command lines. Any recent command line can be recalled if it was entered interactively for a task performing standard formatted reads. This includes TSM, VOLMGR, and OPCOM commands. The recall functions invoke recall mode, list up to 23 of the most recent command lines, re-enter selected command lines, and exit recall mode.

Command line recall stores each command line entered at the terminal in a buffer that holds up to 23 lines. Null lines and $RECALL commands are not saved in the buffer. The recall buffer line size is the same as the line size defined at SYSGEN for the user's terminal. Changing the terminal line size with the TSM $LINESIZE command does not change the recall buffer line size.

Once the buffer is full, entering a new command line causes the buffer to discard its oldest stored line. By doing so, the buffer maintains only the most recently entered command lines. Every command line in the buffer is available for recall.

When users enter recall mode (in response to a standard formatted task read), the most recently entered command is recalled and displayed. Users can then use recall and edit keys to either modify the line, enter it, or ignore it and display the next line in the buffer. If a command line is entered from the buffer, it is used as input for the task issuing the read. At any time, the user can exit recall mode and resume the current task.

Note: Command line recall cannot be invoked from a task issuing reads in no-wait mode, with data format inhibit set, or with the NO ECHO bit set.

10.3.1 Recall Functions

The recall functions, like the edit functions, are assigned to keys in the MPX.PRO file, which also sets the size of the recall buffer. The system MPX.PRO file assigns default keys that are available to all users of valid terminals on the system. However, users can reassign the keys to customize command line recall to their own keyboard layout. See the MPX.PRO section for information on changing the key assignments or buffer size, and disabling command line recall and edit.

To determine the default MPX-32 recall keys for a particular terminal, match each ASCII character code listed in Table 10-2 to the terminal key that generates that code (listed in the terminal documentation). The matching key is the default key for the recall function.

Table 10-2
Recall Functions and the ASCII Character Codes

Action		Recall Function	Default Televideo 9220 ASCII Code	Corresponding Televideo 9220 Key Sequence
Invoke	recall mode	[up]	X'1B5B41'*	<esc> [A
Display	next older command	[up]	X'1B5B41'*	<esc> [A
	next more recent command	[down]	X'1B5B42'*	<esc> [B
List	commands in buffer	[display]	X'07'	<ctrl> G
Exit	recall mode	[exit recall]	X'15'	<ctrl> U
Enter	the recalled command	[return]	X'0D'	<ctrl> M

* These key assignments must be specified with TERMDEF 2-character control codes.

10.3.1.1 Using the Recall Functions

To enter recall mode, press the [up] key. The most recent command line is recalled and displayed. The $RECALL directive may also be used to recall a specific command or range of commands (see the $RECALL directive description in Chapter 1 of this manual for more information).

After entering recall mode, users can:

- list all command lines in the buffer with [display]. The buffer list displays, followed by the unchanged current command line.
- display individual command lines in the buffer with [up] or [down]. If the oldest command line in the buffer is displayed, [up] has no effect. The line remains displayed to signal that it is the end of the buffer. If the most recent command line is displayed, [down] exits recall mode.
- edit the currently recalled command line by using the edit functions. See Table 10-1 for a list of the edit functions.
- re-enter any currently displayed command line with [return]. The line is used as input for the task issuing the read. After the line is entered, the buffer displays its next most recent command line. If the line entered was the most recent in the buffer, recall mode is exited.

To exit recall mode, press the [exit recall] key. If the most recent command line in the buffer is displayed, [down] or [return] also exits recall mode. After exit, the current command line displays unchanged.

10.4 MPX.PRO File

MPX.PRO is a source file that assigns predefined MPX-32 functions to specific terminal keys (see Table 10-3). The file assigns keys for command line recall and edit functions, and for online help. It also sets the default edit mode for command line edit, enables command line recall and edit, and sets the size of the recall line buffer.

Table 10-3
MPX-32 Predefined Functions

Feature	Function	Definition
Command Line Edit	[left]	Moves cursor left 1 character
	[right]	Moves cursor right 1 character
	[back word]	Moves cursor to beginning of word
	[next word]	Moves cursor to beginning of next word
	[begin line]	Moves cursor to beginning of line
	[end line]	Moves cursor to end of line
	[delete char]	Deletes a character
	[delete word]	Deletes word at cursor
	[erase end line]	Erases to end of the line
	[back space]	Deletes and shifts remaining text left as it backspaces
	[insert toggle]	Toggles between insert mode and modify mode
	[insert mode]	Establishes insert mode
	[modify mode]	Establishes modify mode
Command Line Recall	[up]	Accesses recall mode or displays next older command
	[down]	Displays next more recent command
	[display]	Displays list of commands in buffer
	[exit recall]	Exits recall mode
	[return]	Enters the current command line
Online Help	[help]	Accesses Online Help facility

A system MPX.PRO file resides in the system directory and assigns default recall and edit keys and an online help key. The keys assigned in this file are available to all users of valid terminals on the system. See Tables 10-1 and 10-2 for these default key assignments.

If desired, users can copy the MPX.PRO file to their default working directory and change its key assignments or default modes. When a user logs on, MPX-32 initializes the user MPX.PRO file's assignments for keys, buffer size, and default edit mode. If a user MPX.PRO does not exist, MPX-32 searches the system directory for the system MPX.PRO file and initializes its assignments. If an MPX.PRO file is not found, then command line recall and edit is disabled and only the functions available in releases prior to MPX-32 3.4 are available.

Optionally, users can disable all recall and edit functions for their owner name by setting the length of the recall buffer to zero in their user MPX.PRO file. See the Recall Buffer section for more information.

10.5 Customizing an MPX.PRO File

The MPX.PRO file assigns functions to keys by associating an MPX-32 function to one of the following:

- a special character (any ASCII character code generated by a terminal keystroke)
- a Terminal Definition Facility (TERMDEF) 2-character control code (a terminal function defined through TERMDEF for a number of terminal types).
- a hexadecimal value
- an octal value

Any TERMDEF 2-character control code assigned to an MPX.PRO function must first be defined in the TDEFLIST and TERMDEF files. The file TDEFLIST lists all valid control codes and the TERMDEF file contains their terminal-specific definitions. Using TERMDEF 2-character control codes allows different terminals, whose keys generate different character sequences, to respond identically to each recall and edit key. For more information about the TDEFLIST and TERMDEF files, see Chapter 11 of this volume.

Changes to an MPX.PRO file take effect when a user logs onto the system and the changed file is found in the default working directory or the system directory. If users modify their own MPX.PRO file, they must reinitialize their terminal key definitions by using the TSM directive $INIT PRO.

10.5.1 Assigning Predefined Functions to Keys

To assign a function to a key in the MPX.PRO file, use the syntax:

function={~*nn*|^*c*|X'*xx*'|\n}

function	is an MPX-32 terminal function, enclosed in brackets, that can be assigned to a specific key. See Table 10-3 for a list of these codes.
~*nn*	is a TERMDEF 2-character control code. The tilde (~) identifies it as a control code. See Chapter 11 of this manual for more information about TERMDEF control codes.
^*c*	is a special character preceded by the control key (^)
X'*xx*'	is a hexadecimal value. This value can be X'7F', or a value from X'00' to X'1F'. X'03' (ETX) and X'09' (TAB) are reserved by MPX-32 and cannot be assigned an MPX.PRO function.
\n	is a 1-to 3-digit octal value. This value can be \177, or a value from 0 to 37.

Multiple character sequences can only be assigned to functions by using the TERMDEF 2-character control codes.

10.5.2 Setting the Edit Mode

To set the default edit mode, modify the EDT.MODE directive in the MPX.PRO file. This directive specifies whether insert mode or modify mode is the default edit mode. Omitting this directive from MPX.PRO makes modify mode the default edit mode. The directive syntax is:

```
EDT.MODE={INSERT|MODIFY}
```

INSERT	sets insert mode as the default
MODIFY	sets modify mode as the default

10.5.3 Setting the Recall Buffer Size

To set the size of the recall buffer, modify the HIST.BUF directive in the MPX.PRO file. This directive specifies the number of command lines that the buffer maintains and it implicitly enables command line recall and edit. Omitting this directive from MPX.PRO disables the recall functions for a terminal. Any defined edit functions remain available. The directive syntax is:

HIST.BUF=*n*

n	is an integer from 0 to 23 which specifies the size of the recall buffer. A zero disables all recall and edit functions.

10.5.4 Disabling Functions

To disable a recall or edit function, do not assign it in the MPX.PRO file. Any function omitted from MPX.PRO is disabled with the exception of [return], [del], and [left]. These functions, when omitted, default to the following assignments:

function	default
[return]	X'0D'
[del]	X'7F'
[left]	X'08'

To disable all recall and edit functions, set the recall buffer size to zero with the HIST.BUF directive in MPX.PRO. Refer to the "Setting the Recall Buffer Size" section for the HIST.BUF directive syntax.

10.6 MPX.PRO Error Messages

The following error messages are associated with MPX.PRO:

```
*TERMDEF PROCESSING ERROR nn ON LINE xx
*MPX.PRO KEY DEFINITIONS NOT INITIALIZED
```

nn is an error code from the M.GETDEF system service

TERMDEF error occurred while converting a 2-character control code to a control string.

```
*MPX.PRO INITIALIZATION ERROR
*MPX.PRO KEY DEFINITIONS NOT INITIALIZED
```

MPX-32 detected an error while initializing MPX.PRO. For example, the MPX.PRO file cannot be allocated, an error occurred while reading the file, or MPX.PRO was saved instead of stored.

```
*MPX.PRO FILE SYNTAX ERROR ON LINE xx
*MPX.PRO KEY DEFINITIONS NOT INITIALIZED
```

MPX-32 detected a syntax error in the MPX.PRO file on line *xx* . Correct the syntax and reinitialize the file with the $INIT PRO directive.

```
*MPX.PRO DUPLICATE KEY DEFINITION ERROR
*MPX.PRO KEY DEFINITIONS NOT INITIALIZED
```

MPX.PRO file assigns two or more functions to the same terminal key, using MPX-32 to ignore all assignments to the key. Change the key assignments so that none duplicates and reinitialize the file with the $INIT PRO directive.

```
*MPX.PRO FILE RESERVED CONTROL CHARACTER ON LINE xx
*MPX.PRO KEY DEFINITIONS NOT INITIALIZED
```

MPX.PRO file assigns a function to a reserved control character, either X'03' (ETX) or X'09' (TAB). Change the assignment, and reinitialize the file with the $INIT PRO directive.

```
*MPX.PRO FILE DOES NOT EXIST
```

User requested $INIT PRO, but there is no MPX.PRO file in the default working directory or the system directory.

```
*COMMAND LINE RECALL/EDIT IS INVALID FOR THIS DEVICE
```

User requested $INIT PRO from the console or a terminal that is not a TSM TY device.

10.7 Sample MPX.PRO File

The sample file sets the history buffer size to 23 and the default edit mode as modify. It also assigns the MPX-32 functions to specific terminal keys by using TERMDEF control codes. All function names are enclosed in brackets.

The ASCII codes and Televideo 9220 terminal default keys are included only as comments.

```
*********************************************************************
*
*  MPX.PRO file
*
*  For every control string referenced in this file, there must be
*  corresponding entries in the Termdef files TERMDEF and TDEFLIST
*
*********************************************************************
*
** Command Line Recall and Edit Initialization values
*
HIST.BUF = 23                 # of command lines in the History buffer
EDT.MODE = MODIFY             Default mode of edit
*
** Command Line Recall Keys
*                             Defaults for Televideo 9220:
*                             ----------------------------
[return] = ~R1                * = X'0D'      Ctrl-M (return)
[up] = ~R2                    * = ESC [ A    Up arrow
[down] = ~R3                  * = ESC [ B    Down arrow
[display] = ~R4               * = X'07'      Ctrl-G
[exit recall] = ~R5           * = X'15'      Ctrl-U
*
** Command Line Edit Keys
*
[left] = ~E1                  * = ESC [ D    Left arrow (nondestructive)
[right] = ~E2                 * = ESC [ C    Right arrow
[back word] = ~E3             * = X'11'      Ctrl-Q
[next word] = ~E4             * = X'17'      Ctrl-W
[erase end line] = ~E5        * = X'04'      Ctrl-D
[delete word] = ~E6           * = X'12'      Ctrl-R
[delete char] = ~E7           * = X'7F'      Del
[begin line] = ~E8            * = X'02'      Ctrl-B
[end line] = ~E9              * = X'05'      Ctrl-E
[insert toggle] = ~EA         * = X'14'      Ctrl-T
[backspace] = ~EB             * = X'08'      Backspace (destructive)
[insert mode] = ~EC           * = X'0E'      Ctrl-N
[modify mode] = ~ED           * = X'01'      Ctrl-A
*
** Help Key
*
[help] = ~HE                  * = X'10'      Ctrl-P
```

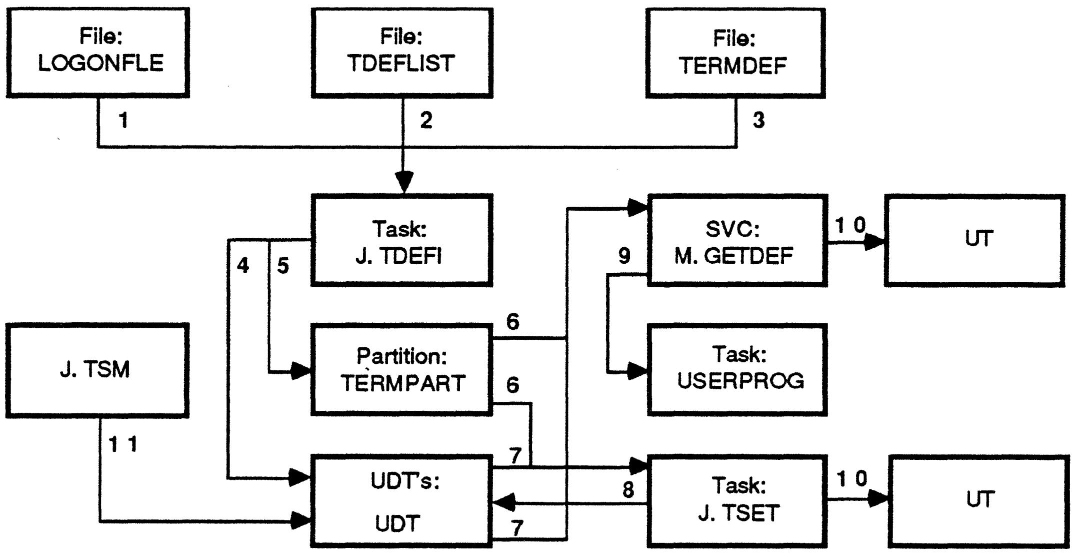
11 Terminal Definition (TERMDEF) Facility

11.1 Introduction

The Terminal Definition (TERMDEF) facility is a system service that makes the type of terminal transparent to a task. To accomplish this, TERMDEF uses the following components:

- TERMPART — the static memory partition that stores the terminal function definition and the terminal type information
- J.TDEFI — the task that initializes TERMPART with valid information and initializes the unit definition tables (UDTs) to default values. J.TDEFI uses the following files: TDEFLIST, LOGONFLE, and the terminal definition (TERMDEF) file
- M.GETDEF — the interface that accesses TERMDEF
- J.TSET — the terminal type set/reset utility that changes the current terminal type setting or resets the terminal type to its default value

Figure 11-1 shows and the following sections describe the relationship between and the functions of the TERMDEF facility components.



1. List of valid terminals
2. List of valid functions
3. Function definitions for all the terminal types
4. Terminal type (index 1 to *n* where *n* = number of unique term =); default and current
5. Subset of function definitions for valid terminals, plus the list of valid terminal types
6. Terminal types allowed, and for SVC, the definition for the requested function
7. Terminal's current and default types or current type only for SVC
8. New current terminal type (reset to default by J.TSM at logon)
9. Length and string to perform requested function or error code
10. Actual data sent to the allocated and opened terminal
11. UDT reset to default type by J.TSM during logon

R2007

Figure 11-1
TERMDEF Facility

11.2 TERMPART

TERMPART is a static SYSGEN partition that contains the terminal function, definition, and type information. TERMPART must be SYSGENed and initialized by J.TDEFI before TERMDEF can be accessed. TERMPART is included by the initialization task J.TDEFI and the terminal set task J.TSET. A program specifies TERMDEF through an SVC that uses TERMPART in an unmapped mode. (TERMPART does not increase the size of the user task.) Specify TERMPART during SYSGEN as follows:

Syntax

NAME=TERMPART,SIZE=*pages***,STRTPG=***logical***,MAP=***physical*
PROJECT=(SYSTEM,R,W) OTHERS=(R)

pages	specify the length of the partition in .5K mapblock pages. Set this value initially to 4 (one mapblock). If the initialization aborts, indicating insufficient space in TERMPART, increase the length by four pages until it no longer aborts.
logical	specifies the logical page number (in hexadecimal) to map the partition if it is included. Set *logical* to X'D0' to prevent mapping TERMPART over the CSECT of the initialization task, J.TDEFI.
physical	is the physical map block number where the partition resides. This physical location must be above the physical location of the non-split portions of the operating system and must not collide with any other defined static partitions, or the split portion of the Operating System.

11.3 J.TDEFI Program

The J.TDEFI program initializes TERMPART with information to be used with M.GETDEF. It also initializes the UDTs to their default settings. The initialization information is in the following files:

- TDEFLIST file — lists the valid 2-character control codes
- TERMDEF file — lists all the terminal definitions that the system can support
- LOGONFLE file — lists the terminal types that the system supports (TERM= in the comment area)

A subset of functions (determined by the contents of the TDEFLIST file) and a subset of terminals (determined by the contents of the LOGONFLE) are taken from TERMDEF.

The valid terminal functions are defined in the TDEFLIST file, which contains the 2-character control codes that J.TDEFI accepts. The functions a terminal supports usually is a subset of these codes, and is defined with the entry for the terminal in the TERMDEF file. J.TDEFI selects a terminal from the TERMDEF file if there is a matching TERM= directive in LOGONFLE. J.TDEFI then selects the terminal function definition if it is defined in TDEFLIST.

When J.TDEFI completes successfully, the following message displays:

```
Initialization of TERMPART complete.  TERMDEF is available.
```

The following errors can occur during initialization:

```
Include of TERMPART by J.TDEFI failed.  TERMDEF not initialized.
Open of TDEFLIST by J.TDEFI failed.  TERMDEF not initialized.
Open of LOGONFLE by J.TDEFI failed.  TERMDEF not initialized.
Open of TERMDEF by J.TDEFI failed.  TERMDEF not initialized.
Overflow of partition by J.TDEFI.  TERMDEF not initialized.
A duplicate terminal name was found.  Initialization continuing.
```

Notes:

If a function or terminal is defined in TERMDEF but not in TDEFLIST or LOGONFLE, it is omitted from TERMPART.

If a terminal is defined in LOGONFLE but not in the TERMDEF file, it is not included as a valid terminal. If the terminal is specified on a line with a terminal address, it will have a type of 0 (undefined or unknown).

If a function is defined in TDEFLIST but not referred to in TERMDEF, it is ignored.

If the contents of the TERMDEF file, the TDEFLIST file, or the TERM= portion of the LOGONFLE are changed, J.TDEFI must be run by owner SYSTEM.

11.4 TDEFLIST File

The TDEFLIST file defines the valid functions that can be processed by J.TDEFI. It is a file of 2-character control codes and comments for those control codes. Each control code must be two characters and is terminated by a blank. The characters can be upper or lower case alphanumerics. Anything else is treated as a comment by J.TDEFI. The following is an example of a TDEFLIST file:

```
*
** This is the TDEFLIST file.  It contains a list of valid 2-character
** control codes that can be used in a TERMDEF.  The format is a 2
** character, upper/lower case, alphanumeric string in the first
** two columns followed by a hyphen - and a one line description.
** The control codes are case sensitive.  Comments must be
** preceded by an asterisk * in column one.
*

** ============================================================

*
** Line and screen editing
*
al - Add line
ce - Clear to end-of-line
cd - Clear to end of display
cl - Clear screen and home cursor
dl - Delete line
nl - New line
*

** Text marking/highlighting
*
mb - Enter blinking
md - Mark do.  Puts the terminal in highlight video mode
me - Mark exit.  Puts the terminal in normal video mode
mr - Mark Reverse.  Puts the terminal in reverse video mode
ue - Underscore mode exit
us - Underscore mode start
*
```

```
**
Cursor movement functions
*
CH - Clear and Home
cm - Cursor movement.  Must also have x/y coordinates
up - Cursor up.  Moves the cursor up one line
do - Cursor down.  Moves the cursor down one line
bc - Backspace.  Moves the cursor left one space
nd - Non-destructive space.  Moves the cursor right one space
hc - Home cursor.  Moves cursor to position 0,0
ho - Home cursor.  Moves cursor to position 0,0
*

**
Cursor manipulation functions
*
HC - Hide cursor.  Turns the cursor off
SC - Show cursor.  Turns the cursor on
Ku - Sent by up arrow
Kd - Sent by down arrow
Kl - Sent by left arrow
Kr - Sent by right arrow
*

**
Command line edit key definitions
*
E1 - Left arrow
E2 - Right arrow
E3 - Back word
E4 - Next word
E5 - Erase end line
E6 - Delete word
E7 - Delete char
E8 - Begin line
E9 - End line
EA - Insert/Modify toggle
EB - Destuctive backspace
EC - Insert mode
ED - Modify mode
*

**
Command line recall key definitions
*
R1 - Return
R2 - Up
R3 - Down
R4 - Display
R5 - Exit Recall
*
```

```
**
Online Help Key definition
*
HE - Help
*

**
Miscellaneous functions
*
co - Columns.  Number of columns supported by the terminal
dw - Disable line wrap at EOL
ew - Enable line wrap at col EOL
KL - Keyboard lockout
KU - Keyboard unlock
li - Lines.  Number of lines supported by the terminal
is - Terminal initialization string
ss - Set standard screen width (80 col)
ws - Set wide screen width (132 col)
xx - Testing ..
```

11.5 TERMDEF File

The TERMDEF file provides the terminal type and functions supported to J.TDEFI.

Syntax

#
# *comment*
#
term1 | *term2* | *term3* | *term4*:*cc*=*cs*:*cc*#*cs*:*cc*=*tdef*:\
cc=*cs*:*cc*=*cs*:*cc*:*cc*#*cs*:

term1	is an alphanumeric character that uniquely identifies the terminal type. This parameter can be omitted.
term2	is a 2-character alphanumeric string that uniquely identifies the terminal type. This parameter can be omitted.
term3	is a 3-character alphanumeric string that uniquely identifies the terminal type. This parameter can be omitted.
term4	is a 4- to 8-character alphanumeric string that uniquely identifies the terminal type. This parameter can be omitted.

Note: The parameters listed above cannot have embedded blanks and at least one of the four parameters must be specified. Only one of each type of parameter can be specified; otherwise the first parameter of a given length is used and all further parameters of that length are ignored.

cc	specifies a 2-character control code. Control codes represent the functions a terminal can perform. *cc* is a 2-character, alphanumeric argument (upper/lower case is significant).
cs	specifies the control string. The control string represents the data that defines the terminal function. See the Control Strings section below for more information.
|	separates one terminal type from another
:	separates terminal function definition pairs (a control code and control string). Each pair starts and ends with a colon. A colon also marks the end of the last terminal type parameter. A control code without a control string is a boolean.
=	separates the control code from its control string
\	at the end of a line, indicates the next line is a continuation of the terminal definition entry
#	separates the control code from the control string, and indicates the control string is an absolute numeric value. This value is returned as ASCII-coded decimal. For example, `pp#229` returns a three-byte string of 229.

Notes:

The TERMDEF file must be stored uncompressed and can be numbered or unnumbered.

11.5.1 Booleans

Booleans indicate the presence or absence of a supported function. A boolean is declared by supplying only the control code. For example, `:bs:` represents destructive backspace.

A TERMDEF file, in this case, without the `:bs:` returns a function not supported error, while a TERMDEF file with `:bs:` returns a single byte 1.

11.5.2 Control Strings

Control strings are the commands that make a terminal perform a function. These functions are defined in the TDEFLIST file. The following characters are used to build a control string.

^	indicates the next character is a control character
\E	starts an escape (X'1B') sequence
\n	is treated as an ASCII newline (X'0A')
\r	is treated as an ASCII carriage return (X'0D')
\t	is treated as an ASCII tab (X'09')
\b	is treated as an ASCII backspace (X'08')
\f	is treated as an ASCII formfeed (X'0C')
\nnn	represents an octal byte value where *n* is from 0 to 8
\\:	is converted to a literal :
\\\\	is converted to a literal \\
%	indicates cursor addressing processing is required (see the Cursor Addressing section for details)

The following are examples of terminal function definitions.

`:cl=^Z:`	Televideo 910 clear screen
`:cl=\E\134:`	Hazeltine 1500 (Encore version) clear screen
`:hc=\EH:`	VT52 home cursor
`:wo=\E[?71:`	TVI9220 auto-wrap out
`:bs:`	Terminal has destructive backspace

11.5.3 Cursor Addressing

Cursor addressing customizes cursor movement for the terminal. The cursor movement parameters are:

%d	indicates the value supplied is a 1- to 8-digit decimal coordinate
%2	indicates the value supplied is a 2-digit decimal coordinate
%3	indicates the value supplied is a 3-digit decimal coordinate
%.	indicates the value supplied is treated as a byte literal
%+	adds the supplied value to the next character (used to bias a value)
%i	increments the supplied values by one (used to one-origin the coordinate pairs)
%r	reverses the supplied values (used for terminals that use column/line as opposed to line/column)
%%	is converted to a literal %

The supplied values are the optional x and y coordinate pair parameter that can be specified when using TERMDEF. See the Accessing TERMDEF with M.GETDEF section for more information.

The following is an example of a terminal function definition for cursor movement.

```
    :cm=\E[%i%3,%3f^@: TVI9220 cursor movement
```

`:`	is the starting colon
`cm`	is the control code for cursor movement
`=`	separates control code from the control string
`\E`	indicates escape
`[`	is the lead-in character for the TVI9220
`%`	indicates "add one to x and y values" (1 origin)
`i%3`	is the 3-digit decimal x coordinate (row)
`,`	is the delimiter used by the TVI9220
`%3f`	is the 3-digit decimal y coordinate (column)
`^`	is the termination character for TVI9220
`@`	is a null
`:`	is the termination colon

The string shown above and the supplied values of 10 and 15, produce the following hexadecimal string:

```
    1B 5B 30 31 31 3B 30 31 36 66 00
```

11.5.4 Sample TERMDEF File

This example is just a sample and may not match the system file exactly.

```
#######################################################
#
# TERMDEF file 1.00 created 09/05/86
#
# Max for any single terminal function definition is 256 bytes.
# Max for any single TERMDEF entry is 1024 bytes.
#
#
#######################################################
#
## Standard TeleVideo 910 terminal (ASCII) except the intensities
## are reversed on marks.  To restore intensities to normal,
## change :md to :md=\E: and change :me to :me=\E(:
#
#
T|TVI910|910|TeleVideo 910 with low intensity as normal mode:\
   :cm=\E=%+ %+ : cd=\EY:CH=\E*:mb=\G2:\
   :md=\E(:me=\E): nl=^ :KU=\E":KL=\E#:\
   :is=\E)\E*:ce=\ET:cl=\E+:ho=\036:\
   :hc=\036:HC=\E.:SC=\E.:\
   :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
## Standard TeleVideo 921 terminal (ASCII)
#
P|TVI921|921|Televideo 921:\
   :cm=\E=%+ %+ :\
   :al=\EE:dl=\ER:\
   :ce=\ET:\
   :md=\E):me=\E(:cl=\E;:hc=\036:\
   :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
## Standard TeleVideo 905 terminal (ASCII)
#
TVI905|905|Televideo 905:\
   :cm=\E=%+ %+ :\
   :al=\EE:dl=\ER:\
   :ce=\ET:\
   :md=\E):me=\E(:cl=\E;:hc=\036:\
   :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
```

```
## Standard TeleVideo 950 terminal (ASCII)
#
I|TVI950|950|Televideo 950 (Not tested):\
   :cm=\E=%+ %+ :\
   :ce=\ET:is=\EY:dl=\ER:al=\EE:\
   :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
## Standard TeleVideo 922 terminal (ANSI/VT52/VT100/VT220)
## in native mode (922 mode).  Note that cl combines
## clear screen and home cursor since clear screen
## doesn't home cursor.
## Also this version highlights to low intensity.
## and is normally in high intensity
#
TVI922|922|Televideo 922:\
   :cm=\E[%i%3;%3f^@:\
   :ce=\E[K:dl=\E[M:al=\E[L:\
   :me=\E[?27h:md=\E[?271:\
   :is=\E[2J\E[H:cl=\E[2J\E[H:hc=\E[H:SC=\E[?25h:HC=\E[?251:\
   :ss=\E[?31:ws=\E[?3h:\
   :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
## Standard TeleVideo 9220 terminal (922 replacement) in
## native mode (9220 mode, much like a VT220) Note: cl
## combines clear screen and home cursor, clear screen
## doesn't home cursor.  Unlike the TVI922 the TVI9220 runs
## best in low intensity and highlights to high intensity
#
A|AN|ANS|TVI9220|Televideo 9220 (9220 MODE) ANSI Terminal:\
   :cm=\E[%i%3;%3f^@:\
   :ce=\E[K:dl=E[M:al=\E[L:\
   :md=\E[?27h:me=\E[?271:\
   :is=\E[2J\E[H:cl=\E[2J\E[H:hc=\E[H:SC=\E[?25h:HC=\E[?251:\
   :ss=\E[?31:ws=\E[?3h:\
   :E1=\E[D:E2=\E[C:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=\E[A:R3=\E[B:R4=^G:R5=^U:HE=^P:
#
```

```
## Encore modification Hazeltine 1500
#
Z|HAZ|Hazeltine 1500 (Encore mod - lead-in is ESCAPE, not ~):\
   :cm=\E^Q%r%.%+ :\
   :me=\E^Y:md=\E\037:hc=\E^R:\
   :ce=\E^O:dl=\E^S:al=\E^Z:cl=\E\034:\
   :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
## Standard VT100 terminal
#
V|VT|VT100|100|DEC VT100 (Not tested):\
   :cm=\E[%i%3;%3H^@^@^@:\
   :ku=\EOA:kd=\EOB:kl=\EOD:kr=\EOC:\
   :up=\E[A:do=\E[B:bc=\E[D:nd=\E[C:\
   :ce=\E[K:cl=\E[H\E[25:\
   :me=\E[m:md=\E[1m:\
   :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
## Commodore Amiga micro computer in VT100 Emulation mode
## (a subset of full VT100 emulation)
#
M|AM|AMI|AMIGA|Amiga in vt100 emulation mode:\
   :al=\E[L:dl=\E[M:\
   :cm=\E[%i%3;%3H^@^@^@:\
   :ku=\E[A:kd=\E[B:kl=\E[D:kr=\E[C:\
   :up=\E[A:do=\E[B:bc=\E[D:nd=\E[C:\
   :ce=\E[K:cl=\E[2J:\
   :md=\E[0;32;40m:me=\E[0;31;40m:\
   :is=\E[0;31;40m\E[2J:li#23:co#80:\
   :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
## Standard VT52 terminal
#
5|VT52|DEC VT52 (Not tested):\
   :cm=\EY%+ %+ :\
   :up=\EA:do=^J:nd=\EC:ce=\EK:\
   :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
```

```
## Atari Corp 520 or 1040 ST micro computer in VT52 emulation
## mode (actually a superset of VT52 w/al,dl).
## Note: This version changes the text color rather than
## inverting the video to highlight text.
## It applies to BIOS text, not GEM text.
#
S|ST|AST|ATARIST|ATARI ST (Atari 520ST & 1040ST VT52 Emulation):\
    :cm=\EY%+ %+ :\
    :al=\EL:dl=\EM:me=\Eb3:md=\Eb2:bc=\ED:hc=\EH:co#80:li#24\
    :up=\EA:do=\EB:nd=\EC:ce=\EK:cl=\EE:SC=\Ee:HC=\Ef:\
    :ew=\Ev:dw=\Ew:is=\Ew\Eb3\EE:\
    :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
    :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
    :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
## Standard Tektronix 4105 terminal
#
4|4105|4105|Tektronix 4105 (Not tested):\
    :cm=E[%i%3;%3f^:\
    :ce=E[K:\
    :is=\E%!0\ELZ\EKDH11\072\EKDH218\EKDH31<\EKDH01;\E\
    KDH)18\EDH'1<\E:\
    :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
    :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
    :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
## Power Series
#
PS|PSXXX0|164|Power Series (Not tested):\
    :cm=\EC%r%+\001%+\001^@^@^@^@^@^@^@^@^@^@:\
    :ce=\EEL:\
    :is=^L^@^@^@^@^@^@^@^@^@:\
    :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
    :E7=\177:E8=^B:E9=^E:EA=^T:\
    :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
```

```
## TeleVideo 910 terminal in Hazeltine emulation mode
#
H|H|1410|910 in 1410 emulation (Not tested):\
   :cm=\E^Q%r%.%+ :\
   :ce=\E^O:al=\E^Z:\
   :E1=^H:E2=^L:E3=^Q:E4=^W:E5=^D:E6=^R:\
   :E7=\177:E8=^B:E9=^E:EA=^T:EB=^X:EC=^N:ED=^A:\
   :R1=^M:R2=^K:R3=^J:R4=^G:R5=^U:HE=^P:
#
##############################################################
#
# As you add terminals, make sure the control codes are
# in the TDEFLIST file and the type is added to LOGONFLE
# if applicable.
#
##############################################################
#
```

11.6 LOGONFLE File

LOGONFLE contains the list of valid terminals and the system console. The type of terminal is specified by the TERM= directive in LOGONFLE. J.TDEFI uses this information to set the terminal's current and default types in the UDT. The current setting can be changed by the J.TSET task. The current type is reset to the default type automatically when logging on to the terminal.

Terminals listed in the TERM= directive provide a subset of terminals for the task J.TDEFI to select from the TERMDEF file.

The console named in the CONS= directive identifies the type of console being used. It has the same format and restrictions as TERM= except only the first one encountered is accepted.

For more information about LOGONFLE, refer to the System Administrator Services chapter in Volume III of the MPX-32 Reference Manual.

Syntax

7EA0 9600 FULL 8 S1 WXON ! Comments and **TERM=***term* CONS=*term*

or

* **LOGONFLE** *comment line* and **TERM=***term* **CONS=***term*

term	is a unique 1- to 8-character alphanumeric terminal name that is also uniquely defined in the TERMDEF file.

With the first method for specifying syntax, *term* is placed in the UDT of terminal 7EA0. The second method for specifying syntax allows terminals to be defined that are not connected to the system; for example, dial-up terminals or a network connection.

Notes:

The directives appear in the comment area of LOGONFLE. This allows J.TINIT to continue processing the LOGONFLE without modification.

The directives for the terminals and the console must appear in a comment field.

11.7 Accessing TERMDEF with M.GETDEF

To access TERMDEF, issue an SVC (2,X'7A' or M.GETDEF). For more information about M.GETDEF, refer to the MPX-32 Reference Manual Volume I, Chapter 6. A TERMDEF information block must be provided that contains the following information:

- the LFC the user is accessing the terminal from
- a 2-letter control code that identifies the control string requested
- the address of a buffer in the task where the SVC will place the control string (the 24-bit address must not use the F or C bits)
- the location where the length of the buffer is specified and the length in bytes of the string M.GETDEF returns in the buffer

The format of a TERMDEF information block is:

	0 7 8 15	16 23 24 31
Word 0	Terminal LFC (TDF.ULFC)	
1	User buffer address (TDF.UADR)	
2	Control code (TDF.UFUN)	Reserved
3	User buffer length (TDF.ULEN)	Return String Length (TDF.USTR)
4	Optional X coordinate (0 org, row) TDF.UXCO	
5	Optional Y coordinate (0 org, column) TDF.UYCO	

Notes:

The TERMDEF information block must be on a word boundary.

The TDF symbolic names are not in the macro library.

The strings returned by M.GETDEF can be used as output to the terminal. For unformatted I/O to the terminal or no-wait I/O (non-TSM I/O), these strings require no additional processing.

For formatted TSM I/O to work, the string must be preceded by a plus sign (+) and, if applicable, followed immediately by a carriage return and line feed. The plus sign (+) in byte one prevents H.TSM from counting the line (and generating `ENTER CR FOR MORE` prompts), and inhibits carriage return/linefeed.

For cursor addressing, TERMDEF makes the coordinates transparent so (no matter how the terminal uses them) the coordinates are always zero origin. The first coordinate is rows from the top of the screen to the bottom and the second is columns from the left side of the screen to the right. Therefore, 0,0 is always the upper lefthand corner of the screen. For standard terminals with 80 columns and 24 rows, references are 0 to 79 and 0 to 23.

11.7.1 Errors

If M.GETDEF generates an error the following occurs:

- CC1 is set.
- The string length is returned as zero.
- The binary error number is returned in the requested function halfword (TDF.UFUN).

Following are the M.GETDEF errors:

Error 1: The supplied LFC is invalid.

M.GETDEF could not locate the LFC provided in the task's FPTs. Either the wrong LFC was specified or the unit for that LFC has not yet been opened.

Error 2: Unknown terminal type.

The value in UDT.CTDF is 0 (unknown or undefined) or out of range. If the number is out of range, the LOGONFLE was changed and the TERM= directive was omitted for the terminal. If UDT.CTDF is zero J.TDEFI ran and the TERM= directive in the LOGONFLE has no corresponding entry in the TERMDEF file.

Error 3: User buffer is too large.

The buffer supplied is larger than 2K words. M.GETDEF does not accept more than one map block crossing.

Error 4: Cannot include partition.

The M.GETDEF service call could not access TERMPART. Either the partition is not SYSGENed in the current system image (C.TDEFA is 0 in this case), or the task J.TDEFI is currently being run and the partition is currently gated as the data is being changed. M.GETDEF does not include TERMPART but accesses it in an unmapped mode with internal gating.

Error 5: Undefined control code requested.

The supplied control code is not defined in the file TDEFLIST. Therefore, it is not in TERMPART.

Error 6: User buffer is too small.

The control string the M.GETDEF service call is trying to return is larger than the buffer provided.

Error 7:	Partition data integrity suspect. TERMPART is present but contains suspect or no data. If data is not present, the task J.TDEFI either was not run or did not complete. Check the console for messages from J.TDEFI. J.TDEFI can be run at any time by SYSTEM. If the data is corrupt, there may be a conflict with the partition name TERMPART; another task may be including it and altering its contents.
Error 8:	Invalid terminal type supplied. The current type of the terminal (UDT.CTDF) is not defined in TERMPART. This error indicates the data in TERMPART is corrupted.
Error 9:	Invalid user buffer address. The address supplied in the TERMDEF information block was invalid. The address expected is a 24-bit address. If M.GETDEF is accessed with a 19-bit address and the task is not running extended, the F-bit is masked and the address is treated as a 24-bit address by the service.
Error 10:	Control code selected is undefined for this terminal. The 2-letter control code is defined in the TDEFLIST file but is not valid for this terminal because it is not defined in the TERMDEF file entry for this terminal. For example, al (add line) is not valid for a VT52 terminal because a VT52 cannot add a line.
Error 11:	TERMDEF is not installed. TERMDEF is not installed because the SYSGEN NOTDEF directive was specified.
Other:	TERMDEF information block address is invalid. This error is indicated by CC1 being set but the contents of the requested control code field and the string length field are not changed. The service was unable to use the address supplied for the TERMDEF information block. This is usually caused by the TERMDEF information block not being word bounded.

11.8 Terminal Type Set/Reset Utility (J.TSET)

J.TSET changes the current type setting for the terminal or resets the type to its default. The type information for a specific terminal is kept in the UDT. The acceptable types are those specified in the file LOGONFLE with the TERM= directive that is processed by the initialization program J.TDEFI.

J.TSET takes the ASCII type supplied on the command line and scans TERMPART for a match. If a match is found, J.TSET stores the index for that entry in the UDT as the current type. If nothing is supplied on the command line, J.TSET stores the default setting in the current setting. This task can only be run interactively on a terminal.

Syntax

J.TSET [*type* | **?** | **HELP** | **SHOW** | **OFF**]

type	is a 1- to 8-character symbolic name for the terminal type as defined by the LOGONFLE and/or the TERMDEF file (for example, TVI910)
?	same as HELP
HELP	produces a brief description of how to run J.TSET
SHOW	lists the valid terminal types that J.TSET accepts. The current type for this terminal is indicated by a < in the first column. The default for this terminal is indicated by a > in the second column.
OFF	sets the current terminal type to 0 (undefined), which disables the TERMDEF facility for this terminal

J.TSET uses the default LFC UT for error and help messages. UT cannot be reassigned. The output file contains the error messages.

Help Message

When HELP or ? is specified on the command line, J.TSET displays the following:

```
usage: J.TSET [type|?|HELP | SHOW | OFF]
```

type	is 1- to 8-character ASCII string defined in the TERMDEF file. If nothing is specified, the task sets the terminal type to the default value.

Completion Messages

Upon successful completion, J.TSET displays one of the following messages:

```
Terminal type has been set as you requested.
Terminal type has been reset to default type.
```

Error Messages

The possible error messages from J.TSET are (the abort flag is not set for these conditions):

```
error: Unable to include the partition TERMPART.  M.INCLUDE: (errnum).
error: Invalid or undefined terminal type was supplied.
error: Integrity of the data in the partition TERMPART is suspect.
error: TERMDEF is not installed.
```

Aborts

The possible aborts with J.TSET are:

```
TD01 - ATTEMPTED TO RUN J.TSET IN BATCH MODE
TD02 - J.TSET WAS UNABLE TO OPEN UT FOR PROCESSING
```

A MPX-32 Device Access

A.1 Description

Throughout the MPX-32 Reference Manual, the generic descriptor *devmnc* indicates that a device can be specified.

Under MPX-32, device addresses are specified using a combination of three levels of identification. They are device type, device channel/controller address, and device address/subaddress.

A device can be specified using the generic device type mnemonic only, which results in allocation of the first available device of the type requested. Device type mnemonics are listed in Table A-1.

A second method of device specification is achieved by using the generic device type mnemonic and specifying the channel/controller address. This results in allocation of the first available device of the type requested on the specified channel or controller.

The third method of device selection requires specification of the device type mnemonic, channel/controller, and device address/subaddress. This method allows specification of a particular device.

Table A-1
Device Type Mnemonics and Codes

Device Type Code	Device Type Mnemonic	Device Description
00	CT	Operator console (not assignable)
01	DC	Any disk unit except memory disk
02	DM	Any moving head or memory disk
03	DF	Any fixed head disk
04	MT	Any magnetic tape unit
05	M9	Any 9-track magnetic tape unit*
06	M7	Any 7-track magnetic tape unit*
08	CR	Any card reader
0A	LP	Any line printer
0B	PT	Any paper tape reader-punch
0C	TY	Any teletypewriter (other than console)
0D	CT	Operator console (assignable)
0E	FL	Floppy disk
0F	NU	Null device
10	CA	Communications adapter (binary synchronous/asynchronous)
11	U0	Available for user-defined applications
12	U1	Available for user-defined applications
13	U2	Available for user-defined applications
14	U3	Available for user-defined applications
15	U4	Available for user-defined applications
16	U5	Available for user-defined applications
17	U6	Available for user-defined applications
18	U7	Available for user-defined applications
19	U8	Available for user-defined applications
1A	U9	Available for user-defined applications
1B	LF	Line printer/floppy controller (used only with SYSGEN)
N/A	ANY	Any nonfloppy disk except memory disk

* When both 7- and 9-track magnetic tape units are configured, the designation must be 7-track.

A.2 Special Device Specifications and Handling

A.2.1 Magnetic Tape/Floppy Disk

For magnetic tape and floppy disks, unblocking, density, a reel identifier, and multivolume number (magnetic tape only) can be included in the device specification.

Syntax

$ASSIGN *lfc* **TO DEV=***devmnc* **[BLOCKED={ Y | N }]**
[DENSITY={N | P | G | 800 | 1600 | 6250 }] [ID=*id***] [MULTIVOL=***number***]**

lfc is a 1- to 3-character logical file code

DEV=*devmnc*
devmnc is the device specification of a configured peripheral device (see the Description section)

[BLOCKED={ Y | N }]
if Y is specified, medium is blocked. If N is specified, medium is not blocked. If not specified the default is blocked.

[DENSITY={N | P | G | 800 | 1600 | 6250 }]
specifies density of high speed XIO tape. If not specified, the default is 6250 bpi. Values are as follows:

Value	Description
N or 800	indicates 800 bpi nonreturn to zero inverted (NRZI).
P or 1600	indicates 1600 bpi phase encoded (PE).
G or 6250	indicates 6250 bpi group coded recording (GCR). This is the default.

[ID=*id***]** *id* specifies a 1- to 4-character identifier for the reel. If not specified, the default is SCRA (scratch).

[MULTIVOL=*number***]**
number is a volume number. If multivolume tape, *number* must be specified. If not specified, the default is not multivolume (0). This option is not valid for use with floppy disks.

When the task that has an assignment to tape is activated, a mount message indicates the name of the task and other information on the system console:

```
MOUNT reel VOL volume ON devmnc
TASK taskname,taskno REPLY R,H,A, OR DEVICE:
jobno
```

reel specifies a 1- to 4-character identifier for the reel. If not specified, the default is SCRA (Scratch).

volume	identifies the volume number to mount if multivolume tape
devmnc	is the device mnemonic for the tape unit selected in response to the assignment. If a specific channel and subaddress are supplied in the assignment, the specific tape drive is selected and named in the message; otherwise, a unit is selected by the system and its complete address is named in the message.
jobno	identifies the job by job number if the task is part of a batch job
taskname	is the name of the task to which the tape is assigned
taskno	is the task number assigned to the task by the system

`R,H,A, OR DEVICE`

the device listed in the message can be allocated and the task resumed (`R`), a different device can be selected (`DEVICE`), the task can be aborted (`A`), or the task can be held with the specified device deallocated (`H`). If an R response is given and a high speed XIO tape drive is being used, its density can be changed when the software select feature is enabled on the tape unit front panel. If specified, it overrides any specification made at assignment. Example usage: `RN`, `R1600`, etc.

Note: Do not insert blanks or commas.

Response:

To indicate the drive specified in the mount message is ready and proceed with the task, mount the tape on the drive and type `R` (resume), optionally followed by a density specification if the drive is a high speed XIO tape unit. To abort the task, type `A` (abort). To hold the task and deallocate the specified device, type `H` (hold). The task can be resumed by the OPCOM CONTINUE directive; at which time, a tape drive is selected by the system and the mount message redisplayed.

To select a tape drive other than the drive specified in the message, enter the mnemonic of the drive to be used. Any of the three levels of device identification can be used. The mount message is reissued. Mount the tape and type `R` if satisfactory, or if not satisfactory, abort, override, or hold as described.

Examples of the three methods of device specification follow:

Type 1 - Generic Device Class

```
    $ASSIGN OUT TO DEV=M9 MUL=1 ID=MVOL
```

In this example, the device assigned to logical file code (LFC) `OUT` is any `9`-track tape unit on any channel. The multivolume reel number is `1`. The reel identifier is `MVOL` and the tape is blocked.

Type 2 - Generic Device Class and Channel/Controller

```
    $ASSIGN OUT TO DEV=M910 ID=MVOL BLO=N
```

In this example, the device assigned to logical file code (LFC) `OUT` is the first available 9-track tape unit on channel `10`. The specification is invalid if a 9-track tape unit does not exist on the channel. The reel identifier is `MVOL`. This is not a multivolume tape and is unblocked.

Type 3 - Specific Device Request

```
    $ASSIGN OUT TO DEV=M91001
```

In this example, the device assigned to logical file code (LFC) `OUT` is the 9-track tape unit `01` on channel `10`. The specification is invalid if unit 01 on channel 10 is not a 9-track tape. The tape reel identifier is SCRA. The tape is blocked and is not multivolume.

A.2.2 Temporary Disk Space

For a temporary disk file the following can be specified: size, blocking, printing or punching, and access.

Syntax

$ASSIGN *lfc* **TO TEM**P[=(*volname*)] [**ACC**ESS=([**R**EAD] [**W**RITE] [**M**ODIFY] [**U**PDATE] [**A**PPEND])]
[**BLO**CKED={ **Y** | **N** }] [**PRI**NT | **PUN**CH] [**SI**ZE=*blocks*]

lfc is a 1- to 3-character logical file code

TEMP[=(*volname*)]
: *volname* is the 1- to 16-character volume name where temporary space is allocated. If not specified, the default is the current working volume or any public volume.

[**ACC**ESS=([**R**EAD] [**W**RITE] [**M**ODIFY] [**U**PDATE] [**A**PPEND])]
: specifies the types of access for the file. If not specified, the default is the access specified at file creation.

[**BLO**CKED={ **Y** | **N** }]
: if Y is specified, the file is blocked. If N is specified, the file is unblocked. If not specified, the default is blocked.

[**PRI**NT | **PUN**CH]
: indicates the file is to be printed (PRINT) or punched (PUNCH) after deassignment

[**SI**ZE=*blocks*]
: *blocks* is the number of 192-word blocks required. If not specified, the default is 16 blocks.

Examples

In the following example, the device assigned to logical file code (LFC) `OUT` is the current working volume or any public volume and the file prints to the SLO device after deassignment.

```
AS OUT TO TEM PRI
```

The following example designates the system volume as the device for the temporary blocked file.

```
AS OUT TO TEMP=(SYSTEM) BLO=Y
```

A.3 GPMC Devices

GPMC/GPDC device specifications follow the general structure just described. The terminal at subaddress `04` on GPMC 01 whose channel address is `20` would be identified as follows:

```
$AS DEV TO DEV=TY2004
```

A.4 Null Device

A special device type, NU, is available for null device specifications. Files accessed using this device type generate an end-of-file (EOF) when a read is attempted and normal completion when a write is attempted.

A.5 System Console

Logical file codes are assigned to the system console by using the device type CT.

A.6 Special System Files

There are four special mnemonics provided for access to special system files: SLO, SBO, SGO and SYC. These are assigned with the $ASSIGN statement, as in:

```
$ASSIGN OUT TO SLO
```

For nonbatch tasks, SLO and SBO files are allocated dynamically by the system and used to disk buffer output to a device selected automatically. For batch tasks, use of SLO and SBO files is identical, except that automatic selection of a device can be overridden by assigning a specific file or device.

A.7 Samples

A description of device selection possibilities is constructed as follows:

Disk

DC	Any disk except memory disk
DM	Any moving head or memory disk
DM08	Any moving head disk on channel 08
DM0801	Moving head disk 01 on channel 08
DM0002	Memory disk 02 on channel 00
DF	Any fixed head disk
DF04	Any fixed head disk on channel 04
DF0401	Fixed head disk 01 on channel 04

Tape

MT	Any magnetic tape
M9	Any 9-track magnetic tape
M910	Any 9-track magnetic tape on channel 10
M91002	9-track magnetic tape 02 on channel 10

Card Equipment

CR	Any card reader
CR78	Any card reader on channel 78
CR7800	Card reader 00 on channel 78

Line Printer

LP	Any line printer
LP7A	Any line printer on channel 7A
LP7A00	Line printer 00 on channel 7A
LP7EA0	Serial printer A0 on ACM channel 7E

B System Services Cross-Reference

B.1 Macro Name Listing

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.ACTV	Activate Task	1,X'52'	H.REXS,15	6.2
M_ACTV	Activate Task	1,X'52'	H.REXS,15	7.2
M.ADRS	Memory Address Inquiry	1,X'44'	H.REXS,3	6.2
M_ADRS	Memory Address Inquiry	1,X'44'	H.REXS,3	7.2
M_ADVANCE	Advance Record	1,X'33'	H.IOCS,7	7.2
	Advance File	1,X'34'	H.IOCS,8	7.2
M.ALOC	Allocate File or Peripheral Device	1,X'40'	H.MONS,21	6.4
M.ANYW	Wait for Any No-wait Operation Complete, Message Interrupt, or Break Interrupt	1,X'7C'	H.REXS,37	6.2
M_ANYWAIT	Wait for Any No-wait Operation Complete, Message Interrupt, or Break Interrupt	1,X'7C'	H.REXS,37	7.2
M_ASSIGN	Assign and Allocate Resource	2,X'52'	H.REXS,21	7.2
M.ASSN	Assign and Allocate Resource	2,X'52'	H.REXS,21	6.2
M.ASYNCH	Set Asynchronous Task Interrupt	1,X'1C'	H.REXS,68	6.2
M_ASYNCH	Set Asynchronous Task Interrupt	1,X'1C'	H.REXS,68	7.2
M_AWAITACTION	End Action Wait	1,X'1D'	H.EXEC,40	7.2
M.BACK	Backspace Record	1,X'35'	H.IOCS,9	6.2
	Backspace File	1,X'36'	H.IOCS,19	6.2
M_BACKSPACE	Backspace Record	1,X'35'	H.IOCS,9	7.2
	Backspace File	1,X'36'	H.IOCS,19	7.2
M.BATCH	Batch Job Entry	2,X'55'	H.REXS,27	6.2
M_BATCH	Batch Job Entry	2,X'55'	H.REXS,27	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.BBTIM	Acquire Current Date/Time in Byte Binary Format	2,X'50'	H.REXS,74	6.2
M_BBTIM	Acquire Current Date/Time in Byte Binary Format	2,X'50'	H.REXS,74	7.2
M.BORT	Abort Specified Task	1,X'56'	H.REXS,19	6.2
	Abort Self	1,X'57'	H.REXS,20	6.2
	Abort With Extended Message	1,X'62'	H.REXS,28	6.2
M_BORT	Abort Specified Task	1,X'56'	H.REXS,19	7.2
	Abort Self	1,X'57'	H.REXS,20	7.2
	Abort With Extended Message	1,X'62'	H.REXS,28	7.2
M.BRK	Break/Task Interrupt Link/Unlink	1,X'6E'	H.REXS,46	6.2
M_BRK	Break/Task Interrupt Link/Unlink	1,X'6E'	H.REXS,46	7.2
M.BRKXIT	Exit From Task Interrupt Level	1,X'70'	H.REXS,48	6.2
M_BRKXIT	Exit From Task Interrupt Level	N/A	N/A	7.2
M.BTIM	Acquire Current Date/Time in Binary Format	2,X'50'	H.REXS,74	6.2
M_BTIM	Acquire Current Date/Time in Binary Format	2,X'50'	H.REXS,74	7.2
M.CDJS	Submit Job from Disc File	1,X'61'	H.MONS,27	6.4
M_CHANPROGFCB	Execute Channel Program File Control Block	N/A	N/A	7.2
M.CLOSER	Close Resource	2,X'43'	H.REMM,22	6.2
M_CLOSER	Close Resource	2,X'43'	H.REMM,22	7.2
M.CLSE	Close File	1,X'39'	H.IOCS,23	6.2
M_CLSE	Close File	1,X'39'	H.IOCS,23	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.CMD	Get Command Line	2,X'61'	H.REXS,88	6.2
M_CMD	Get Command Line	2,X'61'	H.REXS,88	7.2
M.CONABB	Convert ASCII Date/Time to Byte Binary Format	2,X'51'	H.REXS,75	6.2
M_CONABB	Convert ASCII Date/Time to Byte Binary Format	2,X'51'	H.REXS,75	7.2
M.CONADB	Convert ASCII Decimal to Binary	1,X'28'	H.TSM,7	6.2
M_CONADB	Convert ASCII Decimal to Binary	1,X'28'	H.TSM,7	7.2
M.CONAHB	Convert ASCII Hex to Binary	1,X'29'	H.TSM,8	6.2
M_CONAHB	Convert ASCII Hex to Binary	1,X'29'	H.TSM,8	7.2
M.CONASB	Convert ASCII Date/Time to Standard Binary	2,X'51'	H.REXS,75	6.2
M_CONASB	Convert ASCII Date/Time to Standard Binary	2,X'51'	H.REXS,75	7.2
M.CONBAD	Convert Binary to ASCII Decimal	1,X'2A'	H.TSM,9	6.2
M_CONBAD	Convert Binary to ASCII Decimal	1,X'2A'	H.TSM,9	7.2
M.CONBAF	Convert Binary Date/Time to ASCII Format	2,X'51'	H.REXS,75	6.2
M_CONBAF	Convert Binary Date/Time to ASCII Format	2,X'51'	H.REXS,75	7.2
M.CONBAH	Convert Binary to ASCII Hex	1,X'2B'	H.TSM,10	6.2
M_CONBAH	Convert Binary to ASCII Hex	1,X'2B'	H.TSM,10	7.2
M.CONBBA	Convert Byte Binary Date/Time to ASCII	2,X'51'	H.REXS,75	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_CONBBA	Convert Byte Binary Date/Time to ASCII	2,X'51'	H.REXS,75	7.2
M.CONBBY	Convert Binary Date/Time to Byte Binary	2,X'51'	H.REXS,75	6.2
M_CONBBY	Convert Binary Date/Time to Byte Binary	2,X'51'	H.REXS,75	7.2
M.CONBYB	Convert Byte Binary Date/Time to Binary	2,X'51'	H.REXS,75	6.2
M_CONBYB	Convert Byte Binary Date/Time to Binary	2,X'51'	H.REXS,75	7.2
M.CONN	Connect Task to Interrupt	1,X'4B'	H.REXS,10	6.2
M_CONN	Connect Task to Interrupt	1,X'4B'	H.REXS,10	7.2
M_CONSTRUCTPATH	Reconstruct Pathname	2,X'2F'	H.VOMM,16	7.2
M_CONVERTTIME	Convert Time	2,X'51'	H.REXS,75	7.2
M.CPERM	Create Permanent File	2,X'20'	H.VOMM,1	6.2
M.CREATE	Create Permanent File	1,X'75'	H.FISE,12	6.4
M_CREATEFCB	Create File Control Block	N/A	N/A	7.2
M_CREATEP	Create Permanent File	2,X'20'	H.VOMM,1	7.2
M_CREATET	Create Temporary File	2,X'21'	H.VOMM,2	7.2
M.CTIM	Convert System Date/Time Format	2,X'51'	H.REXS,75	6.2
M_CTIM	Convert System Date/Time Format	2,X'51'	H.REXS,75	7.2
M.CWAT	System Console Wait	1,X'3D'	H.IOCS,26	6.2
M_CWAT	System Console Wait	1,X'3D'	H.IOCS,26	7.2
M.DALC	Deallocate File or Peripheral Device	1,X'41'	H.MONS,22	6.4
M.DASN	Deassign and Deallocate Resource	2,X'53'	H.REXS,22	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.DATE	Date and Time Inquiry	1,X'15'	H.REXS,70	6.2
M_DATE	Date and Time Inquiry	1,X'15'	H.REXS,70	7.2
M_DEASSIGN	Deassign and Deallocate Resource	2,X'53'	H.REXS,22	7.2
M.DEBUG	Load and Execute Interactive Debugger	1,X'63'	H.REXS,29	6.2
M_DEBUG	Load and Execute Interactive Debugger	1,X'63'	H.REXS,29	7.2
M.DEFT	Change Defaults	2,X'27'	H.VOMM,8	6.2
M_DEFT	Change Defaults	2,X'27'	H.VOMM,8	7.2
M.DELETE	Delete Permanent File or Non-SYSGEN Memory Partition	1,X'77'	H.FISE,14	6.4
M_DELETER	Delete Resource	2,X'24'	H.VOMM,5	7.2
M.DELR	Delete Resource	2,X'24'	H.VOMM,5	6.2
M.DELTSK	Delete Task	1,X'5A'	H.REXS,31	6.2
M_DELTSK	Delete Task	1,X'5A'	H.REXS,31	7.2
M.DEVID	Get Device Mnemonic or Type Code	1,X'14'	H.REXS,71	6.2
M_DEVID	Get Device Mnemonic or Type Code	1,X'14'	H.REXS,71	7.2
M.DFCB	Create File Control Block	N/A	N/A	5.9.1
M.DIR	Create Directory	2,X'23'	H.VOMM,4	6.2
M_DIR	Create Directory	2,X'23'	H.VOMM,4	7.2
M.DISCON	Disconnect Task from Interrupt	1,X'5D'	H.REXS,38	6.2
M_DISCON	Disconnect Task from Interrupt	1,X'5D'	H.REXS,38	7.2
M_DISMOUNT	Dismount Volume	2,X'4A'	H.REMM,19	7.2
M.DLTT	Delete Timer Entry	1,X'47'	H.REXS,6	6.2
M_DLTT	Delete Timer Entry	1,X'47'	H.REXS,6	7.2
M.DMOUNT	Dismount Volume	2,X'4A'	H.REMM,19	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.DSMI	Disable Message Task Interrupt	1,X'2E'	H.REXS,57	6.2
M_DSMI	Disable Message Task Interrupt	1,X'2E'	H.REXS,57	7.2
M.DSUB	Disable User Break Interrupt	1,X'12'	H.REXS,73	6.2
M_DSUB	Disable User Break Interrupt	1,X'12'	H.REXS,73	7.2
M.DUMP	Memory Dump Request	1,X'4F'	H.REXS,12	6.2
M_DUMP	Memory Dump Request	1,X'4F'	H.REXS,12	7.2
M.EAWAIT	End Action Wait	1,X'1D'	H.EXEC,40	6.2
M.ENMI	Enable Message Task Interrupt	1,X'2F'	H.REXS,58	6.2
M_ENMI	Enable Message Task Interrupt	1,X'2F'	H.REXS,58	7.2
M.ENUB	Enable User Break Interrupt	1,X'13'	H.REXS,72	6.2
M_ENUB	Enable User Break Interrupt	1,X'13'	H.REXS,72	7.2
M.ENVRMT	Get Task Environment	2,X'5E'	H.REXS,85	6.2
M_ENVRMT	Get Task Environment	2,X'5E'	H.REXS,85	7.2
M.EXCL	Free Shared Memory	1,X'79'	H.ALOC,14	6.4
M.EXCLUDE	Exclude Memory Partition	2,X'41'	H.REMM,14	6.2
M_EXCLUDE	Exclude Shared Image	2,X'41'	H.REMM,14	7.2
M.EXIT	Terminate Task Execution	1,X'55'	H.REXS,18	6.2
M_EXIT	Terminate Task Execution	1,X'55'	H.REXS,18	7.2
M.EXTD	Extend File	2,X'25'	H.VOMM,6	6.2
M_EXTENDFILE	Extend File	2,X'25'	H.VOMM,6	7.2
M_EXTSTS	Exit With Status	2,X'5F'	H.REXS,86	7.2
M.FADD	Permanent File Address Inquiry	1,X'43'	H.MONS,2	6.4

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.FD	Free Dynamic Extended Indexed Data Space	1,X'6A'	H.REMM,9	6.2
M.FE	Free Dynamic Task Execution Space	1,X'68'	H.REMM,11	6.2
M.FILE	Open File	1,X'30'	H.IOCS,1	6.4
M_FREEMEMBYTES	Free Memory in Byte Increments	2,X'4C'	H.REMM,29	7.2
M.FSLR	Release Synchronization File Lock	1,X'24'	H.FISE,25	6.4
M.FSLS	Set Synchronization File Lock	1,X'23'	H.FISE,24	6.4
M.FWRD	Advance Record	1,X'33'	H.IOCS,7	6.2
	Advance File	1,X'34'	H.IOCS,8	6.2
M.FXLR	Release Exclusive File Lock	1,X'22'	H.FISE,23	6.4
M.FXLS	Set Exclusive File Lock	1,X'21'	H.FISE,22	6.4
M.GADRL	Get Address Limits	1,X'65'	H.REXS,41	6.2
M.GADRL2	Get Address Limits	2,X'7B'	H.REXS,80	6.2
M.GD	Get Dynamic Extended Data Space	1,X'69'	H.REMM,8	6.2
M.GDD	Get Dynamic Extended Discontiguous Data Space	2,X'7C'	H.MEMM,9	6.2
M.GE	Get Dynamic Task Execution Space	1,X'67'	H.REMM,10	6.2
M_GETCTX	Get User Context	2,X'70'	H.EXEC,41	7.2
M.GETDEF	Get Terminal Function Definition	2,X'7A'	H.TSM,15	6.2
M_GETDEF	Get Terminal Function Definition	2,X'7A'	H.TSM,15	7.2
M_GETMEMBYTES	Get Memory in Byte Increments	2,X'4B'	H.REMM,28	7.2
M_GETTIME	Get Current Date and Time	2,X'50'	H.REXS,74	7.2
M.GMSGP	Get Message Parameters	1,X'7A'	H.REXS,35	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_GMSGP	Get Message Parameters	1,X'7A'	H.REXS,35	7.2
M.GRUNP	Get Run Parameters	1,X'7B'	H.REXS,36	6.2
M_GRUNP	Get Run Parameters	1,X'7B'	H.REXS,36	7.2
M.GTIM	Acquire System Date/Time in Any Format	2,X'50'	H.REXS,74	6.2
M_GTIM	Acquire System Date/Time in Any Format	2,X'50'	H.REXS,74	7.2
M.GTSAD	Get TSA Start Address	2,X'7D'	H.REXS,91	6.2
M_GTSAD	Get TSA Start Address	2,X'7D'	H.REXS,91	7.2
M.HOLD	Program Hold Request	1,X'58'	H.REXS,25	6.2
M_HOLD	Program Hold Request	1,X'58'	H.REXS,25	7.2
M.ID	Get Task Number	1,X'64'	H.REXS,32	6.2
M_ID	Get Task Number	1,X'64'	H.REXS,32	7.2
M.INCL	Get Shared Memory	1,X'72'	H.ALOC,13	6.4
M.INCLUDE	Include Memory Partition	2,X'40'	H.REMM,12	6.2
M_INCLUDE	Include Shared Image	2,X'40'	H.REMM,12	7.2
M_INQUIRER	Resource Inquiry	2,X'48'	H.REMM,27	7.2
M.INQUIRY	Resource Inquiry	2,X'48'	H.REMM,27	6.2
M.INT	Activate Task Interrupt	1,X'6F'	H.REXS,47	6.2
M_INT	Activate Task Interrupt	1,X'6F'	H.REXS,47	7.2
M.IPUBS	Set IPU Bias	2,X'5B'	H.REXS,82	6.2
M_IPUBS	Set IPU Bias	2,X'5B'	H.REXS,82	7.2
M_LIMITS	Get Base Mode Task Address Limits	2,X'5D'	H.REXS,84	7.2
M.LOC	Read Descriptor	2,X'2C'	H.VOMM,13	6.2
M.LOCK	Set Exclusive Resource Lock	2,X'44'	H.REMM,23	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_LOCK	Set Exclusive Resource Lock	2,X'44'	H.REMM,23	7.2
M.LOG	Permanent File Log	1,X'73'	H.MONS,33	6.4
M.LOGR	Log Resource or Directory	2,X'29'	H.VOMM,10	6.2
M_LOGR	Log Resource or Directory	2,X'29'	H.VOMM,10	7.2
M.MEM	Create Memory Partition	2,X'22'	H.VOMM,3	6.2
M_MEM	Create Memory Partition	2,X'22'	H.VOMM,3	7.2
M.MEMB	Get Memory in Byte Increments	2,X'4B'	H.REMM,28	6.2
M.MEMFRE	Free Memory in Byte Increments	2,X'4C'	H.REMM,29	6.2
M.MOD	Modify Descriptor	2,X'2A'	H.VOMM,11	6.2
M_MOD	Modify Descriptor	2,X'2A'	H.VOMM,11	7.2
M.MODU	Modify Descriptor User Area	2,X'31'	H.VOMM,26	6.2
M_MODU	Modify Descriptor User Area	2,X'31'	H.VOMM,26	7.2
M.MOUNT	Mount Volume	2,X'49'	H.REMM,17	6.2
M_MOUNT	Mount Volume	2,X'49'	H.REMM,17	7.2
M.MOVE	Move Data to User Address	2,X'62'	H.REXS,89	6.2
M_MOVE	Move Data to User Address	2,X'62'	H.REXS,89	7.2
M.MYID	Get Task Number	1,X'64'	H.REXS,32	6.2
M_MYID	Get Task Number	1,X'64'	H.REXS,32	7.2
M.NEWRRS	Reformat RRS Entry	2,X'54'	H.REXS,76	6.2
M.OLAY	Load Overlay Segment	1,X'50'	H.REXS,13	6.2
	Load and Execute Overlay	1,X'51'	H.REXS,14	6.2
M.OPENR	Open Resource	2,X'42'	H.REMM,21	6.2
M_OPENR	Open Resource	2,X'42'	H.REMM,21	7.2
M_OPTIONDWORD	Task Option Doubleword Inquiry	2,X'C0'	H.REXS,95	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_OPTIONWORD	Task Option Word Inquiry	1,X'4C'	H.REXS,24	7.2
M.OSREAD	Physical Memory Read	2,X'7E'	H.REXS,93	6.2
M_OSREAD	Physical Memory Read	2,X'7E'	H.REXS,93	7.2
M.OSWRIT	Physical Memory Write	2,X'AF'	H.REXS,94	6.2
M_OSWRIT	Physical Memory Write	2,X'AF'	H.REXS,94	7.2
M.PDEV	Physical Device Inquiry	1,X'42'	H.MONS,1	6.4
M.PERM	Change Temporary File to Permanent	1,X'76'	H.FISE,13	6.4
M.PGOD	Task Option Doubleword Inquiry	2,X'C0'	H.REXS,95	6.2
M.PGOW	Task Option Word Inquiry	1,X'4C'	H.REXS,24	6.2
M.PNAM	Reconstruct Pathname	2,X'2F'	H.VOMM,16	6.2
M.PNAMB	Convert Pathname to Pathname Block	2,X'2E'	H.VOMM,15	6.2
M_PNAMB	Convert Pathname to Pathname Block	2,X'2E'	H.VOMM,15	7.2
M.PRIL	Change Priority Level	1,X'4A'	H.REXS,9	6.2
M_PRIL	Change Priority Level	1,X'4A'	H.REXS,9	7.2
M.PRIV	Reinstate Privilege Mode to Privilege Task	2,X'57'	H.REXS,78	6.2
M_PRIVMODE	Reinstate Privilege Mode to Privilege Task	2,X'57'	H.REXS,78	7.2
M.PTSK	Parameter Task Activation	1,X'5F'	H.REXS,40	6.2
M_PTSK	Parameter Task Activation	1,X'5F'	H.REXS,40	7.2
M_PUTCTX	Put User Context	2,X'71'	H.EXEC,42	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.QATIM	Acquire Current Date/Time in ASCII Format	2,X'50'	H.REXS,74	6.2
M_QATIM	Acquire Current Date/Time in ASCII Format	2,X'50'	H.REXS,74	7.2
M.RADDR	Get Real Physical Address	1,X'0E'	H.REXS,90	6.2
M_RADDR	Get Real Physical Address	1,X'0E'	H.REXS,90	7.2
M.RCVR	Receive Message Link Address	1,X'6B'	H.REXS,43	6.2
M_RCVR	Receive Message Link Address	1,X'6B'	H.REXS,43	7.2
M.READ	Read Record	1,X'31'	H.IOCS,3	6.2
M_READ	Read Record	1,X'31'	H.IOCS,3	7.2
M_READD	Read Descriptor	2,X'2C'	H.VOMM,13	7.2
M.RELP	Release Dual-ported Disc/Set Dual-channel ACM Mode	1,X'27'	H.IOCS,27	6.2
M_RELP	Release Dual-ported Disc/Set Dual-channel ACM Mode	1,X'27'	H.IOCS,27	7.2
M.RENAM	Rename File	2,X'2D'	H.VOMM,14	6.2
M_RENAME	Rename File	2,X'2D'	H.VOMM,14	7.2
M.REPLAC	Replace Permanent File	2,X'30'	H.VOMM,23	6.2
M_REPLACE	Replace Permanent File	2,X'30'	H.VOMM,23	7.2
M.RESP	Reserve Dual-ported Disc/Set Single-channel ACM Mode	1,X'26'	H.IOCS,24	6.2
M_RESP	Reserve Dual-ported Disc/Set Single-channel ACM Mode	1,X'26'	H.IOCS,24	7.2
M_REWIND	Rewind File	1,X'37'	H.IOCS,2	7.2
M.REWRIT	Rewrite Descriptor	2,X'2B'	H.VOMM,12	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_REWRIT	Rewrite Descriptor	2,X'2B'	H.VOMM,12	7.2
M.REWRTU	Rewrite Descriptor User Area	2,X'32'	H.VOMM,27	6.2
M_REWRTU	Rewrite Descriptor User Area	2,X'32'	H.VOMM,27	7.2
M.ROPL	Reset Option Lower	2,X'78'	H.TSM,14	6.2
M_ROPL	Reset Option Lower	2,X'78'	H.TSM,14	7.2
M.RRES	Release Channel Reservation	1,X'3B'	H.IOCS,13	6.2
M_RRES	Release Channel Reservation	1,X'3B'	H.IOCS,13	7.2
M.RSML	Resourcemark Lock	1,X'19'	H.REXS,62	6.2
M_RSML	Resourcemark Lock	1,X'19'	H.REXS,62	7.2
M.RSMU	Resourcemark Unlock	1,X'1A'	H.REXS,63	6.2
M_RSMU	Resourcemark Unlock	1,X'1A'	H.REXS,63	7.2
M.RSRV	Reserve Channel	1,X'3A'	H.IOCS,12	6.2
M_RSRV	Reserve Channel	1,X'3A'	H.IOCS,12	7.2
M.RWND	Rewind File	1,X'37'	H.IOCS,2	6.2
M_SETERA	Set Exception Return Address	2,X'79'	H.REXS,81	7.2
M_SETEXA	Set Exception Handler	2,X'5C'	H.REXS,83	7.2
M.SETS	Set User Status Word	1,X'48'	H.REXS,7	6.2
M_SETS	Set User Status Word	1,X'48'	H.REXS,7	7.2
M.SETSYNC	Set Synchronous Resource Lock	2,X'46'	H.REMM,25	6.2
M_SETSYNC	Set Synchronous Resource Lock	2,X'46'	H.REMM,25	7.2
M.SETT	Create Timer Entry	1,X'45'	H.REXS,4	6.2
M_SETT	Create Timer Entry	1,X'45'	H.REXS,4	7.2
M.SHARE	Share Memory with Another Task	1,X'71'	H.ALOC,12	6.4
M.SMSGR	Send Message to Specified Task	1,X'6C'	H.REXS,44	6.2
M_SMSGR	Send Message to Specified Task	1,X'6C'	H.REXS,44	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.SMULK	Unlock and Dequeue Shared Memory	1,X'1F'	H.ALOC,19	6.4
M.SOPL	Set Option Lower	2,X'77'	H.TSM,13	6.2
M_SOPL	Set Option Lower	2,X'77'	H.TSM,13	7.2
M.SRUNR	Send Run Request to Specified Task	1,X'6D'	H.REXS,45	6.2
M_SRUNR	Send Run Request to Specified Task	1,X'6D'	H.REXS,45	7.2
M.SUAR	Set User Abort Receiver Address	1,X'60'	H.REXS,26	6.2
M_SUAR	Set User Abort Receiver Address	1,X'60'	H.REXS,26	7.2
M.SUME	Resume Task Execution	1,X'53'	H.REXS,16	6.2
M_SUME	Resume Task Execution	1,X'53'	H.REXS,16	7.2
M.SURE	Suspend/Resume	5,X'00'	N/A	6.2
M_SURE	Suspend/Resume	5,X'00'	N/A	7.2
M.SUSP	Suspend Task Execution	1,X'54'	H.REXS,17	6.2
M_SUSP	Suspend Task Execution	1,X'54'	H.REXS,17	7.2
M.SYNCH	Set Synchronous Task Interrupt	1,X'1B'	H.REXS,67	6.2
M_SYNCH	Set Synchronous Task Interrupt	1,X'1B'	H.REXS,67	7.2
M.TBRKON	Trap On-line User's Task	1,X'5C'	H.TSM,6	6.2
M_TBRKON	Trap On-line User's Task	1,X'5C'	H.TSM,6	7.2
M.TDAY	Time-of-Day Inquiry	1,X'4E'	H.REXS,11	6.2
M_TDAY	Time-of-Day Inquiry	1,X'4E'	H.REXS,11	7.2
M.TEMP	Create Temporary File	2,X'21'	H.VOMM,2	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.TEMPER	Change Temporary File to Permanent File	2,X'28'	H.VOMM,9	6.2
M_TEMPFILETOPERM	Change Temporary File to Permanent File	2,X'28'	H.VOMM,9	7.2
M.TRNC	Truncate File	2,X'26'	H.VOMM,7	6.2
M_TRUNCATE	Truncate File	2,X'26'	H.VOMM,7	7.2
M.TSCAN	Scan Terminal Input Buffer	1,X'5B'	H.TSM,2	6.2
M_TSCAN	Scan Terminal Input Buffer	1,X'5B'	H.TSM,2	7.2
M.TSMPC	TSM Procedure Call	2,X'AE'	H.TSM,17	6.2
M_TSMPC	TSM Procedure Call	2,X'AE'	H.TSM,17	7.2
M.TSTE	Arithmetic Exception Inquiry	1,X'4D'	H.REXS,23	6.2
M_TSTE	Arithmetic Exception Inquiry	1,X'4D'	H.REXS,23	7.2
M.TSTS	Test User Status Word	1,X'49'	H.REXS,8	6.2
M_TSTS	Test User Status Word	1,X'49'	H.REXS,8	7.2
M.TSTT	Test Timer Entry	1,X'46'	H.REXS,5	6.2
M_TSTT	Test Timer Entry	1,X'46'	H.REXS,5	7.2
M.TURNON	Activate Program at Given Time of Day	1,X'1E'	H.REXS,66	6.2
M_TURNON	Activate Program at Given Time of Day	1,X'1E'	H.REXS,66	7.2
M.TYPE	System Console Type	1,X'3F'	H.IOCS,14	6.2
M_TYPE	System Console Type	1,X'3F'	H.IOCS,14	7.2
M.UNLOCK	Release Exclusive Resource Lock	2,X'45'	H.REMM,24	6.2
M_UNLOCK	Release Exclusive Resource Lock	2,X'45'	H.REMM,24	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_UNPRIVMODE	Change Task to Unprivileged Mode	2,X'58'	H.REXS,79	7.2
M.UNSYNC	Release Synchronous Resource Lock	2,X'47'	H.REMM,26	6.2
M_UNSYNC	Release Synchronous Resource Lock	2,X'47'	H.REMM,26	7.2
M.UPRIV	Change Task to Unprivileged Mode	2,X'58'	H.REXS,79	6.2
M.UPSP	Upspace	1,X'10'	H.IOCS,20	6.2
M_UPSP	Upspace	1,X'10'	H.IOCS,20	7.2
M.USER	User Name Specification	1,X'74'	H.MONS,34	6.4
M.VADDR	Validate Address Range	2,X'59'	H.REXS,33	6.2
M_VADDR	Validate Address Range	2,X'59'	H.REXS,33	7.2
M.WAIT	Wait I/O	1,X'3C'	H.IOCS,25	6.2
M_WAIT	Wait I/O	1,X'3C'	H.IOCS,25	7.2
M.WEOF	Write EOF	1,X'38'	H.IOCS,5	6.2
M.WRIT	Write Record	1,X'32'	H.IOCS,4	6.2
M_WRITE	Write Record	1,X'32'	H.IOCS,4	7.2
M_WRITEEOF	Write EOF	1,X'38'	H.IOCS,5	7.2
M.XBRKR	Exit from Task Interrupt Level	1,X'70'	H.REXS,48	6.2
M_XBRKR	Exit from Task Interrupt Level	N/A	N/A	7.2
M.XIEA	No-wait I/O End-action Return	1,X'2C'	H.IOCS,34	6.2
M_XIEA	No-wait I/O End-action Return	N/A	N/A	7.2
M.XMEA	Exit from Message End-action Routine	1,X'7E'	H.REXS,50	6.2
M_XMEA	Exit from Message End-action Routine	N/A	N/A	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.XMSGR	Exit from Message Receiver	1,X'5E'	H.REXS,39	6.2
M_XMSGR	Exit from Message Receiver	N/A	N/A	7.2
M.XREA	Exit from Run Request End-action Routine	1,X'7F'	H.REXS,51	6.2
M_XREA	Exit from Run Request End-action Routine	N/A	N/A	7.2
M.XRUNR	Exit Run Receiver	1,X'7D'	H.REXS,49	6.2
M_XRUNR	Exit Run Receiver	N/A	N/A	7.2
M.XTIME	Task CPU Execution Time	1,X'2D'	H.REXS,65	6.2
M_XTIME	Task CPU Execution Time	1,X'2D'	H.REXS,65	7.2
N/A	Allocate File Space	N/A	H.VOMM,19	6.3
N/A	Allocate Resource Descriptor	N/A	H.VOMM,17	6.3
N/A	Create Temporary File	N/A	H.VOMM,24	6.3
N/A	Deallocate File Space	N/A	H.VOMM,20	6.3
N/A	Deallocate Resource Descriptor	N/A	H.VOMM,18	6.3
N/A	Debug Link Service	1,X'66'	H.REXS,42	6.3
N/A	Debug Link Service-Base Mode	1,X'66'	H.REXS,42	7.3
N/A	Eject/Purge Routine	1,X'0D'	H.IOCS,22	6.3
N/A	Eject/Purge Routine-Base Mode	1,X'0D'	H.IOCS,22	7.3
N/A	Erase or Punch Trailer	1,X'3E'	H.IOCS,21	6.3
N/A	Erase or Punch Trailer - Base Mode	1,X'3E'	H.IOCS,21	7.3
N/A	Execute Channel Program	1,X'25'	H.IOCS,10	6.3
N/A	Execute Channel Program - Base Mode	1,X'25'	H.IOCS,10	7.3

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
N/A	Get Extended Memory Array	2,X'7F'	H.MEMM,14	6.3
N/A	Get Extended Memory Array - Base Mode	2,X'7F'	H.MEMM,14	7.3
N/A	Read/Write Authorization File	N/A	H.VOMM,25	6.3
N/A	Release FHD Port	1,X'27'	H.IOCS,27	6.3
N/A	Release FHD Port - Base Mode	1,X'27'	H.IOCS,27	7.3
N/A	Reserve FHD Port	1,X'26'	H.IOCS,24	6.3
N/A	Reserve FHD Port- Base Mode	1,X'26'	H.IOCS,24	7.3
N/A	Reserved for Interactive Debugger	2,X'56'	H.REXS,30	N/A
N/A	Reserved for Rapid File Allocation:			N/A
	Zero MDT	2,X'AA'	H.MDT,1	
	Locate/Read MDT Entry	2,X'AB'	H.MDT,2	
	Update/Create MDT Entry	2,X'AC'	H.MDT,3	
	Delete MDT Entry	2,X'AD'	H.MDT,4	
N/A	Set Tabs in UDT	1,X'59'	H.TSM,5	N/A
N/A	TSM Task Detach	1,X'20'	H.TSM,3	N/A

B.2 Alphabetic Listing

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Abort Self	M.BORT	1,X'57'	H.REXS,20	6.2
	M_BORT	1,X'57'	H.REXS,20	7.2
Abort Specified Task	M.BORT	1,X'56'	H.REXS,19	6.2
	M_BORT	1,X'56'	H.REXS,19	7.2
Abort With Extended Message	M.BORT	1,X'62'	H.REXS,28	6.2
	M_BORT	1,X'62'	H.REXS,28	7.2
Acquire Current Date/Time in ASCII Format	M.QATIM	2,X'50'	H.REXS,74	6.2
	M_QATIM	2,X'50'	H.REXS,74	7.2
Acquire Current Date/Time in Binary Format	M.BTIM	2,X'50'	H.REXS,74	6.2
	M_BTIM	2,X'50'	H.REXS,74	7.2
Acquire Current Date/Time in Byte Binary Format	M.BBTIM	2,X'50'	H.REXS,74	6.2
	M_BBTIM	2,X'50'	H.REXS,74	7.2
Acquire System Date/Time in Any Format	M.GTIM	2,X'50'	H.REXS,74	6.2
	M_GTIM	2,X'50'	H.REXS,74	7.2
Activate Program at Given Time of Day	M.TURNON	1,X'1E'	H.REXS,66	6.2
	M_TURNON	1,X'1E'	H.REXS,66	7.2
Activate Task	M.ACTV	1,X'52'	H.REXS,15	6.2
	M_ACTV	1,X'52'	H.REXS,15	7.2
Activate Task Interrupt	M.INT	1,X'6F'	H.REXS,47	6.2
	M_INT	1,X'6F'	H.REXS,47	7.2
Advance File	M.FWRD	1,X'34'	H.IOCS,8	6.2
	M_ADVANCE	1,X'34'	H.IOCS,8	7.2
Advance Record	M.FWRD	1,X'33'	H.IOCS,7	6.2
	M_ADVANCE	1,X'33'	H.IOCS,7	7.2
Allocate File or Peripheral Device	M.ALOC	1,X'40'	H.MONS,21	6.4
Allocate File Space	N/A	N/A	H.VOMM,19	6.3
Allocate Resource Descriptor	N/A	N/A	H.VOMM,17	6.3
Arithmetic Exception Inquiry	M.TSTE	1,X'4D'	H.REXS,23	6.2
	M_TSTE	1,X'4D'	H.REXS,23	7.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Assign and Allocate	M.ASSN	2,X'52'	H.REXS,21	6.2
Resource	M_ASSIGN	2,X'52'	H.REXS,21	7.2
Backspace File	M.BACK	1,X'36'	H.IOCS,19	6.2
	M_BACKSPACE	1,X'36'	H.IOCS,19	7.2
Backspace Record	M.BACK	1,X'35'	H.IOCS,9	6.2
	M_BACKSPACE	1,X'35'	H.IOCS,9	7.2
Batch Job Entry	M.BATCH	2,X'55'	H.REXS,27	6.2
	M_BATCH	2,X'55'	H.REXS,27	7.2
Break/Task	M.BRK	1,X'6E'	H.REXS,46	6.2
Interrupt Link/Unlink	M_BRK	1,X'6E'	H.REXS,46	7.2
Change Defaults	M.DEFT	2,X'27'	H.VOMM,8	6.2
	M_DEFT	2,X'27'	H.VOMM,8	7.2
Change Priority	M.PRIL	1,X'4A'	H.REXS,9	6.2
Level	M_PRIL	1,X'4A'	H.REXS,9	7.2
Change Task to	M.UPRIV	2,X'58'	H.REXS,79	6.2
Unprivileged Mode	M_UNPRIVMODE	2,X'58'	H.REXS,79	7.2
Change Temporary File to Permanent	M.PERM	1,X'76'	H.FISE,13	6.4
Change Temporary	M.TEMPER	2,X'28'	H.VOMM,9	6.2
File to Permanent File	M_TEMPFILETOPERM	2,X'28'	H.VOMM,9	7.2
Close File	M.CLSE	1,X'39'	H.IOCS,23	6.2
	M_CLSE	1,X'39'	H.IOCS,23	7.2
Close Resource	M.CLOSER	2,X'43'	H.REMM,22	6.2
	M_CLOSER	2,X'43'	H.REMM,22	7.2
Connect Task to	M.CONN	1,X'4B'	H.REXS,10	6.2
Interrupt	M_CONN	1,X'4B'	H.REXS,10	7.2
Convert ASCII	M.CONABB	2,X'51'	H.REXS,75	6.2
Date/Time to Byte Binary Format	M_CONABB	2,X'51'	H.REXS,75	7.2
Convert ASCII	M.CONASB	2,X'51'	H.REXS,75	6.2
Date/Time to Standard Binary	M_CONASB	2,X'51'	H.REXS,75	7.2
Convert ASCII	M.CONADB	1,X'28'	H.TSM,7	6.2
Decimal to Binary	M_CONADB	1,X'28'	H.TSM,7	7.2
Convert ASCII Hex	M.CONAHB	1,X'29'	H.TSM,8	6.2
to Binary	M_CONAHB	1,X'29'	H.TSM,8	7.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Convert Binary Date/Time to ASCII Format	M.CONBAF M_CONBAF	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert Binary Date/Time to Byte Binary	M.CONBBY M_CONBBY	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert Binary to ASCII Decimal	M.CONBAD M_CONBAD	1,X'2A' 1,X'2A'	H.TSM,9 H.TSM,9	6.2 7.2
Convert Binary to ASCII Hex	M.CONBAH M_CONBAH	1,X'2B' 1,X'2B'	H.TSM,10 H.TSM,10	6.2 7.2
Convert Byte Binary Date/Time to ASCII	M.CONBBA M_CONBBA	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert Byte Binary Date/Time to Binary	M.CONBYB M_CONBYB	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert Pathname to Pathname Block	M.PNAMB M_PNAMB	2,X'2E' 2,X'2E'	H.VOMM,15 H.VOMM,15	6.2 7.2
Convert System Date/Time Format	M.CTIM M_CTIM	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert Time	M_CONVERTTIME	2,X'51'	H.REXS,75	7.2
Create Directory	M.DIR M_DIR	2,X'23' 2,X'23'	H.VOMM,4 H.VOMM,4	6.2 7.2
Create File Control Block	M.DFCB	N/A	N/A	5.9.1
Create File Control Block	M_CREATEFCB	N/A	N/A	7.2
Create Memory Partition	M.MEM M_MEM	2,X'22' 2,X'22'	H.VOMM,3 H.VOMM,3	6.2 7.2
Create Permanent File	M.CREATE	1,X'75'	H.FISE,12	6.4
Create Permanent File	M.CPERM M_CREATEP	2,X'20' 2,X'20'	H.VOMM,1 H.VOMM,1	6.2 7.2
Create Temporary File	M.TEMP M_CREATET	2,X'21' 2,X'21'	H.VOMM,2 H.VOMM,2	6.2 7.2
Create Temporary File	N/A	N/A	H.VOMM,24	6.3

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Create Timer Entry	M.SETT	1,X'45'	H.REXS,4	6.2
	M_SETT	1,X'45'	H.REXS,4	7.2
Date and Time Inquiry	M.DATE	1,X'15'	H.REXS,70	6.2
	M_DATE	1,X'15'	H.REXS,70	7.2
Deallocate File or Peripheral Device	M.DALC	1,X'41'	H.MONS,22	6.4
Deallocate File Space	N/A	N/A	H.VOMM,20	6.3
Deallocate Resource Descriptor	N/A	N/A	H.VOMM,18	6.3
Deassign and Deallocate Resource	M.DASN	2,X'53'	H.REXS,22	6.2
	M_DEASSIGN	2,X'53'	H.REXS,22	7.2
Debug Link Service	N/A	1,X'66'	H.REXS,42	6.3
Debug Link Service-Base Mode	N/A	1,X'66'	H.REXS,42	7.3
Delete Permanent File or Non-SYSGEN Memory Partition	M.DELETE	1,X'77'	H.FISE,14	6.4
Delete Resource	M.DELR	2,X'24'	H.VOMM,5	6.2
	M_DELETER	2,X'24'	H.VOMM,5	7.2
Delete Task	M.DELTSK	1,X'5A'	H.REXS,31	6.2
	M_DELTSK	1,X'5A'	H.REXS,31	7.2
Delete Timer Entry	M.DLTT	1,X'47'	H.REXS,6	6.2
	M_DLTT	1,X'47'	H.REXS,6	7.2
Disable Message Task Interrupt	M.DSMI	1,X'2E'	H.REXS,57	6.2
	M_DSMI	1,X'2E'	H.REXS,57	7.2
Disable User Break Interrupt	M.DSUB	1,X'12'	H.REXS,73	6.2
	M_DSUB	1,X'12'	H.REXS,73	7.2
Disconnect Task from Interrupt	M.DISCON	1,X'5D'	H.REXS,38	6.2
	M_DISCON	1,X'5D'	H.REXS,38	7.2
Dismount Volume	M.DMOUNT	2,X'4A'	H.REMM,19	6.2
	M_DISMOUNT	2,X'4A'	H.REMM,19	7.2
Eject/Purge Routine	N/A	1,X'0D'	H.IOCS,22	6.3
Eject/Purge Routine-Base Mode	N/A	1,X'0D'	H.IOCS,22	7.3
Enable Message Task Interrupt	M.ENMI	1,X'2F'	H.REXS,58	6.2
	M_ENMI	1,X'2F'	H.REXS,58	7.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Enable User Break Interrupt	M.ENUB	1,X'13'	H.REXS,72	6.2
	M_ENUB	1,X'13'	H.REXS,72	7.2
End Action Wait	M.EAWAIT	1,X'1D'	H.EXEC,40	6.2
	M_AWAITACTION	1,X'1D'	H.EXEC,40	7.2
Erase or Punch Trailer	N/A	1,X'3E'	H.IOCS,21	6.3
Erase or Punch Trailer - Base Mode	N/A	1,X'3E'	H.IOCS,21	7.3
Exclude Memory Partition	M.EXCLUDE	2,X'41'	H.REMM,14	6.2
Exclude Shared Image	M_EXCLUDE	2,X'41'	H.REMM,14	7.2
Execute Channel Program	N/A	1,X'25'	H.IOCS,10	6.3
Execute Channel Program - Base Mode	N/A	1,X'25'	H.IOCS,10	7.3
Execute Channel Program File Control Block	M_CHANPROGFCB	N/A	N/A	7.2
Exit from Message End-action Routine	M.XMEA	1,X'7E'	H.REXS,50	6.2
	M_XMEA	N/A	N/A	7.2
Exit from Message Receiver	M.XMSGR	1,X'5E'	H.REXS,39	6.2
	M_XMSGR	N/A	N/A	7.2
Exit from Run Request End-action Routine	M.XREA	1,X'7F'	H.REXS,51	6.2
	M_XREA	N/A	N/A	7.2
Exit from Task Interrupt Level	M.BRKXIT	1,X'70'	H.REXS,48	6.2
	M _BRKXIT	N/A	N/A	7.2
	M.XBRKR	1,X'70'	H.REXS,48	6.2
	M _XBRKR	N/A	N/A	7.2
Exit Run Receiver	M.XRUNR	1,X'7D'	H.REXS,49	6.2
	M_XRUNR	N/A	N/A	7.2
Exit With Status	M_EXTSTS	2,X'5F'	H.REXS,86	7.2
Extend File	M.EXTD	2,X'25'	H.VOMM,6	6.2
	M_EXTENDFILE	2,X'25'	H.VOMM,6	7.2
Free Dynamic Extended Indexed Data Space	M.FD	1,X'6A'	H.REMM,9	6.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Free Dynamic Task Execution Space	M.FE	1,X'68'	H.REMM,11	6.2
Free Memory in Byte Increments	M.MEMFRE	2,X'4C'	H.REMM,29	6.2
	M_FREEMEMBYTES	2,X'4C'	H.REMM,29	7.2
Free Shared Memory	M.EXCL	1,X'79'	H.ALOC,14	6.4
Get Address Limits	M.GADRL	1,X'65'	H.REXS,41	6.2
Get Address Limits	M.GADRL2	2,X'7B'	H.REXS,80	6.2
Get Base Mode Task Address Limits	M_LIMITS	2,X'5D'	H.REXS,84	7.2
Get Command Line	M.CMD	2,X'61'	H.REXS,88	6.2
	M_CMD	2,X'61'	H.REXS,88	7.2
Get Current Date and Time	M_GETTIME	2,X'50'	H.REXS,74	7.2
Get Device Mnemonic or Type Code	M.DEVID	1,X'14'	H.REXS,71	6.2
	M_DEVID	1,X'14'	H.REXS,71	7.2
Get Dynamic Extended Data Space	M.GD	1,X'69'	H.REMM,8	6.2
Get Dynamic Extended Discontiguous Data Space	M.GDD	2,X'7C'	H.MEMM,9	6.2
Get Dynamic Task Execution Space	M.GE	1,X'67'	H.REMM,10	6.2
Get Extended Memory Array	N/A	2,X'7F'	H.MEMM,14	6.3
Get Extended Memory Array - Base Mode	N/A	2,X'7F'	H.MEMM,14	7.3
Get Memory in Byte Increments	M.MEMB	2,X'4B'	H.REMM,28	6.2
	M_GETMEMBYTES	2,X'4B'	H.REMM,28	7.2
Get Message Parameters	M.GMSGP	1,X'7A'	H.REXS,35	6.2
	M_GMSGP	1,X'7A'	H.REXS,35	7.2
Get Real Physical Address	M.RADDR	1,X'0E'	H.REXS,90	6.2
	M_RADDR	1,X'0E'	H.REXS,90	7.2
Get Run Parameters	M.GRUNP	1,X'7B'	H.REXS,36	6.2
	M_GRUNP	1,X'7B'	H.REXS,36	7.2
Get Shared Memory	M.INCL	1,X'72'	H.ALOC,13	6.4

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Get Task	M.ENVRMT	2,X'5E'	H.REXS,85	6.2
Environment	M_ENVRMT	2,X'5E'	H.REXS,85	7.2
Get Task Number	M.ID	1,X'64'	H.REXS,32	6.2
	M_ID	1,X'64'	H.REXS,32	7.2
	M.MYID	1,X'64'	H.REXS,32	6.2
	M_MYID	1,X'64'	H.REXS,32	7.2
Get Terminal	M.GETDEF	2,X'7A'	H.TSM,15	6.2
Function Definition	M_GETDEF	2,X'7A'	H.TSM,15	7.2
Get TSA Start	M.GTSAD	2,X'7D'	H.REXS,91	6.2
Address	M_GTSAD	2,X'7D'	H.REXS,91	7.2
Get User Context	M_GETCTX	2,X'70'	H.EXEC,41	7.2
Include Memory Partition	M.INCLUDE	2,X'40'	H.REMM,12	6.2
Include Shared Image	M_INCLUDE	2,X'40'	H.REMM,12	7.2
Load and Execute	M.DEBUG	1,X'63'	H.REXS,29	6.2
Interactive Debugger	M_DEBUG	1,X'63'	H.REXS,29	7.2
Load Overlay Segment	M.OLAY	1,X'50'	H.REXS,13	6.2
Load and Execute Overlay		1,X'51'	H.REXS,14	6.2
Log Resource	M.LOGR	2,X'29'	H.VOMM,10	6.2
or Directory	M_LOGR	2,X'29'	H.VOMM,10	7.2
Memory Address	M.ADRS	1,X'44'	H.REXS,3	6.2
Inquiry	M_ADRS	1,X'44'	H.REXS,3	7.2
Memory Dump	M.DUMP	1,X'4F'	H.REXS,12	6.2
Request	M_DUMP	1,X'4F'	H.REXS,12	7.2
Modify Descriptor	M.MOD	2,X'2A'	H.VOMM,11	6.2
	M_MOD	2,X'2A'	H.VOMM,11	7.2
Modify Descriptor	M.MODU	2,X'31'	H.VOMM,26	6.2
User Area	M_MODU	2,X'31'	H.VOMM,26	7.2
Mount Volume	M.MOUNT	2,X'49'	H.REMM,17	6.2
	M_MOUNT	2,X'49'	H.REMM,17	7.2
Move Data to	M.MOVE	2,X'62'	H.REXS,89	6.2
User Address	M_MOVE	2,X'62'	H.REXS,89	7.2
No-wait I/O End-	M.XIEA	1,X'2C'	H.IOCS,34	6.2
action Return	M_XIEA	N/A	N/A	7.2
Open File	M.FILE	1,X'30'	H.IOCS,1	6.4

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Open Resource	M.OPENR	2,X'42'	H.REMM,21	6.2
	M_OPENR	2,X'42'	H.REMM,21	7.2
Parameter Task Activation	M.PTSK	1,X'5F'	H.REXS,40	6.2
	M_PTSK	1,X'5F'	H.REXS,40	7.2
Permanent File Address Inquiry	M.FADD	1,X'43'	H.MONS,2	6.4
Permanent File Log	M.LOG	1,X'73'	H.MONS,33	6.4
Physical Device Inquiry	M.PDEV	1,X'42'	H.MONS,1	6.4
Physical Memory Read	M.OSREAD	2,X'7E'	H.REXS,93	6.2
	M_OSREAD	2,X'7E'	H.REXS,93	7.2
Physical Memory Write	M.OSWRIT	2,X'AF'	H.REXS,94	6.2
	M_OSWRIT	2,X'AF'	H.REXS,94	7.2
Program Hold Request	M.HOLD	1,X'58'	H.REXS,25	6.2
	M_HOLD	1,X'58'	H.REXS,25	7.2
Put User Context	M_PUTCTX	2,X'71'	H.EXEC,42	7.2
Read Descriptor	M.LOC	2,X'2C'	H.VOMM,13	6.2
	M_READD	2,X'2C'	H.VOMM,13	7.2
Read Record	M.READ	1,X'31'	H.IOCS,3	6.2
	M_READ	1,X'31'	H.IOCS,3	7.2
Read/Write Authorization File	N/A	N/A	H.VOMM,25	6.3
Receive Message Link Address	M.RCVR	1,X'6B'	H.REXS,43	6.2
	M_RCVR	1,X'6B'	H.REXS,43	7.2
Reconstruct Pathname	M.PNAM	2,X'2F'	H.VOMM,16	6.2
	M_CONSTRUCTPATH	2,X'2F'	H.VOMM,16	7.2
Reformat RRS Entry	M.NEWRRS	2,X'54'	H.REXS,76	6.2
Reinstate Privilege Mode to Privilege Task	M.PRIV	2,X'57'	H.REXS,78	6.2
	M_PRIVMODE	2,X'57'	H.REXS,78	7.2
Release Channel Reservation	M.RRES	1,X'3B'	H.IOCS,13	6.2
	M_RRES	1,X'3B'	H.IOCS,13	7.2
Release Dual-ported Disc/Set Dual-channel ACM Mode	M.RELP	1,X'27'	H.IOCS,27	6.2
	M_RELP	1,X'27'	H.IOCS,27	7.2
Release Exclusive File Lock	M.FXLR	1,X'22'	H.FISE,23	6.4
Release Exclusive Resource Lock	M.UNLOCK	2,X'45'	H.REMM,24	6.2
	M_UNLOCK	2,X'45'	H.REMM,24	7.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Release FHD Port	N/A	1,X'27'	H.IOCS,27	6.3
Release FHD Port-Base Mode	N/A	1,X'27'	H.IOCS,27	7.3
Release Synchronization File Lock	M.FSLR	1,X'24'	H.FISE,25	6.4
Release Synchronous Resource Lock	M.UNSYNC	2,X'47'	H.REMM,26	6.2
	M_UNSYNC	2,X'47'	H.REMM,26	7.2
Rename File	M.RENAM	2,X'2D'	H.VOMM,14	6.2
	M_RENAME	2,X'2D'	H.VOMM,14	7.2
Replace Permanent File	M.REPLAC	2,X'30'	H.VOMM,23	6.2
	M_REPLACE	2,X'30'	H.VOMM,23	7.2
Reserve Channel	M.RSRV	1,X'3A'	H.IOCS,12	6.2
	M_RSRV	1,X'3A'	H.IOCS,12	7.2
Reserve Dual-ported Disc/Set Single-channel ACM Mode	M.RESP	1,X'26'	H.IOCS,24	6.2
	M_RESP	1,X'26'	H.IOCS,24	7.2
Reserve FHD Port	N/A	1,X'26'	H.IOCS,24	6.3
Reserve FHD Port-Base Mode	N/A	1,X'26'	H.IOCS,24	7.3
Reserved for Interactive Debugger	N/A	2,X'56'	H.REXS,30	N/A
Reserved for Rapid File Allocation:	N/A			N/A
Zero MDT		2,X'AA'	H.MDT,1	
Locate/Read MDT Entry		2,X'AB'	H.MDT,2	
Update/Create MDT Entry		2,X'AC'	H.MDT,3	
Delete MDT Entry		2,X'AD'	H.MDT,4	
Reset Option Lower	M.ROPL	2,X'78'	H.TSM,14	6.2
	M_ROPL	2,X'78'	H.TSM,14	7.2
Resource Inquiry	M.INQUIRY	2,X'48'	H.REMM,27	6.2
	M_INQUIRER	2,X'48'	H.REMM,27	7.2
Resourcemark Lock	M.RSML	1,X'19'	H.REXS,62	6.2
	M_RSML	1,X'19'	H.REXS,62	7.2
Resourcemark Unlock	M.RSMU	1,X'1A'	H.REXS,63	6.2
	M_RSMU	1,X'1A'	H.REXS,63	7.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Resume Task	M.SUME	1,X'53'	H.REXS,16	6.2
Execution	M_SUME	1,X'53'	H.REXS,16	7.2
Rewind File	M.RWND	1,X'37'	H.IOCS,2	6.2
	M_REWIND	1,X'37'	H.IOCS,2	7.2
Rewrite Descriptor	M.REWRIT	2,X'2B'	H.VOMM,12	6.2
	M_REWRIT	2,X'2B'	H.VOMM,12	7.2
Rewrite Descriptor	M.REWRTU	2,X'32'	H.VOMM,27	6.2
User Area	M_REWRTU	2,X'32'	H.VOMM,27	7.2
Scan Terminal	M.TSCAN	1,X'5B'	H.TSM,2	6.2
Input Buffer	M_TSCAN	1,X'5B'	H.TSM,2	7.2
Send Message to	M.SMSGR	1,X'6C'	H.REXS,44	6.2
Specified Task	M_SMSGR	1,X'6C'	H.REXS,44	7.2
Send Run Request	M.SRUNR	1,X'6D'	H.REXS,45	6.2
to Specified Task	M_SRUNR	1,X'6D'	H.REXS,45	7.2
Set Asynchronous	M.ASYNCH	1,X'1C'	H.REXS,68	6.2
Task Interrupt	M_ASYNCH	1,X'1C'	H.REXS,68	7.2
Set Exception Handler	M_SETEXA	2,X'5C'	H.REXS,83	7.2
Set Exception Return Address	M_SETERA	2,X'79'	H.REXS,81	7.2
Set Exclusive File Lock	M.FXLS	1,X'21'	H.FISE,22	6.4
Set Exclusive	M.LOCK	2,X'44'	H.REMM,23	6.2
Resource Lock	M_LOCK	2,X'44'	H.REMM,23	7.2
Set IPU Bias	M.IPUBS	2,X'5B'	H.REXS,82	6.2
	M_IPUBS	2,X'5B'	H.REXS,82	7.2
Set Option Lower	M.SOPL	2,X'77'	H.TSM,13	6.2
	M_SOPL	2,X'77'	H.TSM,13	7.2
Set Synchronization File Lock	M.FSLS	1,X'23'	H.FISE,24	6.4
Set Synchronous	M.SETSYNC	2,X'46'	H.REMM,25	6.2
Resource Lock	M_SETSYNC	2,X'46'	H.REMM,25	7.2
Set Synchronous	M.SYNCH	1,X'1B'	H.REXS,67	6.2
Task Interrupt	M_SYNCH	1,X'1B'	H.REXS,67	7.2
Set Tabs in UDT	N/A	1,X'59'	H.TSM,5	N/A

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Set User Abort	M.SUAR	1,X'60'	H.REXS,26	6.2
Receiver Address	M_SUAR	1,X'60'	H.REXS,26	7.2
Set User Status	M.SETS	1,X'48'	H.REXS,7	6.2
Word	M_SETS	1,X'48'	H.REXS,7	7.2
Share Memory with Another Task	M.SHARE	1,X'71'	H.ALOC,12	6.4
Submit Job from Disc File	M.CDJS	1,X'61'	H.MONS,27	6.4
Suspend/Resume	M.SURE	5,X'00'	N/A	6.2
	M_SURE	5,X'00'	N/A	7.2
Suspend Task	M.SUSP	1,X'54'	H.REXS,17	6.2
Execution	M_SUSP	1,X'54'	H.REXS,17	7.2
System Console Type	M.TYPE	1,X'3F'	H.IOCS,14	6.2
	M_TYPE	1,X'3F'	H.IOCS,14	7.2
System Console Wait	M.CWAT	1,X'3D'	H.IOCS,26	6.2
	M_CWAT	1,X'3D'	H.IOCS,26	7.2
Task CPU Execution	M.XTIME	1,X'2D'	H.REXS,65	6.2
Time	M_XTIME	1,X'2D'	H.REXS,65	7.2
Task Option	M.PGOD	2,X'C0'	H.REXS,95	6.2
Doubleword Inquiry	M_OPTIONDWORD	2,X'C0'	H.REXS,95	7.2
Task Option Word	M.PGOW	1,X'4C'	H.REXS,24	6.2
Inquiry	M_OPTIONWORD	1,X'4C'	H.REXS,24	7.2
Terminate Task	M.EXIT	1,X'55'	H.REXS,18	6.2
Execution	M_EXIT	1,X'55'	H.REXS,18	7.2
Test Timer Entry	M.TSTT	1,X'46'	H.REXS,5	6.2
	M_TSTT	1,X'46'	H.REXS,5	7.2
Test User Status	M.TSTS	1,X'49'	H.REXS,8	6.2
Word	M_TSTS	1,X'49'	H.REXS,8	7.2
Time-of-Day Inquiry	M.TDAY	1,X'4E'	H.REXS,11	6.2
	M_TDAY	1,X'4E'	H.REXS,11	7.2
Trap On-line User's	M.TBRKON	1,X'5C'	H.TSM,6	6.2
Task	M_TBRKON	1,X'5C'	H.TSM,6	7.2
Truncate File	M.TRNC	2,X'26'	H.VOMM,7	6.2
	M_TRUNCATE	2,X'26'	H.VOMM,7	7.2
TSM Procedure	M.TSMPC	2,X'AE'	H.TSM,17	6.2
Call	M_TSMPC	2,X'AE'	H.TSM,17	7.2
TSM Task Detach	N/A	1,X'20'	H.TSM,3	N/A
Unlock and Dequeue Shared Memory	M.SMULK	1,X'1F'	H.ALOC,19	6.4

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Upspace	M.UPSP	1,X'10'	H.IOCS,20	6.2
	M_UPSP	1,X'10'	H.IOCS,20	7.2
User Name Specification	M.USER	1,X'74'	H.MONS,34	6.4
Validate Address Range	M.VADDR	2,X'59'	H.REXS,33	6.2
	M_VADDR	2,X'59'	H.REXS,33	7.2
Wait for Any No-wait Operation Complete, Message Interrupt, or Break Interrupt	M.ANYW	1,X'7C'	H.REXS,37	6.2
	M_ANYWAIT	1,X'7C'	H.REXS,37	7.2
Wait I/O	M.WAIT	1,X'3C'	H.IOCS,25	6.2
	M_WAIT	1,X'3C'	H.IOCS,25	7.2
Write EOF	M.WEOF	1,X'38'	H.IOCS,5	6.2
	M_WRITEEOF	1,X'38'	H.IOCS,5	7.2
Write Record	M.WRIT	1,X'32'	H.IOCS,4	6.2
	M_WRITE	1,X'32'	H.IOCS,4	7.2

B.3 SVC Listing

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
00-0A	Reserved			
0B	Reserved for Vector Processor			
0C	Reserved			
0D	Eject/Purge Routine	H.IOCS,22	N/A	6.3
	Eject/Purge Routine - Base Mode	H.IOCS,22	N/A	7.3
0E	Get Real Physical Address	H.REXS,90	M.RADDR M_RADDR	6.2 7.2
0F	Reserved for Vector Processor		N/A	N/A
10	Upspace	H.IOCS,20	M.UPSP M_UPSP	6.2 7.2
11	Reserved			
12	Disable User Break Interrupt	H.REXS,73	M.DSUB M_DSUB	6.2 7.2
13	Enable User Break Interrupt	H.REXS,72	M.ENUB M_ENUB	6.2 7.2
14	Get Device Mnemonic or Type Code	H.REXS,71	M.DEVID M_DEVID	6.2 7.2
15	Date and Time Inquiry	H.REXS,70	M.DATE M_DATE	6.2 7.2
16	ADI Maximum IOCBs	N/A	M.ADIMAX	N/A
17	ADI I/O	N/A	M.ADIO	N/A
18	ADI EAI	N/A	M.ADIEAI	N/A
19	Resourcemark Lock	H.REXS,62	M.RSML M_RSML	6.2 7.2
1A	Resourcemark Unlock	H.REXS,63	M.RSMU M_RSMU	6.2 7.2
1B	Set Synchronous Task Interrupt	H.REXS,67	M.SYNCH M_SYNCH	6.2 7.2
1C	Set Asynchronous Task Interrupt	H.REXS,68	M.ASYNCH M_ASYNCH	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
1D	End Action Wait	H.EXEC,40	M.EAWAIT M_AWAITACTION	6.2 7.2
1E	Activate Program at Given Time of Day	H.REXS,66	M.TURNON M_TURNON	6.2 7.2
1F	Unlock and Dequeue Shared Memory	H.ALOC,19	M.SMULK	6.4
20	TSM Task Detach	H.TSM,3	N/A	N/A
21	Set Exclusive File Lock	H.FISE,22	M.FXLS	6.4
22	Release Exclusive File Lock	H.FISE,23	M.FXLR	6.4
23	Set Synchronization File Lock	H.FISE,24	M.FSLS	6.4
24	Release Synchronization File Lock	H.FISE,25	M.FSLR	6.4
25	Execute Channel Program	H.IOCS,10	N/A	6.3
	Execute Channel Program - Base Mode	H.IOCS,10	N/A	7.3
26	ReserveFHD Port	H.IOCS,24	N/A	6.3
	Reserve FHD Port - Base Mode		N/A	7.3
	Reserve Dual-ported Disc/Set Single-channel ACM Mode		M.RESP M_RESP	6.2 7.2
27	Release FHD Port	H.IOCS,27	N/A	6.3
	Release FHD Port - Base Mode		N/A	7.3
	Release Dual-ported Disc/Set Dual-channel ACM Mode		M.RELP M_RELP	6.2 7.2
28	Convert ASCII Decimal to Binary	H.TSM,7	M.CONADB M_CONADB	6.2 7.2
29	Convert ASCII Hex to Binary	H.TSM,8	M.CONAHB M_CONAHB	6.2 7.2

SVC Listing

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
2A	Convert Binary to ASCII Decimal	H.TSM,9	M.CONBAD M_CONBAD	6.2 7.2
2B	Convert Binary to ASCII Hex	H.TSM,10	M.CONBAH M_CONBAH	6.2 7.2
2C	No-wait I/O End-action Return	H.IOCS,34	M.XIEA	6.2
2D	Task CPU Execution Time	H.REXS,65	M.XTIME M_XTIME	6.2 7.2
2E	Disable Message Task Interrupt	H.REXS,57	M.DSMI M_DSMI	6.2 7.2
2F	Enable Message Task Interrupt	H.REXS,58	M.ENMI M_ENMI	6.2 7.2
30	Open File	H.IOCS,1	M.FILE	6.4
31	Read Record	H.IOCS,3	M.READ M_READ	6.2 7.2
32	Write Record	H.IOCS,4	M.WRIT M_WRITE	6.2 7.2
33	Advance Record	H.IOCS,7	M.FWRD M_ADVANCE	6.2 7.2
34	Advance File	H.IOCS,8	M.FWRD M_ADVANCE	6.2 7.2
35	Backspace Record	H.IOCS,9	M.BACK M_BACKSPACE	6.2 7.2
36	Backspace File	H.IOCS,19	M.BACK M_BACKSPACE	6.2 7.2
37	Rewind File	H.IOCS,2	M.RWND M_REWIND	6.2 7.2
38	Write EOF	H.IOCS,5	M.WEOF M_WRITEEOF	6.2 7.2
39	Close File	H.IOCS,23	M.CLSE M_CLSE	6.2 7.2
3A	Reserve Channel	H.IOCS,12	M.RSRV M_RSRV	6.2 7.2
3B	Release Channel Reservation	H.IOCS,13	M.RRES M_RRES	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
3C	Wait I/O	H.IOCS,25	M.WAIT M_WAIT	6.2 7.2
3D	System Console Wait	H.IOCS,26	M.CWAT M_CWAT	6.2 7.2
3E	Erase or Punch Trailer	H.IOCS,21	N/A	6.3
	Erase or Punch Trailer - Base Mode	H.IOCS,21	N/A	7.3
3F	System Console Type	H.IOCS,14	M.TYPE M_TYPE	6.2 7.2
40	Allocate File or Peripheral Device	H.MONS,21	M.ALOC	6.4
41	Deallocate File or Peripheral Device	H.MONS,22	M.DALC	6.4
42	Physical Device Inquiry	H.MONS,1	M.PDEV	6.4
43	Permanent File Address Inquiry	H.MONS,2	M.FADD	6.4
44	Memory Address Inquiry	H.REXS,3	M.ADRS M_ADRS	6.2 7.2
45	Create Timer Entry	H.REXS,4	M.SETT M_SETT	6.2 7.2
46	Test Timer Entry	H.REXS,5	M.TSTT M_TSTT	6.2 7.2
47	Delete Timer Entry	H.REXS,6	M.DLTT M_DLTT	6.2 7.2
48	Set User Status Word	H.REXS,7	M.SETS M_SETS	6.2 7.2
49	Test User Status Word	H.REXS,8	M.TSTS M_TSTS	6.2 7.2
4A	Change Priority Level	H.REXS,9	M.PRIL M_PRIL	6.2 7.2
4B	Connect Task to Interrupt	H.REXS,10	M.CONN M_CONN	6.2 7.2
4C	Task Option Word Inquiry	H.REXS,24	M.PGOW M_OPTIONWORD	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
4D	Arithmetic Exception Inquiry	H.REXS,23	M.TSTE M_TSTE	6.2 7.2
4E	Time-of-Day Inquiry	H.REXS,11	M.TDAY M_TDAY	6.2 7.2
4F	Memory Dump Request	H.REXS,12	M.DUMP M_DUMP	6.2 7.2
50	Load Overlay Segment	H.REXS,13	M.OLAY	6.2
51	Load and Execute Overlay	H.REXS,14	M.OLAY	6.2
52	Activate Task	H.REXS,15	M.ACTV M_ACTV	6.2 7.2
53	Resume Task Execution	H.REXS,16	M.SUME M_SUME	6.2 7.2
54	Suspend Task Execution	H.REXS,17	M.SUSP M_SUSP	6.2 7.2
55	Terminate Task Execution	H.REXS,18	M.EXIT M_EXIT	6.2 7.2
56	Abort Specified Task	H.REXS,19	M.BORT M_BORT	6.2 7.2
57	Abort Self	H.REXS,20	M.BORT M_BORT	6.2 7.2
58	Program Hold Request	H.REXS,25	M.HOLD M_HOLD	6.2 7.2
59	Set Tabs in UDT	H.TSM,5	N/A	N/A
5A	Delete Task	H.REXS,31	M.DELTSK M_DELTSK	6.2 7.2
5B	Scan Terminal Input Buffer	H.TSM,2	M.TSCAN M_TSCAN	6.2 7.2
5C	Trap On-line User's Task	H.TSM,6	M.TBRKON M_TBRKON	6.2 7.2
5D	Disconnect Task from Interrupt	H.REXS,38	M.DISCON M_DISCON	6.2 7.2
5E	Exit from Message Receiver	H.REXS,39	M.XMSGR	6.2
5F	Parameter Task Activation	H.REXS,40	M.PTSK M_PTSK	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
60	Set User Abort Receiver Address	H.REXS,26	M.SUAR M_SUAR	6.2 7.2
61	Submit Job from Disc File	H.MONS,27	M.CDJS	6.4
62	Abort With Extended Message	H.REXS,28	M.BORT M_BORT	6.2 7.2
63	Load and Execute Interactive Debugger	H.REXS,29	M.DEBUG M_DEBUG	6.2 7.2
64	Get Task Number	H.REXS,32	M.ID M_ID M.MYID M_MYID	6.2 7.2 6.2 7.2
65	Get Address Limits	H.REXS,41	M.GADRL	6.2
66	Debug Link Service	H.REXS,42	N/A	6.3
	Debug Link Service - Base Mode	H.REXS,42	N/A	7.3
67	Get Dynamic Task Execution Space	H.REMM,10	M.GE	6.2
68	Free Dynamic Task Execution Space	H.REMM,11	M.FE	6.2
69	Get Dynamic Extended Data Space	H.REMM,8	M.GD	6.2
6A	Free Dynamic Extended Indexed Data Space	H.REMM,9	M.FD	6.2
6B	Receive Message Link Address	H.REXS,43	M.RCVR M_RCVR	6.2 7.2
6C	Send Message to Specified Task	H.REXS,44	M.SMSGR M_SMSGR	6.2 7.2
6D	Send Run Request to Specified Task	H.REXS,45	M.SRUNR M_SRUNR	6.2 7.2
6E	Break/Task Interrupt Link/Unlink	H.REXS,46	M.BRK M_BRK	6.2 7.2
6F	Activate Task Interrupt	H.REXS,47	M.INT M_INT	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
70	Exit from Task Interrupt Level	H.REXS,48	M.BRKXIT M.XBRKR	6.2 6.2
71	Share Memory with Another Task	H.ALOC,12	M.SHARE	6.4
72	Get Shared Memory	H.ALOC,13	M.INCL	6.4
73	Permanent File Log	H.MONS,33	M.LOG	6.4
74	User Name Specification	H.MONS,34	M.USER	6.4
75	Create Permanent File	H.FISE,12	M.CREATE	6.4
76	Change Temporary File to Permanent	H.FISE,13	M.PERM	6.4
77	Delete Permanent File or Non-SYSGEN Memory Partition	H.FISE,14	M.DELETE	6.4
78	Reserved			
79	Free Shared Memory	H.ALOC,14	M.EXCL	6.4
7A	Get Message Parameters	H.REXS,35	M.GMSGP M_GMSGP	6.2 7.2
7B	Get Run Parameters	H.REXS,36	M.GRUNP M_GRUNP	6.2 7.2
7C	Wait for Any No-wait Operation Complete, Message Interrupt, or Break Interrupt	H.REXS,37	M.ANYW M_ANYWAIT	6.2 7.2
7D	Exit Run Receiver	H.REXS,49	M.XRUNR	6.2
7E	Exit from Message End-action Routine	H.REXS,50	M.XMEA	6.2
7F	Exit from Run Request End-action Routine	H.REXS,51	M.XREA	6.2
80-FFF	Available for customer use			

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
00-1F	Reserved			
20	Create Permanent File	H.VOMM,1	M.CPERM M_CREATEP	6.2 7.2
21	Create Temporary File	H.VOMM,2	M.TEMP M_CREATET	6.2 7.2
22	Create Memory Partition	H.VOMM,3	M.MEM M_MEM	6.2 7.2
23	Create Directory	H.VOMM,4	M.DIR M_DIR	6.2 7.2
24	Delete Resource	H.VOMM,5	M.DELR M_DELETER	6.2 7.2
25	Extend File	H.VOMM,6	M.EXTD M_EXTENDFILE	6.2 7.2
26	Truncate File	H.VOMM,7	M.TRNC M_TRUNCATE	6.2 7.2
27	Change Defaults	H.VOMM,8	M.DEFT M_DEFT	6.2 7.2
28	Change Temporary File to Permanent File	H.VOMM,9	M.TEMPER M_TEMPFILETOPERM	6.2 7.2
29	Log Resource or Directory	H.VOMM,10	M.LOGR M_LOGR	6.2 7.2
2A	Modify Descriptor	H.VOMM,11	M.MOD M_MOD	6.2 7.2
2B	Rewrite Descriptor	H.VOMM,12	M.REWRIT M_REWRIT	6.2 7.2
2C	Read Descriptor	H.VOMM,13	M.LOC M_READD	6.2 7.2
2D	Rename File	H.VOMM,14	M.RENAM M_RENAME	6.2 7.2
2E	Convert Pathname to Pathname Block	H.VOMM,15	M.PNAMB M_PNAMB	6.2 7.2
2F	Reconstruct Pathname	H.VOMM,16	M.PNAM M_CONSTRUCTPATH	6.2 7.2
30	Replace Permanent File	H.VOMM,23	M.REPLAC M_REPLACE	6.2 7.2
31	Modify Descriptor User Area	H.VOMM,26	M.MODU M_MODU	6.2 7.2

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
32	Rewrite Descriptor User Area	H.VOMM,27	M.REWRTU M_REWRTU	6.2 7.2
33	DBX Interface to H.PTRAC	N/A	N/A	N/A
34	Reserved for H.PTRAC			
35-3F	Reserved			
40	Include Memory Partition	H.REMM,12	M.INCLUDE	6.2
	Include Shared Image		M _INCLUDE	7.2
41	Exclude Memory Partition	H.REMM,14	M.EXCLUDE	6.2
	Exclude Shared Image		M _EXCLUDE	7.2
42	Open Resource	H.REMM,21	M.OPENR M_OPENR	6.2 7.2
43	Close Resource	H.REMM,22	M.CLOSER M_CLOSER	6.2 7.2
44	Set Exclusive Resource Lock	H.REMM,23	M.LOCK M_LOCK	6.2 7.2
45	Release Exclusive Resource Lock	H.REMM,24	M.UNLOCK M_UNLOCK	6.2 7.2
46	Set Synchronous Resource Lock	H.REMM,25	M.SETSYNC M_SETSYNC	6.2 7.2
47	Release Synchronous Resource Lock	H.REMM,26	M.UNSYNC M_UNSYNC	6.2 7.2
48	Resource Inquiry	H.REMM,27	M.INQUIRY M_INQUIRER	6.2 7.2
49	Mount Volume	H.REMM,17	M.MOUNT M_MOUNT	6.2 7.2
4A	Dismount Volume	H.REMM,19	M.DMOUNT M_DISMOUNT	6.2 7.2
4B	Get Memory in Byte Increments	H.REMM,28	M.MEMB M_GETMEMBYTES	6.2 7.2

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
4C	Free Memory in Byte Increments	H.REMM,29	M.MEMFRE M_FREEMEMBYTES	6.2 7.2
4D-4E	Reserved			
4F	Reserved			
50	Acquire Current Date/Time in ASCII Format	H.REXS,74	M.QATIM M_QATIM	6.2 7.2
	Acquire Current Date/Time in Binary Format	H.REXS,74	M.BTIM M_BTIM	6.2 7.2
	Acquire Current Date/Time in Byte Binary Format	H.REXS,74	M.BBTIM M_BBTIM	6.2 7.2
	Acquire System Date/Time in Any Format	H.REXS,74	M.GTIM M_GTIM	6.2 7.2
	Get Current Date and Time	H.REXS,74	M_GETTIME	7.2
51	Convert ASCII Date/Time to Byte Binary Format	H.REXS,75	M.CONABB M_CONABB	6.2 7.2
	Convert ASCII Date/Time to Standard Binary	H.REXS,75	M.CONASB M_CONASB	6.2 7.2
	Convert Binary Date/Time to ASCII Format	H.REXS,75	M.CONBAF M_CONBAF	6.2 7.2
	Convert Binary Date/Time to Byte Binary	H.REXS,75	M.CONBBY M_CONBBY	6.2 7.2
	Convert Byte Binary Date/Time to ASCII	H.REXS,75	M.CONBBA M_CONBBA	6.2 7.2
	Convert Byte Binary Date/Time to Binary	H.REXS,75	M.CONBYB M_CONBYB	6.2 7.2
	Convert System Date/Time Format	H.REXS,75	M.CTIM M_CTIM	6.2 7.2
	Convert Time	H.REXS,75	M_CONVERTTIME	7.2
52	Assign and Allocate Resource	H.REXS,21	M.ASSN M_ASSIGN	6.2 7.2

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
53	Deassign and Deallocate Resource	H.REXS,22	M.DASN M_DEASSIGN	6.2 7.2
54	Reformat RRS Entry	H.REXS,76	M.NEWRRS	6.2
55	Batch Job Entry	H.REXS,27	M.BATCH M_BATCH	6.2 7.2
56	Reserved for Interactive Debugger	H.REXS,30	N/A	N/A
57	Reinstate Privilege Mode to Privilege Task	H.REXS,78	M.PRIV M_PRIVMODE	6.2 7.2
58	Change Task to Unprivileged Mode	H.REXS,79	M.UPRIV M_UNPRIVMODE	6.2 7.2
59	Validate Address Range	H.REXS,33	M.VADDR M_VADDR	6.2 7.2
5A	Reserved			
5B	Set IPU Bias	H.REXS,82	M.IPUBS M_IPUBS	6.2 7.2
5C	Set Exception Handler	H.REXS,83	M_SETEXA	7.2
5D	Get Base Mode Task Address Limits	H.REXS,84	M_LIMITS	7.2
5E	Get Task Environment	H.REXS,85	M.ENVRMT M_ENVRMT	6.2 7.2
5F	Exit With Status	H.REXS,86	M_EXTSTS	7.2
60	Reserved			
61	Get Command Line	H.REXS,88	M.CMD M_CMD	6.2 7.2
62	Move Data to User Address	H.REXS,89	M.MOVE M_MOVE	6.2 7.2
63-6F	Reserved			
70	Get User Context	H.EXEC,41	M_GETCTX	7.2
71	Put User Context	H.EXEC,42	M_PUTCTX	7.2
72-74	Reserved for Symbolic Debugger/X32			
75	Reserved for MPX-32			

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
76	Allocate Shadow Memory	H.SHAD	N/A	N/A
77	Set Option Lower	H.TSM,13	M.SOPL M_SOPL	6.2 7.2
78	Reset Option Lower	H.TSM,14	M.ROPL M_ROPL	6.2 7.2
79	Set Exception Return Address	H.REXS,81	M_SETERA	7.2
7A	Get Terminal Function Definition	H.TSM,15	M.GETDEF M_GETDEF	6.2 7.2
7B	Get Address Limits	H.REXS,80	M.GADRL2	6.2
7C	Get Dynamic Extended Discontiguous Data Space	H.MEMM,9	M.GDD	6.2
7D	Get TSA Start Address	H.REXS,91	M.GTSAD M_GTSAD	6.2 7.2
7E	Physical Memory Read	H.REXS,93	M.OSREAD M_OSREAD	6.2 7.2
7F	Get Extended Memory Array	H.MEMM,14	N/A	6.3
	Get Extended Memory Array - Base Mode	H.MEMM,14	N/A	7.3
80-9F	Reserved for ACX-32			
A0-A3	Reserved for Swapper			
A4-A9	Reserved for Ada			
AA-AD	Reserved for Rapid File Allocation:		N/A	N/A
	Zero MDT	H.MDT,1		
	Locate/Read MDT Entry	H.MDT,2		
	Update/Create MDT Entry	H.MDT,3		
	Delete MDT Entry	H.MDT,4		
AE	TSM Procedure Call	H.TSM,17	M.TSMPC M_TSMPC	6.2 7.2
AF	Physical Memory Write	H.REXS,94	M.OSWRIT M_OSWRIT	6.2 7.2
B0-BE	Reserved for RMSS			
BF	Reserved			

SVC Listing

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
C0	Task Option Doubleword Inquiry	H.REXS,95	M.PGOD M_OPTIONDWORD	6.2 7.2
C1-C7	Reserved			
N/A	Allocate File Space	H.VOMM,19	N/A	6.3
N/A	Allocate Resource Descriptor	H.VOMM,17	N/A	6.3
N/A	Create File Control Block	N/A N/A	M.DFCB M_CREATEFCB	5.9.1 7.2
N/A	Create Temporary File	H.VOMM,24	N/A	6.3
N/A	Deallocate File Space	H.VOMM,20	N/A	6.3
N/A	Deallocate Resource Descriptor	H.VOMM,18	N/A	6.3
N/A	Execute Channel Program File Control Block	N/A	M_CHANPROGFCB	7.2
N/A	Read/Write Authorization File	H.VOMM,25	N/A	6.3

SVC 5,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
00	Suspend/Resume	N/A	M.SURE M_SURE	6.2 7.2

C MPX-32 Abort and Crash Codes

C.1 AC – Accounting

AC01	INSUFFICIENT SLO SPACE FOR ACCOUNTING LISTING

C.2 AD – Address Specification Trap Handler (H.IP0C)

AD01	ADDRESS SPECIFICATION ERROR OCCURRED WITHIN THE OPERATING SYSTEM
AD02	ADDRESS SPECIFICATION ERROR OCCURRED WITHIN THE CURRENT TASK
AD03	TRAP OCCURRED WHILE NO TASKS WERE IN ACTIVE STATE
AD04	TRAP OCCURRED WITHIN ANOTHER INTERRUPT TRAP ROUTINE

C.3 AL – Allocator (H.ALOC) (Compatibility Mode Only)

AL01-AL06	Reserved
AL07	THE COMBINED FILE ASSIGNMENTS FOR A TASK EXCEEDS NUMBER SPECIFIED. THE CATALOGED ASSIGNMENTS ARE COMBINED WITH THOSE DEFINED BY $ASSIGN STATEMENTS. SEE CATALOGER FILES DIRECTIVE AND RECATALOG IF NEEDED.
AL08	AN ASSIGNED PERMANENT FILE IS NONEXISTENT
AL09	AN ASSIGNED DEVICE IS NOT CONFIGURED IN THE SYSTEM. AN ASSIGNED DEVICE IS OFF-LINE.
AL10-AL11	Reserved
AL12	UNABLE TO LOAD PROGRAM BECAUSE OF I/O ERROR OR ADDRESSING INCONSISTENCIES IN LOAD MODULE PREAMBLE
AL13	AN UNRECOVERABLE I/O ERROR HAS OCCURRED DURING THE READ OF THE TASK PREAMBLE INTO THE TSA
AL14	Reserved
AL15	AN ASSIGNED DEVICE TYPE IS NOT CONFIGURED IN THE SYSTEM
AL16	A RESIDENT REQUEST HAS BEEN ISSUED FOR A TASK REQUIRING AN SLO, SBO, SGO OR SYC FILE. RESIDENT TASKS CANNOT USE SYSTEM FILES.
AL17-AL18	Reserved

AL19	A FILE CODE TO FILE CODE ASSIGNMENT (ASSIGN4) HAS BEEN MADE TO AN UNDEFINED FILE CODE. A FILE CODE MUST BE DEFINED BEFORE A SECOND FILE CODE CAN BE EQUATED BY AN ASSIGN4.
AL20	USER ATTEMPTED DEALLOCATION OF TSA
AL21	DESTROYED TASK MIDL WAS DETECTED WHILE ATTEMPTING TO ALLOCATE DYNAMIC EXECUTION SPACE
AL22	A SOFTWARE CHECKSUM ERROR HAS OCCURRED DURING TASK LOADING
AL23	AN INVALID USER NAME IS CATALOGED WITH THE TASK. THE USER NAME IS NOT CONTAINED IN THE M.KEY FILE OR A VALID KEY IS NOT SPECIFIED.
AL24	ACCESS TO AN ASSIGNED PERMANENT FILE IS BY PASSWORD ONLY, AND A VALID PASSWORD WAS NOT INCLUDED ON THE CATALOGED ASSIGNMENT OR JOB CONTROL STATEMENT ASSIGNMENT
AL25	UNDEFINED RESOURCE REQUIREMENT SUMMARY (RRS) TYPE (INTERNAL FORMAT OF AN ASSIGNMENT STATEMENT IS WRONG)
AL26	THE TASK HAS REQUESTED MORE BLOCKING BUFFERS THAN WERE SPECIFIED DURING CATALOG. SEE CATALOGER BUFFER DIRECTIVE AND RECATALOG IF NEEDED.
AL27	THERE ARE NO FREE ENTRIES IN SHARED MEMORY TABLE FOR GLOBAL, DATAPOOL, CSECT, OR OTHER SHARED AREAS
AL28	TASK IS ATTEMPTING TO SHARE AN UNDEFINED GLOBAL OR DATAPOOL MEMORY PARTITION
AL29	TASK IS ATTEMPTING TO EXCLUDE UNDEFINED MEMORY PARTITION
AL30	THE REQUESTED DEVICE IS ALREADY ASSIGNED TO THE REQUESTING TASK VIA ANOTHER FILE CODE. USE ASSIGN4 OR DEALLOCATE BEFORE REALLOCATING.
AL31	LOGICAL FILE CODE ALREADY ALLOCATED BY CALLER (E.G., A CARD READER MAY BE ASSIGNED TO LFC 'IN' AND A MAGNETIC TAPE CANNOT BE ASSIGNED TO THE SAME FILE CODE). USE ASSIGN4 OR DEALLOCATE BEFORE REALLOCATING.
AL32	DYNAMIC COMMON BLOCK MAY NOT BE ASSIGNED VIA ASSIGN1 DIRECTIVE
AL33	SHARED MEMORY DEFINITION CONFLICTS WITH CALLER'S ADDRESS SPACE
AL34	SHARED MEMORY PARTITION NOT DEFINED IN DIRECTORY
AL35	ATTEMPT TO SHARE A DIRECTORY ENTRY THAT IS NOT A MEMORY PARTITION

AL36	INVALID PASSWORD SPECIFIED FOR SHARED MEMORY PARTITION
AL37	ATTEMPT TO EXCLUDE UNDEFINED SHARED MEMORY PARTITION
AL38	ATTEMPT TO ACTIVATE A PRIVILEGED TASK BY UNAUTHORIZED OWNER
AL39	SHARED MEMORY ENTRY NOT FOUND
AL40	PARTITION DEFINITION NOT FOUND IN DIRECTORY
AL41	DIRECTORY DEFINITION NOT A DYNAMIC PARTITION
AL42	INVALID PASSWORD FOR A MEMORY PARTITION
AL43	TASK HAS ATTEMPTED TO ALLOCATE AN UNSHARED RESOURCE THAT WAS NOT AVAILABLE DURING TASK ACTIVATION IN A MEMORY-ONLY ENVIRONMENT
AL44	UNABLE TO RESUME 'SYSBUILD' TASK DURING INITIAL TASK ACTIVATION IN A MEMORY-ONLY ENVIRONMENT
AL45	UNABLE TO DEALLOCATE INPUT DEVICE AFTER DYNAMIC TASK ACTIVATION IN A MEMORY-ONLY ENVIRONMENT
AL46	TASK HAS ATTEMPTED TO SHARE MEMORY VIA A DYNAMIC MEMORY PARTITION IN A MEMORY-ONLY ENVIRONMENT
AL47	DYNAMIC MEMORY PARTITIONS CANNOT BE GREATER THAN 1 MEGABYTE
AL48	THE USER HAS ATTEMPTED TO EXCLUDE A SHARED PARTITION WHOSE ASSOCIATED MAP BLOCKS ARE NOT DESIGNATED AS BEING SHARED IN THE TASK'S TSA
AL49	THE TASK'S DSECT SPACE REQUIREMENTS OVERLAP THE TASK'S TSA SPACE REQUIREMENTS
AL50	THE TASK'S DSECT SPACE REQUIREMENTS OVERLAP THE TASK'S CSECT SPACE REQUIREMENTS, OR IF NO CSECT, LOAD MODULE IS TOO LARGE TO FIT IN USER'S ADDRESS SPACE
AL51	DESTROYED TASK MIDL DETECTED WHILE ATTEMPTING TO ALLOCATE SYSTEM BUFFER SPACE
AL52	AN ERROR CONDITION PERTAINING TO FILE SYSTEM STRUCTURES HAS OCCURRED. THIS ERROR IS NOT A FUNCTION OF THE COMPATIBILITY INTERFACE.
AL53	DESTROYED TASK MIDL WAS DETECTED WHILE ATTEMPTING TO ALLOCATE EXTENDED INDEXED DATA SPACE
AL54	INVALID COMPATIBLE RRS TYPE
AL55	ACCESS MODE IS NOT ALLOWED

C.4 AT – ANSI Labeled Tapes

AT01	INCORRECT OR NO RUN PARAMETERS RECEIVED
AT02	INCORRECT STATUS RETURNED FROM J.ATAPE RUN REQUEST
AT03	AN ERROR OCCURRED
AT04	I/O ERROR OCCURRED ON TAPE

C.5 AU – Auto-Start Trap Processor

AU01	TRAP OCCURRED ON AUTO-START

C.6 BT – Block Mode Timeout Trap

BT01	BLOCK MODE TIMEOUT TRAP

C.7 CM – Call Monitor Interrupt Processor (H.IP27 and H.IP0A)

CM01	CALL MONITOR INTERRUPT PROCESSOR CANNOT LOCATE THE 'CALM' INSTRUCTION
CM02	EXPECTED 'CALM' INSTRUCTION DOES NOT HAVE CALM (X'30') OPCODE
CM03	INVALID 'CALM' NUMBER
CM04	'CALM' NUMBER TOO LOW (OUT OF BOUNDS)
CM05	'CALM' NUMBER TOO BIG (OUT OF BOUNDS)

C.8 CP – Cache

CP01	CACHE PARITY ERROR OCCURRED WITHIN THE OPERATING SYSTEM
CP02	CACHE PARITY ERROR OCCURRED IN TASK BODY
CP03	TRAP OCCURRED WHILE NO TASKS WERE IN ACTIVE STATE
CP04	TRAP OCCURRED IN ANOTHER INTERRUPT TRAP ROUTINE

C.9 EX – Exit/Abort

EX01	AN ABORT HAS OCCURRED IN THE TASK EXIT SEQUENCE
EX02	AN ABORT HAS OCCURRED DURING THE TASK ABORT SEQUENCE AND HAS BEEN CHANGED TO A DELETE (KILL) TASK SEQUENCE
EX03	USER ATTEMPTED TO GO TO AN ANY WAIT STATE FROM AN END-ACTION ROUTINE

C.10 FS – File System (H.MONS)(Compatibility Mode Only)

FS01	UNRECOVERABLE I/O ERROR TO THE DIRECTORY
FS02	UNRECOVERABLE I/O ERROR TO FILE SPACE ALLOCATION MAP
FS03	ATTEMPT TO ADD A NEW FILE, BUT THE DIRECTORY IS FULL
FS04	A DISC ALLOCATION MAP CHECKSUM ERROR WAS DETECTED
FS05	ATTEMPT TO ALLOCATE DISC SPACE THAT IS ALREADY ALLOCATED
FS06	ATTEMPT TO DEALLOCATE DISC SPACE THAT IS NOT ALLOCATED
FS07	USER HAS CALLED AN ENTRY POINT IN H.FISE THAT NO LONGER EXISTS

C.11 HE – Online Help Facility

HE01	ABNORMAL TERMINATION WHILE TRANSLATING HELP FILES (HELPT)

C.12 HT – Halt Trap Processor (H.IPHT)

HT01	AN ATTEMPT WAS MADE TO EXECUTE A HALT INSTRUCTION IN USER'S PROGRAM
HT02	AN ATTEMPT WAS MADE TO EXECUTE A HALT INSTRUCTION IN AN INTERRUPT TRAP ROUTINE
HT03	AN ATTEMPT WAS MADE TO EXECUTE A HALT INSTRUCTION WHEN NO TASKS WERE IN AN ACTIVE STATE
HT04	Reserved
HT05	AN ATTEMPT WAS MADE TO EXECUTE A HALT INSTRUCTION WHEN USER WAS UNMAPPED

C.13 IO – Input/Output Control Supervisor (H.IOCS)

IO01	Reserved
IO02	AN UNPRIVILEGED TASK IS ATTEMPTING TO READ OR WRITE DATA INTO AN UNMAPPED ADDRESS
IO03	AN UNPRIVILEGED TASK IS ATTEMPTING TO READ DATA INTO PROTECTED MEMORY
IO04-IO05	Reserved
IO06	INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.
IO07	THE TASK HAS ATTEMPTED TO PERFORM AN OPERATION WHICH IS NOT VALID FOR THE DEVICE TO WHICH THE USER'S FILE IS ASSIGNED(E.G., A READ OPERATION SPECIFIED FOR A FILE ASSIGNED TO THE LINE PRINTER).
IO08	DEVICE ASSIGNMENT IS REQUIRED FOR AN UNPRIVILEGED TASK TO USE THIS SERVICE
IO09	ILLEGAL OPERATION ON THE SYC FILE
IO10-IO14	Reserved
IO15	A TASK HAS REQUESTED A TYPE OPERATION AND THE TYPE CONTROL PARAMETER BLOCK(TCPB) SPECIFIED INDICATES THAT AN OPERATION ASSOCIATED WITH THAT TCPB IS ALREADY IN PROGRESS
IO16	INVALID BLOCKING BUFFER CONTROL CELL(S) ENCOUNTERED DURING WRITE OF BLOCKED FILE. THIS ERROR IS USUALLY CAUSED BY A USER SPECIFIED BLOCKING BUFFER THAT HAS BEEN DESTROYED.
IO17	OPEN ATTEMPTED ON A FILE AND FPT HAS NO MATCHING FILE CODE. PROBABLE CAUSE: (1) BAD OR MISSING RRS IN PREAMBLE (2) LFC IN FCB HAS BEEN DESTROYED.
IO18	Reserved
IO19	AN ERROR HAS OCCURRED IN THE REMM CLOSE PROCEDURE
IO20	AN ERROR HAS OCCURRED IN THE REMM OPEN PROCEDURE
IO21	IOCS HAS ENCOUNTERED AN UNRECOVERABLE I/O ERROR IN ATTEMPTING TO PROCESS AN I/O REQUEST ON BEHALF OF A TASK
IO22	AN ILLEGAL IOCS ENTRY POINT HAS BEEN ENTERED BY A TASK
IO23	A H.VOMM DENIAL HAS OCCURRED IN READING THE RESOURCE DESCRIPTOR TO GET MORE SEGMENT DEFINITIONS

IO24	ILLEGAL ADDRESS, TRANSFER COUNT OR TRANSFER TYPE (I.E., IMPROPER BOUNDING FOR DATA TYPE) SPECIFIED IN THE FCB
IO25-IO27	Reserved
IO28	ILLEGAL OPERATION ATTEMPTED ON AN OUTPUT ACTIVE FILE OR DEVICE
IO29	Reserved
IO30	ILLEGAL OR UNEXPECTED VOLUME NUMBER OR REEL ID ENCOUNTERED ON MAGNETIC TAPE
IO31	Reserved
IO32	CALLING TASK HAS ATTEMPTED TO PERFORM A SECOND READ ON A '$' STATEMENT THROUGH THE SYC FILE
IO33	READ WITH BYTE GRANULARITY REQUEST MADE WITH NEGATIVE BYTE OFFSET
IO34	READ WITH BYTE GRANULARITY REQUEST MADE WITHOUT SETTING RANDOM ACCESS BIT IN FCB
IO35	READ WITH BYTE GRANULARITY REQUESTS ARE VALID FOR UNBLOCKED FILES ONLY
IO36-IO37	Reserved
IO38	WRITE ATTEMPTED ON UNIT OPENED IN READ-ONLY MODE. A READ-WRITE OPEN WILL BE FORCED TO READ-ONLY IF TASK HAS ONLY READ ACCESS TO UNIT.
IO39	Reserved
IO40	INVALID TRANSFER COUNT. TRANSFER COUNT TOO LARGE FOR TRANSFER TYPE, TRANSFER COUNT NOT AN EVEN MULTIPLE OF TRANSFER TYPE, OR DATA ADDRESS NOT BOUNDED FOR TRANSFER TYPE.
IO41	BLOCKING ERROR DURING NON-DEVICE ACCESS
IO42	BLOCKED DATA MANAGEMENT MODULE (H.BKDM) IS NOT CONFIGURED IN THE SYSTEM
IO43	INPUT/OUTPUT CONTROL LIST (IOCL) OR DATA ADDRESS NOT IN CONTIGUOUS 'E' MEMORY (GPMC DEVICES ONLY)
IO44	NON-DEVICE ACCESS I/O ERROR. THIS ERROR MAY BE THE RESULT OF CHANNEL/CONTROLLER INITIALIZATION FAILURE.
IO45	MULTIVOLUME MAGNETIC TAPE MODULE (H.MVMT) IS NOT CONFIGURED IN THE SYSTEM
IO46	Reserved
IO47	CLASS 'E' DEVICE TCW IS NOT IN CLASS 'E' MEMORY. THIS TYPE OF ERROR INDICATES A MAP FAILURE.

IO48-IO49	Reserved
IO50	AN UNPRIVILEGED USER ATTEMPTED TO EXECUTE A PHYSICAL CHANNEL PROGRAM
IO51	A 'TESTSTAR' COMMAND WAS USED IN A LOGICAL CHANNEL PROGRAM
IO52	A LOGICAL CHANNEL WAS TOO LARGE TO BE MOVED TO MEMORY POOL
IO53	A 'TIC' COMMAND FOLLOWS A 'TIC' COMMAND IN A LOGICAL CHANNEL PROGRAM
IO54	A 'TIC' COMMAND ATTEMPTED TO TRANSFER TO AN ADDRESS WHICH IS NOT WORD BOUNDED
IO55	ILLEGAL ADDRESS IN LOGICAL IOCL. ADDRESS IS NOT IN USER'S LOGICAL ADDRESS SPACE.
IO56	A READ-BACKWARD COMMAND WAS USED IN A LOGICAL CHANNEL PROGRAM
IO57	ILLEGAL IOCL ADDRESS. IOCL MUST BE LOCATED IN THE FIRST 128K WORDS OF MEMORY.
IO58-IO60	Reserved
IO61	INVALID LFC IN FCB
IO62	ERROR OCCURRED ON IMPLICIT OPEN
IO63-IO76	Reserved
IO77	ATTEMPT TO USE DATA FLOW CONTROL (OTHER THAN WISM), THAT IS NOT SUPPORTED BY THE CURRENTLY INSTALLED CONTROLLER
IO78	ATTEMPT TO ISSUE AN EXECUTE CHANNEL PROGRAM TO A WRITE SUB-CHANNEL AND THE SUB-CHANNEL WAS NOT IN DUAL CHANNEL MODE
IO79	Reserved
IO80	ILLEGAL ACCESS MODE FOR VOLUME RESOURCE
IO81-IO97	Reserved
IO98	H.VOMM DENIAL HAS OCCURRED ON IOCS AUTOMATIC FILE EXTENSION REQUEST FOR THE LFC SPECIFIED IN THE ABORT MESSAGE
IO99	INTERNAL SYSTEM ERROR DETECTED AT THE ADDRESS RELATIVE TO IOCS WHICH IS SPECIFIED IN THE ABORT MESSAGE

C.14 IP – IPU

IP01	ABNORMAL TASK TERMINATION IN IPU

C.15 LD – Task Activation Loading (H.TAMM)

LD01	LOAD CODE SECTION ERROR
LD02	CODE SECTION CHECKSUM ERROR
LD03	BIAS CODE ERROR
LD04	CODE MATRIX CHECKSUM ERROR
LD05	LOAD DATA SECTION ERROR
LD06	DATA SECTION CHECKSUM ERROR
LD07	BIAS DATA ERROR
LD08	DATA MATRIX CHECKSUM ERROR
LD09	GCF R/O RELOCATION ERROR
LD10	GCF R/W RELOCATION ERROR

C.16 MC – Machine Check Trap

MC01	MACHINE CHECK TRAP

C.17 MF – Map Fault Trap

MF01	A MAP FAULT TRAP HAS OCCURRED. THIS IS THE RESULT OF A BAD MEMORY REFERENCE OUTSIDE OF THE USER'S ADDRESSABLE SPACE.

C.18 MM – Memory Disk

MM01	REQUEST FOR MEMORY DISC I/O TO A LOCATION OUTSIDE THE MEMORY DISC BOUNDARIES

C.19 MP – Memory Parity Trap (H.IP02)

MP01	MEMORY ERROR OCCURRED IN A TASK'S LOGICAL ADDRESS SPACE. THIS IS AN INTERNAL OR CPU FAILURE. RERUN TASK.
MP02	MEMORY ERROR OCCURRED IN ANOTHER INTERRUPT TRAP ROUTINE (NESTED TRAPS, CONTEXT LOST)
MP03	MEMORY ERROR OCCURRED WHILE NO TASKS WERE IN THE ACTIVE STATE
MP04	MEMORY ERROR OCCURRED IN A MAP BLOCK RESERVED FOR THE O/S
MP05	ERROR OCCURRED WHILE CURRENT TASK WAS IN THE UNMAPPED MODE

C.20 MS – System Services (H.MONS) (Compatibility Mode Only)

MS01	PERMANENT FILE ADDRESS INQUIRY SERVICE FOUND A NUMBER OF ALLOCATION UNITS IN THE UNIT DEFINITION TABLE THAT DO NOT CORRESPOND TO ANY KNOWN DISC.
MS02-MS08	Reserved
MS09	TASK HAS ATTEMPTED TO CONNECT A TASK TO AN INTERRUPT LEVEL NOT DEFINED FOR INDIRECTLY CONNECTED TASKS
MS10-MS11	Reserved
MS12	OVERLAY IS PASSWORD PROTECTED
MS13-MS15	Reserved
MS16	TASK HAS REQUESTED DYNAMIC ALLOCATION WITH AN INVALID FUNCTION CODE
MS17	FILE NAME CONTAINS CHARACTERS OUTSIDE RANGE OF X'20' TO X'5F', INCLUSIVELY
MS18-MS20	Reserved
MS21	MULTIVOLUME MAGNETIC TAPE ALLOCATION REQUEST MADE TO SCRATCH (SCRA) TAPE
MS22	MULTI-VOLUME MAGNETIC TAPE ALLOCATION REQUEST MADE ON SHARED TAPE DRIVE
MS23	TASK HAS ISSUED A 'MOUNT MESSAGE ONLY' ALLOCATION REQUEST TO A NON-ALLOCATED DRIVE OR TO A DEVICE WHICH IS NOT A MAGNETIC TAPE

MS24	TASK HAS SPECIFIED AN ILLEGAL VOLUME NUMBER (ZERO IF TAPE IS MULTIVOLUME, NONZERO IF TAPE IS SINGLE VOLUME)
MS25-MS27	Reserved
MS28	A PERMANENT FILE LOG HAS BEEN REQUESTED, BUT THE ADDRESS SPECIFIED FOR STORAGE OF THE DIRECTORY ENTRY IS NOT CONTAINED WITHIN THE CALLING TASK'S LOGICAL ADDRESS SPACE
MS29	Reserved
MS30	TASK HAS ATTEMPTED TO OBTAIN A PERMANENT FILE LOG IN A MEMORY-ONLY ENVIRONMENT
MS31	USER ATTEMPTED TO GO TO THE ANY-WAIT STATE FROM AN END-ACTION ROUTINE
MS32	Reserved
MS33	ALLOCATION ERROR IN RTM M.ALOC CALL
MS34-MS86	Reserved
MS87	NO DENIAL RETURN ADDRESS SPECIFIED ON CALM M.ALOC EMULATION

C.21 NM – Nonpresent Memory Trap

NM01	A NONPRESENT MEMORY TRAP ERROR CONDITION HAS OCCURRED.

C.22 OC – Operator Communications

OC01	THE OPERATOR HAS REQUESTED THAT THE TASK BE ABORTED

C.23 PT – Task Activation (J.TSM)

PT01	INVALID ATTEMPT TO MULTICOPY A UNIQUE TASK
PT02	FILE SPECIFIED IS NOT IN DIRECTORY
PT03	UNABLE TO ALLOCATE FILE
PT04	FILE IS NOT A VALID LOAD MODULE OR EXECUTABLE IMAGE
PT05	DQE IS NOT AVAILABLE
PT06	READ ERROR ON RESOURCE DESCRIPTOR

PT07	READ ERROR ON LOAD MODULE
PT08	INSUFFICIENT LOGICAL/PHYSICAL ADDRESS SPACE FOR TASK ACTIVATION
PT09	CALLING TASK IS UNPRIVILEGED
PT10	INVALID PRIORITY
PT11	INVALID SEND BUFFER ADDRESS OR SIZE
PT12	INVALID RETURN BUFFER ADDRESS OR SIZE
PT13	INVALID NO-WAIT MODE END ACTION ROUTINE ADDRESS
PT14	MEMORY POOL UNAVAILABLE
PT15	DESTINATION TASK RECEIVER QUEUE FULL
PT16	INVALID PSB ADDRESS
PT17	RRS LIST EXCEEDS 384 WORDS
PT18	INVALID RRS ENTRY IN PARAMETER BLOCK

C.24 PV – Privilege Violation Trap

PV01	PRIVILEGE VIOLATION TRAP

C.25 RC – Record Manager

RC01	LESS THAN ONE BLOCK ON READ
RC02	NOT A MULTIPLE NUMBER OF BLOCKS READ
RC03	NO MORE IOC'S AVAILABLE
RC04	ERROR CONDITION ON READ
RC05	PREMATURE END-OF-FILE
RC06	END-OF-MEDIUM ON OUTPUT FILE
RC07	WRITE ATTEMPTED ON UNOPENED FILE
RC08	USER RECORD SIZE TOO LARGE
RC09	READ NOT ALLOWED AFTER WRITE
RC10	ERROR ON WRITE
RC11	END-OF-MEDIUM ON OUTPUT FILE

RC12	INTERNAL FILE POSITION ERROR
RC13	RESOURCE CANNOT BE OPENED
RC14	INTERNAL FILE POSITION ERROR
RC15	INVALID BLOCKING BUFFER CELL

C.26 RE – Restart

RE01	RESTART IS INVALID IN BATCH OR COMMAND FILE MODE

C.27 RF – Rapid File Allocation

RF01	INVALID PATHNAME
RF02	PATHNAME CONSISTS OF VOLUME ONLY
RF03	VOLUME NOT MOUNTED
RF04	Reserved
RF05	FILE IS NOT A PERMANENT FILE
RF06	Reserved
RF07	RESOURCE DOES NOT EXIST
RF08	RESOURCE NAME IN USE
RF09	Reserved
RF10	MDT ENTRY UNAVAILABLE
RF11-RF14	Reserved
RF15	VOLUME MUST BE MOUNTED PUBLIC
RF16-RF59	Reserved
RF60	INVALID MODE
RF61-RF98	Reserved
RF99	WARNING, INPUT ERRORS ENCOUNTERED, CHECK SLO OUTPUT

C.28 RM – Resource Management (H.REMM)

RM01	UNABLE TO LOCATE RESOURCE
RM02	ACCESS MODE NOT ALLOWED

RM – Resource Management (H.REMM)

RM03	TOO MANY ASSIGNMENTS
RM04	BLOCKING BUFFER SPACE NOT AVAILABLE OR INVALID BUFFER ADDRESS
RM05	SHARED MEMORY TABLE (SMT) ENTRY NOT FOUND
RM06	TOO MANY MOUNT REQUESTS
RM07	STATIC ASSIGN TO DYNAMIC COMMON
RM08	UNRECOVERABLE I/O ERROR
RM09	INVALID USAGE SPECIFICATION
RM10	INVALID PARAMETER ADDRESS
RM11	INVALID RESOURCE REQUIREMENT SUMMARY (RRS) ENTRY
RM12	INVALID LFC TO LFC ASSIGNMENT
RM13	DEVICE NOT IN SYSTEM OR OFF-LINE
RM14	RESOURCE ALREADY ALLOCATED BY TASK
RM15	INVALID SYC/SGO ASSIGNMENT
RM16	COMMON CONFLICTS WITH TASK ADDRESS SPACE
RM17	DUPLICATE LFC ASSIGNMENT
RM18	INVALID DEVICE SPECIFICATION
RM19	INVALID RESOURCE ID (RID)
RM20	VOLUME UNASSIGNED OR ACCESS NOT ALLOWED
RM21	UNABLE TO MOUNT. J.MOUNT RUN REQUEST FAILED
RM22	RESOURCE MARKED FOR DELETION
RM23	ASSIGNED DEVICE IS MARKED OFF-LINE
RM24	UNABLE TO LOCATE MOUNTED VOLUME TABLE(MVT) ENTRY
RM25	RANDOM ACCESS NOT ALLOWED
RM26	ATTEMPT TO WRITE ON SYC
RM27	RESOURCE ALREADY OPENED IN DIFFERENT MODE
RM28	INVALID ACCESS SPECIFICATION AT OPEN
RM29	INVALID FILE CONTROL BLOCK(FCB) ADDRESS OR UNASSIGNED LFC IN FCB
RM30	INVALID ALLOCATION INDEX

RM31	RESOURCE NOT OPEN
RM32	LOCK NOT OWNED BY THIS TASK
RM33	RESOURCE IS NOT ALLOCATED IN A SHARABLE MODE
RM34	SYSTEM ADMINISTRATOR ATTRIBUTE IS REQUIRED TO MOUNT A PUBLIC VOLUME
RM35	RESOURCE IS NOT A SHARED IMAGE
RM36	PHYSICAL MEMORY ALREADY ALLOCATED
RM37	ATTEMPT TO ALLOCATE NONPRESENT PHYSICAL MEMORY
RM38	TIME OUT WAITING FOR RESOURCE
RM39	UNABLE TO PERFORM WRITE BACK
RM40	INVALID LOAD MODULE
RM41	INVALID PHYSICAL ADDRESS SPECIFIED
RM42	USER REQUESTED ABORT OF MOUNT PROCESS
RM43	USER REQUESTED HOLD ON MOUNT PROCESS
RM44	WRITEBACK REQUESTED AND SHARED IMAGE HAS NO WRITEBACK SECTION
RM45	LOADING ERROR DURING INCLUSION OF READ ONLY SECTION OF SHARED IMAGE
RM46	UNABLE TO OBTAIN RESOURCE DESCRIPTOR LOCK (MULTIPORT ONLY)
RM47	LOADING ERROR DURING INCLUSION OF READ/WRITE SECTION OF SHARED IMAGE
RM48	INCOMPATIBLE LOAD ADDRESSES FOR SHARED IMAGE
RM49	TASK HAS REQUESTED EXCESSIVE NUMBER OF MULTICOPIED SHARED IMAGES WITH NO READ ONLY SECTION
RM50	RESOURCE IS LOCKED BY ANOTHER TASK
RM51	SHAREABLE RESOURCE IS ALLOCATED BY ANOTHER TASK IN AN INCOMPATIBLE ACCESS MODE
RM52	VOLUME SPACE IS NOT AVAILABLE
RM53	ASSIGNED DEVICE IS NOT AVAILABLE
RM54	UNABLE TO ALLOCATE RESOURCE FOR SPECIFIED USAGE
RM55	ALLOCATED RESOURCE TABLE (ART) SPACE IS NOT AVAILABLE

RM56	TASK REQUIRES SHADOW MEMORY AND NONE IS CONFIGURED
RM57	VOLUME IS NOT AVAILABLE FOR MOUNT WITH REQUESTED USAGE
RM58	SHARED MEMORY TABLE (SMT) SPACE IS NOT AVAILABLE
RM59	MOUNTED VOLUME TABLE (MVT) SPACE IS NOT AVAILABLE
RM60	RESOURCE DESCRIPTOR SPACE DEFINITION CONFLICT
RM61	UNABLE TO LOCATE OR RETRIEVE RESOURCE DESCRIPTOR
RM62	INVALID OPTION IN CNP
RM63	SEGMENTED TASK SUPPORT NOT PRESENT.
RM64	THE TASK'S DSECT SPACE REQUIREMENTS OVERLAP THE TASK'S TASK SERVICE AREA(TSA) SPACE REQUIREMENTS
RM65	THE TASK'S DSECT SPACE REQUIREMENTS OVERLAP THE TASK'S CSECT SPACE REQUIREMENTS, OR IF NO CSECT, LOAD MODULE IS TOO LARGE TO FIT IN USER'S ADDRESS SPACE
RM66	SOFTWARE CHECKSUM. ERROR MAY BE FIXED BY RECATALOGING.
RM67	EXCESSIVE MEMORY REQUEST
RM68	EXCESSIVE VOLUME SPACE REQUESTED
RM69	INVALID USERNAME SPECIFIED
RM70	INVALID PRIVILEGED ACTIVATION
RM71	Reserved
RM72	UNABLE TO RESUME SYSINIT ON TAPE ACTIVATION
RM73	FILE OVERLAP HAS OCCURRED. PLEASE CHECK THE SYSTEM CONSOLE
RM74	LOADING ERROR
RM75	INVALID WORK VOLUME/DIRECTORY
RM76	USER ATTEMPTED DEALLOCATION OF TSA
RM77	A TASK HAS DESTROYED THE ALLOCATION LINKAGES IN ITS DYNAMIC EXPANSION SPACE
RM78	UNABLE TO LOAD TASK DEBUGGER WITH TASK
RM79	INVALID CALLER NOTIFICATION PACKET (CNP) ADDRESS
RM80	SHARED IMAGE VERSION LEVEL IS NOT COMPATIBLE WITH EXECUTABLE IMAGE

RM81	INVALID ACTIVATION OF A BASE MODE TASK ON A SYSTEM CONFIGURED FOR NON-BASE TASK EXECUTION.
RM82	INVALID ACTIVATION OF AN ADA TASK ON A SYSTEM CONFIGURED WITHOUT ADA SUPPORT.
RM83	INSUFFICIENT LOGICAL ADDRESS SPACE TO ACTIVATE TASK
RM84	INVALID LOGICAL POSITION FOR EXTENDED MPX
RM85	PTRACE DEBUG REQUESTED AND H.PTRAC NOT CONFIGURED
RM86	CANNOT DISMOUNT THE SYSTEM VOLUME.
RM87	PUBLIC VOLUME DISMOUNT DENIED DUE TO COMPATIBLE MODE PUBLIC DISMOUNT OPTION SET FOR THIS SYSTEM.
RM88	PUBLIC DISMOUNT DENIED. SYSTEM ADMINISTRATOR ATTRIBUTE REQUIRED FOR THIS OPERATION.
RM89	PUBLIC DISMOUNT DENIED DUE TO MISSING OPTION FOR PUBLIC VOLUME IN THE DISMOUNT REQUEST
RM90	GCL LOADMODULE OR SHIM CANNOT BE RELOCATABLE
RM91	UNABLE TO ACCESS VOLUME DUE TO PENDING PHYSICAL DISMOUNT.
RM92	READ ONLY OR READ WRITE LOAD ADDRESS IS INVALID
RM93	UNABLE TO PERFORM PHYSICAL MOUNT DUE TO SYSTEM SHUTDOWN IN PROGRESS.
RM94	J.MOUNT ATTEMPTED TO MOUNT AN UNFORMATTED DISC VOLUME.
RM95	AN UNBIASED TASK REQUIRES SHADOW MEMORY ON A SYSTEM WITH NO OVERLAPPING CPU/IPU SHADOW REGION
RM96	A BIASED TASK REQUIRES SHADOW MEMORY THAT DOES NOT EXIST ON THE SPECIFIED PROCESSOR
RM97	Reserved
RM98	THE TASK REQUIRES MORE SHADOW MEMORY THAN EXISTS

C.29 RX – Resident Executive Services (H.REXS)

RX01	Reserved
RX02	INVALID FUNCTION CODE SPECIFIED FOR REQUEST TO CREATE A TIMER ENTRY. VALID CODES ARE ACP (1), RSP OR RST (2), STB (3), RSB (4) AND RQI (5).
RX03	TASK ATTEMPTED TO SET/RESET A BIT OUTSIDE OF A STATIC PARTITION OR THE OPERATING SYSTEM.

RX04	THE REQUESTING TASK IS UNPRIVILEGED OR HAS ATTEMPTED TO CREATE A TIMER ENTRY TO REQUEST AN INTERRUPT WITH A PRIORITY LEVEL OUTSIDE THE RANGE OF X'12' TO X'7F', INCLUSIVELY
RX05	INVALID FUNCTION CODE HAS BEEN SPECIFIED FOR REQUEST TO SET USER STATUS WORD
RX06	UNPRIVILEGED TASK ATTEMPTED TO RESET A TASK PRIORITY LEVEL, OR A PRIVILEGED TASK ATTEMPTED TO RESET A TASK PRIORITY TO A LEVEL OUTSIDE THE RANGE OF 1 TO 64, INCLUSIVELY
RX07	CANNOT LOAD OVERLAY SEGMENT DUE TO SOFTWARE CHECKSUM OR DATA ERROR
RX08	OVERLAY IS NOT IN THE DIRECTORY
RX09	Reserved
RX10	OVERLAY HAS AN INVALID PREAMBLE
RX11	AN UNRECOVERABLE I/O ERROR HAS OCCURRED DURING OVERLAY LOADING
RX12	Reserved
RX13	FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE
RX14	DESTINATION BUFFER ADDRESS IS INVALID OR PROTECTED
RX15	ATTEMPT TO SET EXCEPTION RETURN ADDRESS WHEN ARITHMETIC EXCEPTION NOT IN PROGRESS
RX16-RX24	Reserved
RX25	OPERATOR HAS ABORTED TASK IN RESPONSE TO MOUNT MESSAGE
RX26-RX28	Reserved
RX29	TASK HAS ATTEMPTED TO LOAD THE INTERACTIVE TASK DEBUGGER OVERLAY IN A MEMORY-ONLY ENVIRONMENT
RX30-RX31	Reserved
RX32	INVALID DQE ADDRESS
RX33	OVERLAY LINKAGES HAVE BEEN DESTROYED BY LOADING A LARGER OVERLAY
RX34	TASK HAS MADE A BREAK RECEIVER EXIT CALL WHILE NO BREAK IS ACTIVE
RX35	Reserved

RX36	STATUS IN REGISTER ZERO IS NOT A ZERO OR A VALID ABORT CODE
RX37-RX85	Reserved
RX86	TASK HAS MADE AN END ACTION ROUTINE EXIT WHILE END ACTION WAS NOT ACTIVE
RX87	Reserved
RX88	RESERVED FOR DEBUG LINK SERVICE
RX89	AN UNPRIVILEGED TASK HAS ATTEMPTED TO REESTABLISH AN ABORT RECEIVER (OTHER THAN M.IOEX)
RX90	TASK HAS MADE A RUN REQUEST END ACTION ROUTINE EXIT WHILE THE RUN REQUEST INTERRUPT WAS NOT ACTIVE
RX91	TASK HAS ATTEMPTED NORMAL EXIT WITH A TASK INTERRUPT STILL ACTIVE
RX92	TASK HAS ATTEMPTED NORMAL EXIT WITH MESSAGES IN ITS RECEIVER QUEUE
RX93	AN INVALID RECEIVER EXIT BLOCK (RXB) ADDRESS WAS ENCOUNTERED DURING MESSAGE EXIT
RX94	AN INVALID RECEIVER EXIT BLOCK (RXB) RETURN BUFFER ADDRESS WAS ENCOUNTERED DURING MESSAGE EXIT
RX95	TASK HAS MADE A MESSAGE EXIT WHILE THE MESSAGE INTERRUPT WAS NOT ACTIVE
RX96	AN INVALID RECEIVER EXIT BLOCK (RXB) ADDRESS WAS ENCOUNTERED DURING RUN RECEIVER EXIT
RX97	AN INVALID RECEIVER EXIT BLOCK (RXB) RETURN BUFFER ADDRESS WAS ENCOUNTERED DURING RUN RECEIVER EXIT
RX98	TASK HAS MADE A RUN RECEIVER EXIT WHILE THE RUN RECEIVER INTERRUPT WAS NOT ACTIVE
RX99	TASK HAS MADE A MESSAGE END-ACTION ROUTINE EXIT WHILE THE MESSAGE INTERRUPT WAS NOT ACTIVE

C.30 SB – System Binary Output

SB01	AN I/O ERROR HAS BEEN ENCOUNTERED ON THE DEVICE ASSIGNED AS THE SYSTEM BINARY (PUNCHED) OUTPUT DEVICE
SB02	THE SYSTEM OUTPUT PROGRAM HAS ENCOUNTERED AN UNRECOVERABLE I/O ERROR IN ATTEMPTING TO READ A PUNCHED OUTPUT FILE FROM DISC
SB03	DENIAL OF FILE CODE TO FILE CODE ALLOCATION FOR J.SOUT2 INDICATES LOSS OF SYSTEM INTEGRITY

SB – System Binary Output

SB04	SYSTEM BINARY OUTPUT ABORTED BY OPERATOR
SB05	NO TIMER ENTRY FOR SYSTEM BINARY OUTPUT (SYSTEM FAULT)
SB06	FIVE ECHO CHECK ERRORS DETECTED WHILE ATTEMPTING TO PUNCH A SINGLE CARD

C.31 SC – System Check Trap Processor

SC01	SYSTEM CHECK TRAP OCCURRED AT AN ADDRESS LOCATED WITHIN THE OPERATING SYSTEM
SC02	SYSTEM CHECK TRAP OCCURRED WITHIN THE CURRENT TASK'S SPACE
SC03	SYSTEM CHECK TRAP OCCURRED AT A TIME WHEN THERE WERE NO TASKS CURRENTLY BEING EXECUTED (C.PRNO EQUALS ZERO)
SC04	SYSTEM CHECK TRAP OCCURRED WITHIN ANOTHER TRAP (C.GINT DOES NOT EQUAL '1')

C.32 SD – SCSI Disk

SD00	NO ADDITIONAL SENSE INFORMATION
SD01	NO INDEX/SECTOR SIGNAL
SD02	NO SEEK COMPLETE
SD03	WRITE FAULT
SD04	DRIVE NOT READY
SD05	DRIVE NOT SELECTED
SD06	NO TRACK ZERO FOUND
SD07	MULTIPLE DRIVES SELECTED
SD08	LOGICAL UNIT COMMUNICATIONS FAILURE
SD09	TRACK FOLLOWING ERROR
SD10-SD15	Reserved
SD16	ID CRC OR ECC ERROR
SD17	UNRECOVERED READ ERROR OF DATA BLOCKS
SD18	NO ADDRESS MARK FOUND IN ID FIELD

SD19	NO ADDRESS MARK FOUND IN DATA FIELD
SD20	NO RECORD FOUND
SD21	SEEK POSITIONING ERROR
SD22	DATA SYNCHRONIZATION MARK ERROR
SD23	RECOVERED READ DATA WITH TARGET'S READ RETRIES (NOT WITH ECC)
SD24	RECOVERED READ DATA WITH TARGET'S ECC CORRECTION (NOT WITH RETRIES)
SD25	DEFECT LIST ERROR
SD26	PARAMETER OVERRUN
SD27	SYNCHRONOUS TRANSFER ERROR
SD28	PRIMARY DEFECT LIST NOT FOUND
SD29	COMPARE ERROR
SD30	RECOVERED ID WITH TARGET'S ECC CORRECTION
SD31	Reserved
SD32	INVALID COMMAND OPERATION CODE
SD33	ILLEGAL LOGICAL BLOCK ADDRESS. ADDRESS GREATER THAN THE LBA RETURNED BY THE READ CAPACITY DATA WITH PMI BIT NOT SET IN CDB
SD34	ILLEGAL FUNCTION FOR DEVICE TYPE
SD35	Reserved
SD36	ILLEGAL FIELD IN CDB
SD37	INVALID LUN
SD38	INVALID FIELD IN PARAMETER LIST
SD39	WRITE PROTECTED
SD40	MEDIUM CHANGE
SD41	POWER ON OR RESET OR BUS DEVICE RESET OCCURRED
SD42	MODE SELECT PARAMETERS CHANGED
SD43-SD47	Reserved
SD48	IMCOMPATIBLE CARTRIDGE

SD49	MEDIUM FORMAT CORRUPTED
SD50	NO DEFECT SPARE LOCATION AVAILABLE
SD51-SD63	Reserved
SD64	RAM FAILURE
SD65	DATA PATH DIAGNOSTIC FAILURE
SD66	POWER ON DIAGNOSTIC FAILURE
SD67	MESSAGE REJECT ERROR
SD68	INTERNAL CONTROLLER ERROR
SD69	SELECT/RESELECT FAILED
SD70	UNSUCCESSFUL SOFT RESET
SD71	SCSI INTERFACE PARITY ERROR
SD72	INITIATOR DETECTED ERROR
SD73	INAPPROPRIATE/ILLEGAL MESSAGE

C.33 SG – System Generator (SYSGEN)

SG01	INVALID LOADER FUNCTION CODE IN BINARY OBJECT MODULE FROM THE SYSTEM RESIDENT MODULE (OBJ) FILE
SG02	INVALID BINARY RECORD READ FROM SYSTEM RESIDENT MODULE (OBJ) FILE (BYTE 0 MUST BE X'FF' OR X'DF')
SG03	SEQUENCE ERROR IN MODULE BEING READ FROM TEMPORARY FILE
SG04	CHECKSUM ERROR IN MODULE BEING READ FROM TEMPORARY FILE
SG05	UNABLE TO FIND CDT AND/OR UDT FOR I/O MODULE LOAD
SG06	UNABLE TO OBTAIN ADDITIONAL MEMORY REQUIRED FOR RESIDENT SYSTEM IMAGE MODULE LOADING
SG07	UNABLE TO OBTAIN MEMORY REQUIRED FOR RESIDENT SYSTEM IMAGE CONSTRUCTION
SG08	NON-RELOCATABLE BYTE STRING ENCOUNTERED IN BINARY MODULE BEING PROCESSED FROM TEMPORARY FILE
SG09	UNABLE TO ALLOCATE TEMPORARY FILE SPACE
SG10	OVERRUN OF SYSGEN ADDRESS SPACE BY SYSTEM BEING GENERATED. PROBABLE ERRONEOUS SIZE SPECIFICATION IN PATCH OR POOL DIRECTIVE.

SG11	SEQUENCE ERROR WHILE READING OBJECT MODULE FROM FILE ASSIGNED TO 'OBJ'
SG12	CHECKSUM ERROR WHILE READING OBJECT MODULE FROM FILE ASSIGNED TO 'OBJ'
SG13	UNABLE TO ALLOCATE DISC SPACE FOR SYMTAB FILE. POSSIBLE CAUSES ARE INSUFFICIENT DISC SPACE OR ACCESS RIGHTS DENIAL.
SG14	UNABLE TO ALLOCATE DISC SPACE FOR SYSTEM IMAGE FILE. POSSIBLE CAUSES ARE INSUFFICIENT DISC SPACE, ACCESS RIGHTS DENIAL, OR ATTEMPTING TO SYSGEN OVER CURRENT DEFAULT IMAGE.
SG15	MAXIMUM NUMBER (240) OF SYMBOL TABLE/PATCH FILE ENTRIES EXCEEDED
SG16	MISSING SYSTEM OR SYMTAB DIRECTIVE
SG17	INVALID IPU INTERVAL TIMER PRIORITY. MUST NOT BE BETWEEN X'78' AND X'7F'.
SG18	MAXIMUM SIZE OF 88K FOR TARGET SYSTEM HAS BEEN EXCEEDED
SG19	ATTEMPT TO DEFINE INTERRUPT VECTORING ROUTINE AS SYSTEM REENTRANT. ONLY DEVICE HANDLERS MAY BE SYSTEM REENTRANT.
SG20	UNABLE TO FIND "LINK" DEVICE IN UDT
SG21	INSUFFICIENT ROOM IN MEMORY POOL FOR DOWNLOAD FILE LIST
SG22	Reserved
SG23	SHARE DIRECTIVE SPECIFIED WITHOUT ENOUGH SMT ENTRIES. ENTRIES MUST EXCEED OR BE EQUAL TO THE NUMBER OF PARTITIONS PLUS MEMORY DISCS.
SG24	ATTEMPT TO DEFINE PARTITION STARTING MAPBLOCK NUMBER IN OPERATING SYSTEM AREA
SG25	ATTEMPT TO DEFINE PARTITION STARTING MAPBLOCK NUMBER IN NON-CONFIGURED PHYSICAL MEMORY
SG26	ATTEMPT TO USE A MODULE INCOMPATIBLE WITH THE TARGET MACHINE TYPE. THE OFFENDING MODULE NAME IS THE LAST ENTRY ON THE LISTING FOLLOWED BY THREE ASTERISKS (***).
SG27	THE DEVICE SPECIFIED IN EITHER THE SWAPDEV, SID, LOD OR POD DIRECTIVE IS NOT INCLUDED IN THE CONFIGURATION BEING BUILT
SG28	THE NULL DEVICE SPECIFICATION WHICH IS REQUIRED TO BE INCLUDED IN EVERY CONFIGURATION IS MISSING

SG29	SYSINIT OBJECT MODULE MISSING ON SYSGEN OBJECT INPUT FILE (OBJ). IT MUST BE THE LAST MODULE.
SG30	THE FILE ASSIGNED TO FILE CODE OBJ DOES NOT CONTAIN VALID OBJECT CODE
SG31	THE GENERATED IMAGE CONTAINS UNSATISFIED EXTERNAL REFERENCES. SEE THE SLO OUTPUT FOR MORE DETAILS. THIS IS NOT A FATAL ABORT AND THE SYSTEM IMAGE IS PRODUCED.
SG32	ONE OR MORE REQUESTED OBJECT MODULES COULD NOT BE LOCATED ON THE INPUT OBJECT FILE. SEE THE SLO OUTPUT FOR MORE DETAILS. THIS IS NOT A FATAL ABORT AND THE SYSTEM IMAGE IS PRODUCED.
SG33	EVENT TRACE HAS BEEN ENABLED WITH NO MEMORY PARTITION RESERVED FROM X'78000' TO X'80000'
SG34	Reserved
SG35	INSUFFICIENT MEMORY POOL FOR STATIC PARTITION
SG36	UNMAPPED DEBUG MODULE (H.DBUG2) IS MISSING ON SYSGEN OBJECT INPUT FILE. IT MUST BE THE LAST MODULE IF THE SYSTEM DEBUGGER IS TO BE CONFIGURED.
SG37	COMMUNICATION REGION + DSECT + ADAPTIVE REGION EXCEEDS 16KW
SG38	MPX EXTENDED CODE AREA EXTENDS PAST LOGICAL LIMIT
SG39	INVALID MPX EXTENDED CODE AREA LOGICAL MAP START
SG40	DIRECTIVE ERRORS ENCOUNTERED. IMAGE PRODUCED.
SG41	H.IPPF COULD NOT BE LOCATED ON THE INPUT OBJECT FILE. MODULE IS NECESSARY FOR DEMAND PAGE.
SG42-SG97	Reserved
SG98	ERROR ENCOUNTERED DURING OBJECT PROCESSING PRECEDED BY MESSAGE DESCRIBING THE ERROR CONDITION
SG99	DIRECTIVE ERRORS ENCOUNTERED

C.34 SH – Shadow Memory (J.SHAD)

SH01	J.SHAD ABORTED. SEE OUTPUT (UT IF INTERACTIVE OR SLO IF BATCH), FOR ACTUAL ERROR DESCRIPTION(S).

C.35 SN – System Input Task (J.SSIN)

SN00	INVALID RUN REQUEST PARAMETERS

C.36 SS – Sort/Merge (FSORT2)

SS01	CTL NOT ALLOCATED
SS02	HEADER DIRECTIVE MISSING
SS03	CONTROL FILE EMPTY
SS04	DIRECTIVE CODE NOT VALID
SS05-SS06	Reserved
SS07	OUTPUT FILE CODE (OUT) NOT ALLOCATED
SS08	RECORD LENGTH NOT DIVISIBLE INTO INPUT PHYSICAL RECORD LENGTH
SS09	RECORD LENGTH EXCEEDS INPUT PHYSICAL RECORD LENGTH
SS10	INPUT RECORD LENGTH EXCEEDS MAXIMUM ALLOWED (4095)
SS11	RECORD LENGTH NOT DIVISIBLE INTO OUTPUT PHYSICAL RECORD LENGTH
SS12	RECORD LENGTH EXCEEDS OUTPUT BLOCK LENGTH
SS13	OUTPUT PHYSICAL RECORD LENGTH EXCEEDS MAXIMUM ALLOWED (4095)
SS14	..1 PRESENT BUT NOT A DISC FILE
SS15	..2 PRESENT BUT NOT A DISC FILE
SS16	COMPARISON INDICATOR NOT VALID
SS17	Reserved
SS18	WK1 HAS BEEN ALLOCATED BY THE USER
SS19	WK2 HAS BEEN ALLOCATED BY THE USER
SS20	FIELD DIRECTIVE ERROR: STARTING POSITION IS GREATER THAN FIELD ENDING POSITION
SS21	FIELD DIRECTIVE ERROR: STARTING POSITION EXCEEDS RECORD LENGTH
SS22	FIELD DIRECTIVE ERROR: ENDING POSITION EXCEEDS LOGICAL RECORD LENGTH
SS23-SS27	Reserved
SS28	INAPPROPRIATE COMBINATION OF TOURNAMENT PARAMETERS EXCEEDS MEMORY POOL LIMITS
SS29	DISC SPACE CANNOT BE ALLOCATED FOR WORK FILE 1

SS30	DISC SPACE CANNOT BE ALLOCATED FOR WORK FILE 2
SS31	FILE TO FILE ALLOCATION FOR WORKFILE HAS FAILED
SS32	SORT BUFFER TOO SMALL
SS33-SS39	Reserved
SS40	INPUT FILES ARE EMPTY: NO RECORD INPUT OR SORTED
SS41	WK1 OR WK2 FILES TOO SMALL
SS42	MERGE ONLY SELECTED BUT NO MERGE FILES (MG1-MG8) ARE ASSIGNED
SS43-SS47	Reserved
SS48	SORT ATTEMPTED WITHOUT GOOD CALL TO SORT:HDR
SS49-SS57	Reserved
SS58	INAPPROPRIATE COMBINATION OF BUFFER PARAMETERS DETECTED DURING OUTPUT PHASE
SS59	END OF MEDIUM DETECTED ON THE OUT FILE
SS60-SS68	Reserved
SS69	COMPARE TABLE TYPE DESTROYED: SORT PROBLEM
SS70-SS97	Reserved
SS98	ERROR OPENING FILE LO
SS99	ERROR OPENING FILE OUT

C.37 ST – System Output Task (J.SOUT)

ST01	UNRECOVERABLE WRITE ERROR TO DESTINATION DEVICE
ST02	UNABLE TO PERFORM ALLOCATION OF SEPARATOR FILE CODE
ST03	UNABLE TO ISSUE MAGNETIC TAPE MOUNT MESSAGE VIA ALLOCATION SERVICE

Whenever a system output task aborts, the task may be restarted with the OPCOM REPRINT or REPUNCH commands.

C.38 SV – SVC Trap Processor (H.IP06)

SV01	UNPRIVILEGED TASK ATTEMPTING TO USE M.CALL
SV02	INVALID SVC NUMBER

SV03	UNPRIVILEGED TASK ATTEMPTING TO USE A 'PRIVILEGED-ONLY' SERVICE
SV04	INVALID SVC TYPE
SV05	UNPRIVILEGED TASK ATTEMPTING TO USE M.RTRN
SV06	INVALID MODULE NUMBER OR ENTRY POINT
SV07	ATTEMPTING TO USE A SVC WHICH IS INVALID FOR BASE REGISTER OPERATIONS
SV08	SVC 0, 1 OR 2 ATTEMPTED THAT WOULD RESULT IN A TSA STACK OVERFLOW (I.E. T.REGP GREATER THAN T.LASTP)
SV09	ATTEMPT TO USE A COMPATIBLE MODE SERVICE WITH NOCMS SPECIFIED IN SYSGEN

C.39 SW – Swap Scheduler Task (J.SWAPR)

SW01	I/O ERROR ON INSWAP OR OUTSWAP
SW02	EOM DETECTED ON SWAP FILE
SW03	CAN NOT CREATE SWAP FILE SPACE DIRECTORY IN MEMORY POOL
SW04	SWAP FILE SPACE DIRECTORY IS FULL
SW05	TASK HAS REQUESTED INSWAP BUT WAS NEVER OUTSWAPPED

C.40 SX – System Output Executive (J.SOEX)

SX01	INVALID RUN REQUEST HEADCELL COUNT
SX02	LOAD MODULE J.SOUT DOES NOT EXIST

C.41 SY – System Initialization (SYSINIT)

SY01	SYSTEM HALT OCCURRED DURING SYSINIT PHASE ONE PROCESSING
SY02	SYSTEM HALT DUE TO MEMORY PARITY ERROR BEING DETECTED IN THE OPERATING SYSTEM

C.42 TD – Terminal Type Set/Reset Utility (J.TSET)

TD01	ATTEMPTED TO RUN J.TSET IN BATCH MODE
TD02	J.TSET WAS UNABLE TO OPEN UT FOR PROCESSING

C.43 TS – Terminal Support

TS01	USER REQUESTED REMOVAL FROM A BREAK REQUEST
TS02	USER REQUESTED REMOVAL FROM A RESOURCE WAIT STATE QUEUE
TS03	TASK RUNNING FROM SPECIFIED TERMINAL WAS ABORTED WHEN THE TERMINAL DISCONNECTED
TS04	REMOVAL OF A JOB WAS REQUESTED

C.44 UI – Undefined Instruction Trap

UI01	UNDEFINED INSTRUCTION TRAP
UI02	UNEXPECTED DEBUGX32 BREAKPOINT FOUND AND DEBUGX32 NOT ATTACHED

C.45 VF – Volume Formatter (J.VFMT)

VF01	ERROR HAS OCCURRED. SEE SLO FILE FOR EXPLANATION.
VF02	OPEN FAILURE ON AUDIT TRAIL DEVICE/FILE
VF03	EOF/EOM ON AUDIT TRAIL DEVICE/FILE
VF04	I/O ERROR ON AUDIT TRAIL DEVICE/FILE

C.46 VM – Volume Management Module (H.VOMM)

In some cases, H.VOMM displays H.REMM abort conditions. If a user calls an H.VOMM service which in turn calls an H.REMM service for processing and an abort condition occurs within the H.REMM processing, the abort condition is returned to H.VOMM which displays it to the user in the format `10`*xx* where *xx* is the specific H.REMM abort condition. For example, abort condition `1026` indicates H.REMM error 26 has occurred. The TSM $ERR command can be used to determine the reason for the error, i.e., `$ERR RM26`.

VM01	INVALID PATHNAME
VM02	PATHNAME CONSISTS OF VOLUME ONLY

VM03	VOLUME NOT MOUNTED
VM04	DIRECTORY DOES NOT EXIST
VM05	DIRECTORY NAME IN USE
VM06	DIRECTORY CREATION NOT ALLOWED AT SPECIFIED LEVEL
VM07	RESOURCE DOES NOT EXIST
VM08	RESOURCE ALREADY EXISTS
VM09	RESOURCE DESCRIPTOR UNAVAILABLE
VM10	DIRECTORY ENTRY UNAVAILABLE
VM11	REQUIRED FILE SPACE UNAVAILABLE
VM12	UNRECOVERABLE I/O ERROR READING DMAP
VM13	UNRECOVERABLE I/O ERROR WRITING DMAP
VM14	UNRECOVERABLE I/O ERROR READING RESOURCE DESCRIPTOR
VM15	UNRECOVERABLE I/O ERROR WRITING RESOURCE DESCRIPTOR
VM16	UNRECOVERABLE I/O ERROR READING SMAP
VM17	UNRECOVERABLE I/O ERROR WRITING SMAP
VM18	UNRECOVERABLE I/O ERROR READING DIRECTORY
VM19	UNRECOVERABLE I/O ERROR WRITING DIRECTORY
VM20	PROJECTGROUP NAME OR KEY INVALID
VM21	Reserved
VM22	INVALID FILE CONTROL BLOCK(FCB) OR LFC
VM23	PARAMETER ADDRESS SPECIFICATION ERROR
VM24	RESOURCE DESCRIPTOR NOT CURRENTLY ALLOCATED
VM25	PATHNAME BLOCK OVERFLOW
VM26	FILE SPACE NOT CURRENTLY ALLOCATED
VM27	'CHANGE DEFAULTS' NOT ALLOWED
VM28	RESOURCE CANNOT BE ACCESSED IN REQUESTED MODE OR DEFAULT SYSTEM IMAGE FILE CANNOT BE DELETED
VM29	OPERATION NOT ALLOWED ON THIS RESOURCE TYPE (RESOURCE IS NOT CORRECT TYPE)
VM30	REQUIRED PARAMETER WAS NOT SPECIFIED

VM31	FILE EXTENSION DENIED. SEGMENT DEFINITION AREA FULL.
VM32	FILE EXTENSION DENIED. FILE WOULD EXCEED MAXIMUM SIZE ALLOWED.
VM33	I/O ERROR OCCURRED WHEN RESOURCE WAS ZEROED
VM34	REPLACEMENT FILE CANNOT BE ALLOCATED
VM35	INVALID DIRECTORY ENTRY
VM36	DIRECTORY AND FILE ARE NOT ON THE SAME VOLUME
VM37	AN UNIMPLEMENTED ENTRY POINT HAS BEEN CALLED
VM38	REPLACEMENT FILE IS ALLOCATED BY ANOTHER TASK AND BIT 0 IN THE CNP OPTION FIELD IS NOT SET, OR FILE IS ALLOCATED BY OTHER CPU IN MULTI-PORT ENVIRONMENT
VM39	OUT OF SYSTEM SPACE
VM40	CANNOT ALLOCATE FAT/FPT WHEN CREATING A TEMPORARY FILE
VM41	DEALLOCATE ERROR IN ZEROING FILE
VM42	RESOURCE DESCRIPTOR DESTROYED OR THE RESOURCE DESCRIPTOR AND THE DIRECTORY ENTRY LINKAGE HAS BEEN DESTROYED
VM43	INVALID RESOURCE SPECIFICATION
VM44	INTERNAL LOGIC ERROR FROM RESOURCE MANAGEMENT MODULE (H.REMM). ABORT TASK, TRY A DIFFERENT TASK AND IF IT FAILS, REBOOT SYSTEM.
VM45	ATTEMPTED TO MODIFY MORE THAN ONE RESOURCE DESCRIPTOR AT THE SAME TIME OR ATTEMPTED TO REWRITE A RESOURCE DESCRIPTOR PRIOR TO MODIFYING IT
VM46	RESOURCE DESCRIPTOR IS LOCKED BY ANOTHER CPU (MULTI-PORT ONLY)
VM47	DIRECTORY CONTAINS ACTIVE ENTRIES AND CANNOT BE DELETED
VM48	A RESOURCE DESCRIPTOR'S LINK COUNT IS ZERO
VM49	ATTEMPTING TO DELETE A PERMANENT RESOURCE WITHOUT SPECIFYING A PATHNAME OR PATHNAME BLOCK VECTOR
VM50	RESOURCE DESCRIPTOR CONTAINS UNEXPECTED RESOURCE DESCRIPTOR TYPE
VM51	DIRECTORY ENTRY DELETED BUT FAILED TO RELEASE FILE SPACE

VM52	AN ATTEMPT WAS MADE TO DEALLOCATE FREE SPACE OR TO ALLOCATE SPACE THAT IS CURRENTLY ALLOCATED ON A VOLUME OTHER THAN SYSTEM DISC
VM53	THE FILE SPACE CREATED IS LESS THAN THE SPACE REQUESTED
VM54-VM98	Reserved
VM99	AN ATTEMPT WAS MADE TO DEALLOCATE FREE SPACE OR TO ALLOCATE SPACE THAT IS CURRENTLY ALLOCATED ON THE SYSTEM VOLUME

C.47 VO – Volume Manager (VOLMGR)

VO01	ERROR HAS OCCURRED. SEE SLO FILE FOR EXPLANATION.
VO02	OPEN FAILURE ON AUDIT TRAIL DEVICE/FILE
VO03	EOF/EOM ON AUDIT TRAIL DEVICE/FILE
VO04	I/O ERROR ON AUDIT TRAIL DEVICE/FILE
VO05	Reserved
VO06	I/O ERROR ON THE TAPE DURING SAVE OPERATION. TAPE HAS BEEN BACKSPACED TO THE END OF THE LAST SAVED FILE. ALL FILES ON THE IMAGE PRIOR TO THE TAPE I/O ERROR ARE SAVED ON THE TAPE.

C.48 Crash Codes

When system crash occurs as a result of a trap handler entry, the CPU halts with the registers containing the following information:

Register	Contents	
0	PSD Word 0 (when trap generated)	
1	PSD Word 1 (when trap generated)	
2	Real address of instruction causing trap	
3	Instruction causing trap	
4	CPU status word (from trap handler)	
5	Crash code:	
	MP01=X'4D503031'	(See H.IP02 Codes)
	NM01=X'4E4D3031'	(Nonpresent Memory - H.IP03)
	UI01=X'55493031'	(Undefined Instruction - H.IP04)
	PV01=X'50563031'	(Privilege Violation - H.IP05)
	MC01=X'4D433031'	(Machine Check - H.IP07)
	SC01=X'53433031'	(System Check - H.IP08)
	MF01=X'4D463031'	(Map Fault - H.IP09)
	CP01=X'43503031'	(Cache Parity Error - H.IP10) 32/67, 32/87 and 32/97
	SW01=X'53573031'	(See SWAPR codes)
6	Real address of register save block	
7	C'TRAP'=X'54524150'	

For further description, see Volume I, Chapter 2.

D Numerical Information

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5

87D13C01

Numerical Information

2^n	n	2^{-n}
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 380 086 994 171 142 578 125

87D13D01

E Powers of Integers

E.1 Powers of Sixteen in Decimal

16^n	n	16^{-n}
1	0	0.10000 00000 00000 00000 x 10
16	1	0.62500 00000 00000 00000 x 10^{-1}
256	2	0.39062 50000 00000 00000 x 10^{-2}
4 096	3	0.24414 06250 00000 00000 x 10^{-3}
65 536	4	0.15258 78906 25000 00000 x 10^{-4}
1 048 576	5	0.95367 43164 06250 00000 x 10^{-6}
16 777 216	6	0.59604 64477 53906 25000 x 10^{-7}
268 435 456	7	0.37252 90298 46191 40625 x 10^{-8}
4 294 967 296	8	0.23283 06436 53869 62891 x 10^{-9}
68 719 476 736	9	0.14551 91522 83668 51807 x 10^{-10}
1 099 511 627 776	10	0.90949 47017 72928 23792 x 10^{-11}
17 592 186 044 416	11	0.56843 41886 08080 14870 x 10^{-13}
281 474 976 710 656	12	0.35527 13678 80050 09294 x 10^{-14}
4 503 599 627 370 496	13	0.22204 46049 25031 30808 x 10^{-15}
72 057 594 037 927 936	14	0.13877 78780 78144 56755 x 10^{-16}
1 152 921 504 606 846 976	15	0.86736 17379 88403 54721 x 10^{-18}

E.2 Powers of Ten in Hexadecimal

10^n				n	10^{-n}					
			1	0	1.0000	0000	0000	0000		
			A	1	0.1999	9999	9999	999A		
			64	2	0.28F5	C28F	5C28	F5C3	x	16^{-1}
			3E8	3	0.4189	374B	C6A7	EF9E	x	16^{-2}
			2710	4	0.68DB	8BAC	710C	B296	x	16^{-3}
		1	86A0	5	0.A7C5	AC47	1B47	8423	x	16^{-4}
		F	4240	6	0.10C6	F7A0	B5ED	8D37	x	16^{-4}
		98	9680	7	0.1AD7	F29A	BCAF	4858	x	16^{-5}
		5F5	E100	8	0.2AF3	1DC4	6118	73BF	x	16^{-6}
		3B9A	CA00	9	0.44B8	2FA0	9B5A	52CC	x	16^{-7}
	2	540B	E400	10	0.6DF3	7F67	5EF6	EADF	x	16^{-8}
	17	4876	E800	11	0.AFEB	FF0B	CB24	AAFF	x	16^{-9}
	E8	D4A5	1000	12	0.1197	9981	2DEA	1119	x	16^{-9}
	918	4E72	A000	13	0.1C25	C268	4976	81C2	x	16^{-10}
	5AF3	107A	4000	14	0.2D09	370D	4257	3604	x	16^{-11}
3	8D7E	A4C6	8000	15	0.480E	BE7B	9D58	566D	x	16^{-12}
23	86F2	6FC1	0000	16	0.734A	CA5F	6226	F0AE	x	16^{-13}
163	4578	5D8A	0000	17	0.B877	AA3	36A4	B449	x	16^{-14}
DF0	B6B3	A764	0000	18	0.1272	5DD1	D243	ABA1	x	16^{-14}
8AC7	2304	89E8	0000	19	0.1D83	C94F	B6D2	AC35	x	16^{-15}

F ASCII Interchange Code Set

Row	Col	0	1	2	3	4	5	6	7
Bit Positions 4 5 6 7	0 1 2 3	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
0000	0	NUL 12-0-9-8-1	DLE 12-11-9-8-1	SP No Punch	0 0	@ 8-4	P 11-7	‘ 8-1	p 12-11-7
0001	1	SOH 12-9-1	DC1 11-9-1	! 12-8-7	1 1	A 12-1	Q 11-8	a 12-0-1	q 12-11-8
0010	2	STX 12-9-2	DC2 11-9-2	" 8-7	2 2	B 12-2	R 11-9	b 12-0-2	r 12-11-9
0011	3	ETX 12-9-3	DC3 11-9-3	# 8-3	3 3	C 12-3	S 0-2	c 12-0-3	s 11-0-2
0100	4	EOT 9-7	DC4 9-8-4	$ 11-8-3	4 4	D 12-4	T 0-3	d 12-0-4	t 11-0-3
0101	5	ENQ 0-9-8-5	NAK 9-8-5	% 0-8-4	5 5	E 12-5	U 0-4	e 12-0-5	u 11-0-4
0110	6	ACK 0-9-8-6	SYN 9-2	& 12	6 6	F 12-6	V 0-5	f 12-0-6	v 11-0-5
0111	7	BEL 0-9-8-7	ETB 0-9-6	' 8-5	7 7	G 12-7	W 0-6	g 12-0-7	w 11-0-6
1000	8	BS 11-9-6	CAN 11-9-8	(12-8-5	8 8	H 12-8	X 0-7	h 12-0-8	x 11-0-7
1001	9	HT 12-9-5	EM 11-9-8-1	) 11-8-5	9 9	I 12-9	Y 0-8	i 12-0-9	y 11-0-8
1010	A	LF 0-9-5	SUB 9-8-7	* 11-8-4	: 8-2	J 11-1	Z 0-9	j 12-11-1	z 11-0-9
1011	B	VT 12-9-8-3	ESC 0-9-7	+ 12-8-6	; 11-8-6	K 11-2	[12-8-2	k 12-11-2	{ 12-0
1100	C	FF 12-9-8-4	FS 11-9-8-4	0-8-3	< 12-8-4	L 11-3	\ 0-8-2	l 12-11-3	¦ 12-11
1101	D	CR 12-9-8-5	GS 11-9-8-5	- 11	= 8-6	M 11-4	] 11-8-5	m 12-11-4	} 11-0
1110	E	SO 12-9-8-6	RS 11-9-8-6	. 12-8-3	> 0-8-6	N 11-5	Λ 11-8-7	n 12-11-5	~ 11-0-1
1111	F	SI 12-9-8-7	US 11-9-8-7	/ 0-1	? 0-8-7	O 11-6	_ 0-8-5	o 12-11-6	DEL 12-9-7

87D13B01

ASCII Interchange Code Set

Some positions in the ASCII code chart may have different graphic representation on various devices as:

ASCII	IBM 029
!	|
[	¢
]	!
Λ	>

Control Characters:

NUL	-	Null
SOH	-	Start of Heading (CC)
STX	-	Start of Text (CC)
ETX	-	End of Text (CC)
EOT	-	End of Transmission (CC)
ENQ	-	Enquiry (CC)
ACK	-	Acknowledge (CC)
BEL	-	Bell (audible or attention signal)
BS	-	Backspace (FE)
HT	-	Horizontal Tabulation (punch card skip) (FE)
LF	-	Line Feed (FE)
VT	-	Vertical Tabulation (FE)
FF	-	Form Feed (FE)
CR	-	Carriage Return (FE)
SO	-	Shift Out
SI	-	Shift In
DLE	-	Data Link Escape (CC)
DC1	-	Device Control 1
DC2	-	Device Control 2
DC3	-	Device Control 3
DC4	-	Device Control 4 (stop)
NAK	-	Negative Acknowledge (CC)
SYN	-	Synchronous Idle (CC)
ETB	-	End of Transmission Block (CC)
CAN	-	Cancel
EM	-	End of Medium
SS	-	Start of Special Sequence
ESC	-	Escape
FS	-	File Separator (IS)
GS	-	Group Separator (IS)
RS	-	Record Separator (IS)
US	-	Unit Separator (IS)
DEL	-	Delete
SP	-	Space (normally nonprinting)
(CC)	-	Communication Control
(FE)	-	Format Effector
(IS)	-	Information Separator

87D13C02

G IOP/MFP Panel Mode Commands

AS	Clear address stop
AS=*xxxxxxxx*	Set address stop at address *xxxxxxxx*
BAS	Read base registers
BAS*n*=*xxxxxxxx*	Write base register *n* (0-7) with *xxxxxxxx*
CLE	Clear memory
CRMD=*xxxxxxxxxxxx*	Load CRAM with *xxxxxxxxxxxx*
=*xxxxxxxxxxxx*	Load CRAM with data and increment address
CS	Read control switches
CS=*xxxxxxxx*	Set control switches to *xxxxxxxx*
EA	Read effective address
EXEC	Execute CRAM
GPR	Read general purpose registers
GPR*n*=*xxxxxxxx*	Write general purpose register *n* (0-7) with *xxxxxxxx*
HALT	Halt
IPL	IPL from default address
IPL=*xxxx*	IPL from channel/subaddress *xxxx*
IS	Clear instruction stop
IS=*xxxxxxxx*	Set instruction stop at address *xxxxxxxx*
MA=*xxxxxx*	Read physical memory address location *xxxxxx*
<ret>	Increment and read memory address
MAV=*xxxxxx*	Read virtual memory address location *xxxxxx*
<ret>	Increment and read memory address
MD=*xxxxxxxx*	Write memory data *xxxxxxxx* into last location addressed
=*xxxxxxxx*	Increment and write memory data *xxxxxxxx*
<ret>	Increment and write previous data
MSGE	Message between primary and secondary panels (IOP only)
OVR	Toggle clock override

PC=*xxxxxx*	Load program counter with address *xxxxxx*
PRIP	Set primary panel (master; IOP only)
PSD	Read program status doubleword (1 and 2)
PSD=*xxxxxxxx*	Write program status word (2) with *xxxxxxxx*
PSW	Read program status word (1)
PSW=*xxxxxxxx*	Write program status word (1) with *xxxxxxxx*
RS	Clear read operand stop
RS=*xxxxxxxx*	Set read operand stop at address *xxxxxxxx*
RST	Reset
RUN	Run
SECP	Set secondary panel (master and slave; IOP only)
STEP	Instruction step
<ret>	Continuation of instruction step
WS	Clear write operand stop
WS=*xxxxxxxx*	Set write operand stop at address *xxxxxxxx*
@@C	Enter console mode
@@P	Enter panel mode
(LF)	Repeat command

Notes:

1. Press the return key (<ret>) after each command.
2. LOCK ON and LOCK OFF are not supported by the CRT panel.

Console Mode

To change from panel mode to console mode, enter @@C<ret>.

Upon receipt of the <ret> following the @@C command, the firmware moves the cursor on the CRT to the extreme left margin of the next line.

To return to the panel mode, enter @@P<ret>. When the panel mode is selected, // is the prompt.

H Standard Date and Time Formats

H.1 Description

With the advent of the new MPX-32 file system, proper maintenance of the system date and time becomes more important than ever before as all file system resources will be time stamped to aid in management. It is vital the date and time be kept in a manner that is at once useful in this application and also convenient to convert into other formats that the user might require.

System date and time are kept in standard binary format. This format consists of two words: the first word contains the date and the second word contains the time. The date is maintained as the number of days since January 1, 1960 and the time is maintained as the binary count of system time units since midnight, adjusted to 100 microsecond granularity.

For the convenience of the user, monitor service calls are provided to convert the date and time between any of three standard formats. These are:

1. Binary Format (described above)
2. Byte Binary Format
3. ASCII Format (sometimes referred to as quad ASCII format)

Byte binary format time consists of two words: the first word contains date information and the second word contains time information. In byte binary format, the date is kept as four distinct values instead of one. Byte 0 of the date word is the binary century, byte 1 is the binary year in that century, byte 2 is the binary month and byte 3 the binary day of the month. Time is kept in a similar manner with byte 0 being the hour, byte 1 the minute, byte 2 the second, and byte 3 the number of clock ticks.

ASCII format consists of four words of information. The first two words contain the ASCII century, year, month, and day in successive halfwords. The second two words contain the hour, minutes, seconds, and clock ticks in a similar fashion. In ASCII format, use of a 120-hertz clock can cause truncation of the clock tick fields, allowing for only two ASCII digits.

H.2 Date/Time Standard Formats

Binary

Date	Time
Days since 1/1/60	Clock ticks since midnight
Word 1	Word 2

Byte Binary

Date				Time			
Bin Cent.	Bin Year	Bin Month	Bin Day	Bin Hour	Bin Min.	Bin Sec.	Bin Ints.
Word 1				Word 2			

Quad ASCII

Date

Century	Year	Month	Day
Word 1		Word 2	

Time

Hour	Minute	Second	Interrupt
Word 3		Word 4	

I Compressed Source Format

Compressed source files are blocked files that consist of 120 byte records. The last record may be less than 120 bytes and has a data type code of 9F. The structure of a compressed record is described below.

Each record contains 6 control bytes:

1 byte	data type code, BF (9F indicates last record)
1 byte	byte count, number of data bytes in record
2 bytes	checksum, halfword sum of data bytes
2 bytes	sequence number, record sequence number starting at zero

Data is recorded as follows:

1 byte	blank count, number of blanks before data
1 byte	data count, number of data bytes
n-bytes	actual ASCII data

.
. (this sequence is repeated until the end of a line is reached)
.

1 byte	EOL character, FF

J Map Block Address Assignments

Map Block # Decimal/Hex	Page # Decimal/Hex	Address Range Hexadecimal
00/00	00/00	00000 - 01FFF
01/01	04/04	02000 - 03FFF
02/02	08/08	04000 - 05FFF
03/03	12/0C	06000 - 07FFF
04/04	16/10	08000 - 09FFF
05/05	20/14	0A000 - 0BFFF
06/06	24/18	0C000 - 0DFFF
07/07	28/1C	0E000 - 0FFFF
08/08	32/20	10000 - 11FFF
09/09	36/24	12000 - 13FFF
10/0A	40/28	14000 - 15FFF
11/0B	44/2C	16000 - 17FFF
12/0C	48/30	18000 - 19FFF
13/0D	52/34	1A000 - 1BFFF
14/0E	56/38	1C000 - 1DFFF
15/0F	60/3C	1E000 - 1FFFF
16/10	64/40	20000 - 21FFF
17/11	68/44	22000 - 23FFF
18/12	72/48	24000 - 25FFF
19/13	76/4C	26000 - 27FFF
20/14	80/50	28000 - 29FFF
21/15	84/54	2A000 - 2BFFF
22/16	88/58	2C000 - 2DFFF
23/17	92/5C	2E000 - 2FFFF
24/18	96/60	30000 - 31FFF
25/19	100/64	32000 - 33FFF
26/1A	104/68	34000 - 35FFF
27/1B	108/6C	36000 - 37FFF
28/1C	112/70	38000 - 39FFF
29/1D	116/74	3A000 - 3BFFF
30/1E	120/78	3C000 - 3DFFF
31/1F	124/7C	3E000 - 3FFFF
32/20	128/80	40000 - 41FFF
33/21	132/84	42000 - 43FFF
34/22	136/88	44000 - 45FFF
35/23	140/8C	46000 - 47FFF
36/24	144/90	48000 - 49FFF
37/25	148/94	4A000 - 4BFFF

Map Block Address Assignments

Map Block # Decimal/Hex	Page # Decimal/Hex	Address Range Hexadecimal
38/26	152/98	4C000 - 4DFFF
39/27	156/9C	4E000 - 4FFFF
40/28	160/A0	50000 - 51FFF
41/29	164/A4	52000 - 53FFF
42/2A	168/A8	54000 - 55FFF
43/2B	172/AC	56000 - 57FFF
44/2C	176/B0	58000 - 59FFF
45/2D	180/B4	5A000 - 5BFFF
46/2E	184/B8	5C000 - 5DFFF
47/2F	188/BC	5E000 - 5FFFF
48/30	192/C0	60000 - 61FFF
49/31	196/C4	62000 - 63FFF
50/32	200/C8	64000 - 65FFF
51/33	204/CC	66000 - 67FFF
52/34	208/D0	68000 - 69FFF
53/35	212/D4	6A000 - 6BFFF
54/36	216/D8	6C000 - 6DFFF
55/37	220/DC	6E000 - 6FFFF
56/38	224/E0	70000 - 71FFF
57/39	228/E4	72000 - 73FFF
58/3A	232/E8	74000 - 75FFF
59/3B	236/EC	76000 - 77FFF
60/3C	240/F0	78000 - 79FFF
61/3D	244/F4	7A000 - 7BFFF
62/3E	248/F8	7C000 - 7DFFF
63/3F	252/FC	7E000 - 7FFFF
Extended Memory 128KW to 256KW - 1B		
64/40	256/100	80000 - FFFFF
Extended Memory 256KW to 384KW - 1B		
128/80	512/200	100000 - 17FFFF
Extended Memory 384KW to 512KW - 1B		
192/C0	768/300	180000 - 1FFFFF
Extended Memory 512KW to 1024KW - 1B		
256/100	1024/400	200000 - 3FFFFF
Extended Memory 1024KW to 2048KW - 1B		
512/200	2048/800	400000 - 7FFFFF
Extended Memory 2048KW to 4096KW - 1B		
1024/400	4096/1000	800000 - FFFFFF

K Control Switches

While rebooting the system, various initialization processes can be inhibited or enabled by setting the appropriate control switches. The switch assignments are:

Switch	Function if Set
0	Inhibits volume clean-up by J.MOUNT.
1	SYSINIT enters the system debugger before processing patches.
2	Inhibits patch processing (see Reference Manual, Volume III, Chapter 9, Entry Conditions).
3	Inhibits terminal initialization.
4	Inhibits accounting functions including the M.KEY, M.PRJCT, M.ACCNT, and M.ERR files.
5	Inhibits processing of the sequential task activation table at IPL time.
6	If J.MOUNT encounters an invalid resource descriptor due to an invalid resource descriptor type field or space definition, it branches and links to the system debugger (if present) with R2 pointing to the resource descriptor.
7	J.MOUNT prereads the file space bit map (SMAP) or the resource descriptor allocation bit map (DMAP). J.MOUNT will not perform file overlap protection.
8	Delete spooled output files instead of resubmitting them for processing.
9	Inhibits activating LOADACS during IPL or RESTART operations.
10	Enables faster memory initialization by checking only one location per map block to determine if that map block is present. It is not recommended that this switch be set on the first IPL after power up.
11	Inhibits initialization of the memory descriptor table (MDT).
12	For RMSS: inhibits booting of nodes while J.BOOT executes.

The control switches can be accessed by the console. The proper time to set the switches is while the system is waiting for the date and time to be entered. To set, for example, switch 3, the following must be entered on the IOP/MFP console:

```
ENTER DATE AND TIME: @@P
//CS=10000000    Terminal Initialization Inhibited
//@@C
<CR>
INVALID DATE FORMAT=MM/DD/XX
ENTER DATE AND TIME:
```

Refer to the CONCEPT 32/2000 Operations manual for instructions for setting control switches on the Amiga console.

During power up, control switches are prezeroed if the proper firmware revision level has been installed. Power up without prezeroing can cause unexpected system responses due to incorrect control settings.

All control switch settings are preserved during system reboots not involving system power up (i.e., online restart and IPL).

L Data Structures

L.1 Introduction

This appendix contains some of the more frequently used data structures. Below is a list of those structures.

Caller Notification Packet (CNP)
Controller Definition Table (CDT)
Dispatch Queue Entry (DQE)
File Control Block (FCB), 16 Word
File Control Block (FCB), 8 Word
File Control Block (FCB), High Speed Data
File Pointer Table (FPT)
Parameter Task Activation Block (PTASK)
TSM Procedure Call Block (PCB)
Pathname Blocks (PNB)
Post Program-Controlled Interrupt Notification Packet (PPCI)
Parameter Receive Block (PRB)
Parameter Send Block (PSB)
Resource Create Block (RCB)
Resource Identifiers (RID)
Resource Logging Block (RLB)
Resource Requirement Summary (RRS) Entries
Receiver Exit Block (RXB)
Type Control Parameter Block (TCPB)
Unit Definition Table (UDT)

L.2 Caller Notification Packet (CNP)

The caller notification packet (CNP) is the mechanism used by the Resource Management Module (H.REMM) and the Volume Management Module (H.VOMM) for handling abnormal conditions that may result during resource requests. All or part of this structure can be used by a particular service being called. The CNP must be on a word boundary.

	0 7 8 15	16 23 24 31
Word 0	Time-out value (CP.TIMO)	
1	Abnormal return address (CP.ABRET)	
2	Option field (CP.OPTS). See Note 1.	Status field (CP.STAT). See Note 2.
3-4	Reserved (See Note 3.)	
5	Automatic open FCB address (CP.FCBA)	

Notes:

1. A bit sequence and/or value used to provide additional information that can be necessary to fully define the calling sequence for a particular service.
2. A right-justified numeric value identifying the return status for this call.
3. Refer to the individual system service description in the MPX-32 Reference Manual Volume I for interpretation of these words.

L.3 Controller Definition Table (CDT)

The controller definition table (CDT) is a system resident structure used to identify information required by handlers and the I/O processor for a specific controller. The CDT is built by the SYSGEN process, one for each controller configured on the system. The CDT identifies devices (UDTs) associated with the controller, the handler address associated with the controller, and defines other pertinent controller information.

	0 7	8 15	16 23	24 31
Word 0	String forward address (CDT.FIOQ)			
1	String backward address (CDT.BIOQ)			
2	Link priority (CDT.LPRI). See Note 1.	Number of entries in list (CDT.IOCT). See Note 2.	Class (CDT.CLAS). See Note 3.	Flags (CDT.FLG2). See Note 4.
3	CDT index (CDT.INDX)		Device type code (CDT.DTC) See Note 5.	Interrupt priority level (CDT.IPL)
4	Number units on controller (CDT.NUOC)	Number requests outstanding (CDT.IORO)	Channel number (CDT.CHAN)	Subaddress of first device (CDT.SUBA)
5	Program number if reserved (CDT.PNRC)	Interrupt handler address (CDT.SIHA) or controller information block (CDT.CIF)		
6	Flags (CDT.FLGS). See Note 6.	UDT address of first device on controller (CDT.UDTA)		
7	I/O status (CDT.IOST). See Note 7.	TI address (CDT.TIAD) or SI address if extended I/O (CDT.SIAD)		
8	UDT address unit 0* (CDT.UT0)			
9-23	UDT address unit 1* (CDT.UT1) through UDT address unit 15* (CDT.UTF)			

*Initialized by SYSGEN

Notes:

1. Always zero (head cell)
2. Number of entries in list (zero if none)

3. Values in CDT.CLAS are assigned as follows:

Value	Meaning
X'0D'	TCW type with extended addressing capability
X'0E'	TCW type
X'0F'	extended I/O

4. Bits in CDT.FLG2 are assigned as follows:

Bit	Meaning if Set
0	SCSI device (CDT.SCSI)
1-7	reserved for future use

5. For example, 01 for any disk, 04 for any tape, etc. Valid device type codes are listed in Appendix A.

6. Bits in CDT.FLGS are assigned as follows:

Bit	Meaning if Set
0	extended I/O device (CDT.FCLS)
1	I/O outstanding (set by handler, reset by IOCS) (CDT.IOU1)
2	GPMC device (CDT.GPMC)
3	initialization (INC) needs to be performed for this controller (CDT.FINT)
4	D-class (CDT.XGPM)
5	used only when IOQs are linked to the CDT. Set when SIO is accepted by the controller. Reset when IOQ is unlinked from the CDT or when I/O is reported complete to IOCS in the case of operator intervention type errors (CDT.IOU5).
6	IOP controller (CDT.IOP)
7	controller malfunction (CDT.MALF)

7. Bits in CDT.IOST are assigned as follows:

Bit	Meaning if Set
0	IOQ linked to UDT (CDT.NIOQ)
1	multiplexing controller (CDT.MUXC)
2	use standard XIO interface
3	16MB GPMC (CDT.XGPS)
4	cache controller (CDT.CAC)
5	H.F8XIO has determined if the controller is pre-8512-2 or not (CDT.CKFL)
6	controller not pre-8512-2 (CDT.FLOW)
7	reserved for FMS

L.4 Dispatch Queue Entry (DQE)

The dispatch queue entry (DQE) contains all of the core-resident information required to describe an active task to the system. It is always linked to the CPU scheduler state chain that describes the current execution status of the associated task.

Word No. (Decimal)	Byte (Hex)	0 – 7	8 – 15	16 – 23	24 – 31
0	0	DQE.SF			
1	4	DQE.SB			
2	8	DQE.CUP	DQE.BUP	DQE.IOP	DQE.US
3	C	DQE.NUM/DQE.TAN			
4-5	10	DQE.ON			
6-7	18	DQE.LMN			
8-9	20	DQE.PSN			
10	28	DQE.USW			
11	2C	DQE.USHF			
12	30	DQE.MSD			
13	34	DQE.KCTR			
14	38	DQE.MMSG	DQE.MRUN	DQE.MNWI	DQE.GQFN
15	3C	DQE.UF2	DQE.IPUF	DQE.NWIO	DQE.SOPO
16	40	DQE.CQC			
17	44	DQE.SH	DQE.SHF	DQE.TIFC	DQE.RILT
18	48	DQE.UTS1			
19	4C	DQE.UTS2			
20	50	DQE.DSW			
21	54	DQE.PRS			
22	58	DQE.PRM			
23	5C	Reserved	DQE.TSKF	DQE.MSPN	DQE.MST
24	60	DQE.PSSF			
25	64	DQE.PSSB			
26	68	DQE.PSPR	DQE.PSCT	DQE.ILN	DQE.RESU
27	6C	DQE.TISF			
28	70	DQE.TISB			
29	74	DQE.TIPR	DQE.TICT	DQE.SWIF	DQE.UBIO
30	78	DQE.RRSF			
31	7C	DQE.RRSB			
32	80	DQE.RRPR	DQE.RRCT	DQE.NSCT	
33	84	DQE.MRSF			
34	88	DQE.MRSB			

Dispatch Queue Entry (DQE)

Word No. (Decimal)	Byte (Hex)	0 7	8 15	16 23	24 31
35	8C	DQE.MRPR	DQE.MRCT	DQE.NWRR	DQE.NWMR
36	90	DQE.RTI	DQE.NWLM	DQE.ATI	Reserved
37	94	DQE.SAIR/DQE.TAD			
38-40	98	DQE.ABC			
41	A4	DQE.TSAP			
42-43	A8	DQE.SRID/DQE.PGOL			
	AC	DQE.SRID/DQE.PGOC		DQE.SRID/Reserved	
44-51	B0	DQE.CDIR/DQE.CVOL			
52	D0	DQE.GID	Reserved	DQE.ASH	
53	D4	DQE.ACX2			
54	D8	DQE.MRQ	DQE.MEM	DQE.MEMR	
55	DC	DQE.MRT	Reserved	DQE.RMMR	
56	E0	DQE.MAPN		DQE.CME	
57	E4	DQE.CMH		DQE.CMS	
58-63	FC	Reserved			

Byte (Hex)	Symbol	Description
0	DQE.SF	String forward linkage address; Field length = 1W; Standard linked list format; Contains address of next (top-to-bottom) entry in chain.
4	DQE.SB	String backward linkage address; Field length = 1W; Standard linked list format; Contains address of next (bottom-to-top) entry in chain.
8	DQE.CUP	Current user priority; Standard linked list format; This priority is adjusted for priority migration based on situational priority increments. Situational priority increments are based on the base level priority (DQE.BUP) of the task.
	DQE.BUP	Base priority of user task; Field length = 1B; Used by scheduler to generate DQE.CUP (current priority) based on any situational priority increments.

	DQE.IOP	I/O priority;
		Field length = 1B;
		Initially set from base priority;
		Used for I/O queue priority.

Byte (Hex)	Symbol	Description

	DQE.US	State chain index for this user task;
		Field length = 1B;
		Range: zero through X'1E';
		Indicates current state of this task, such as ready-to-run priority, I/O wait, resource block, etc.

Label	Index	Task description
FREE	00	DQE is available (in free list)
PREA	01	activation in progress
CURR	02	currently executing task or is pre-empted time-distribution task in quantum stage one
SQRT	03	ready to run (priority level 1 to 54)
SQ55	04	ready to run (priority level 55)
SQ56	05	ready to run (priority level 56)
SQ57	06	ready to run (priority level 57)
SQ58	07	ready to run (priority level 58)
SQ59	08	ready to run (priority level 59)
SQ60	09	ready to run (priority level 60)
SQ61	0A	ready to run (priority level 61)
SQ62	0B	ready to run (priority level 62)
SQ63	0C	ready to run (priority level 63)
SQ64	0D	ready to run (priority level 64)
SWTI	0E	waiting for terminal input
SWIO	0F	waiting for I/O
SWSM	10	waiting for message complete
SWSR	11	waiting for run request complete
SWLO	12	waiting for low speed output
SUSP	13	waiting for timer expiration, resume request, or message interrupt
RUNW	14	waiting for timer expiration, or run request
HOLD	15	waiting for a continue request
ANYW	16	waiting for timer expiration, no-wait I/O complete, no-wait message complete, no-wait run request complete, message interrupt, or break interrupt
SWDC	17	waiting for disk space
SWDV	18	waiting for device allocation
SWFI	19	waiting for file system
MRQ	1A	waiting for memory
SWMP	1B	waiting for memory pool
SWGQ	1C	waiting in general wait queue
CIPU	1D	current IPU task in execution
RIPU	1E	IPU requesting state

Dispatch Queue Entry (DQE)

Byte (Hex)	Symbol	Description
C	DQE.NUM	DQE entry number; Field length = 1B; Used as an index to DQE address table (DAT); Range: one through "N"(for MPL index compatibility); Used by scheduler to set C.PRNO to reflect the currently executing task. This value is also used as the MPL index. It is used by the scheduler to initialize the CPIX in the PSD before loading the map for this task.
	DQE.TAN	Task activation sequence number; Field length = 1W; This number is assigned by the activation service and uniquely identifies a task. **Note:** The most significant byte of this value is the DQE entry number and is accessible as DQE.NUM.
10	DQE.ON	Owner name; Field length = 1D.
18	DQE.LMN	Load module name; Field length = 1D.
20	DQE.PSN	Pseudonym associated with task; Field length = 1D; This parameter is an optional argument accepted by the pseudo task activation service. It can be used to uniquely identify a task within a subsystem, such as multibatch. It contains descriptive information useful to the system operator or to other tasks within a subsystem. Conventions used to generate a pseudonym are determined by the associated subsystem. A system-wide convention should be used to establish pseudonym prefix conventions to avoid confusion between subsystems.
28	DQE.USW	User status word; Field length = 1W.
2C	DQE.USHF	Scheduling flags; Field length = 1W; Used by the scheduler to indicate special status conditions.

Byte (Hex)	Symbol	Description	
		Bit	Meaning When Set
		00	load protection image requested (DQE.LPI)
		01	single copy load module (DQE.SING)
		02	task is indirectly connected (DQE.INDC)
		03	task is privileged (DQE.PRIV)
		04	task has message receiver (DQE.MSGR)
		05	task has break receiver (DQE.BRKR)
		06	task quantum stage one expired (DQE.QS1X)
		07	task quantum stage two expired (DQE.QS2X)
		08	in-swap I/O error (DQE.INER)
		09	wait I/O request outstanding (DQE.WIOA)
		10	wait I/O complete before in-progress notification (DQE.WIOC)
		11	inhibit message pseudointerrupt (DQE.INMI)
		12	batch origin task (DQE.BAOR)
		13	running in TSM environment (DQE.TMOR)
		14	task abort in progress (DQE.ABRT)
		15	task is in pre-exit state (DQE.PRXT)
		16	run receiver mode (DQE.RRMD)
		17	wait send message outstanding (DQE.WMSA)
		18	wait message complete before link to wait queue (DQE.WMSC)
		19	wait mode send run request outstanding (DQE.WRRA)
		20	wait mode send run request complete before link to wait queue (DQE.WRRC)
		21	debug associated with task (DQE.DBAT)
		22	real-time task (DQE.RT)
		23	time-distribution task initial dispatch (DQE.TDID) Set by: • H.ALOC1 on activation of T/D task. • S.EXEC51 when task is linked to wait state. • H.EXEC7 on completion of inswap or other memory request. Reset by: • S.EXEC20 on initial dispatch of task after activation • Wait state termination • In-swap
		24	task delete in progress (DQE.DELP)
		25	task abort (with abort receiver) in progress (DQE.ABRA)
		26	abort receiver established (DQE.ABRC)
		27	asynchronous abort/delete inhibited (DQE.ADIN)
		28	asynchronous delete deferred (DQE.ADDF)
		29	task is inactive (DQE.INAC)
		30	asynchronous abort deferred (DQE.AADF)
		31	activation timer in effect (DQE.ACTT)

Dispatch Queue Entry (DQE)

Byte (Hex)	Symbol	Description
30	DQE.MSD	Physical address of MIDL in TSA; Field length = 1W.
34	DQE.KCTR	Kill/abort timer; Field length = 1W.
38	DQE.MMSG	Maximum number of no wait messages allowed to be sent by this task; Field length = 1B.
	DQE.MRUN	Maximum number of no-wait run requests allowed to be sent by this task; Field length = 1B.
	DQE.MNWI	Maximum number of no-wait I/O requests allowed to be concurrently outstanding for this task; Field length = 1B.
	DQE.GQFN	Contains the generalized queue (SWGQ) function code; Field length = 1B; Function codes are queued as follows:

Code	Meaning
01	volume resource (QVRES)
02	ART space (QART)
03	mount in progress (QMNT)
04	resourcemark lock (QRSM)
05	reserved for eventmark (QEVM)
06	read wait for writer (QGEN)
07	shared memory table (QSMT)
08	synchronous resource lock (QSRL)
09	mounted volume table (QMVT)
0A	dual-port lock (QDPLK)
0B	suspend dual-port lock (QSUSP)
0C	debug wait (QDBGW)
0D	remote message area (QMSG)
0E	remote message event (QSER)
0F	remote allocate area (QASMP)
10	remote deallocate area (QDSMP)
11	remote abort area (QAMSG)
12	remote enable/disable area (QOMSG)
13	wait for TSM (QWTSM)

Byte (Hex)	Symbol	Description
3C	DQE.UF2	Scheduling flags; Field length = 1B;
	DQE.IPUF	IPU flag byte; Field length = 1B;
	DQE.NWIO	Number of no-wait I/O requests; Field length = 1B.
	DQE.SOPO	Priority bias only swapping control flags; Field length = 1B;

DQE.UF2:

Bit	Meaning if Set
0	enable debug mode break (DQE.EDB)
1	generalized wait queue time-out (DQE.GQTO)
2	task interrupts are synchronized (DQE.SYNC)
3	task is part of a job (DQE.JOB)
4	ACX-32 task flag (DQE.ACX)
5	special arithmetic function requested (DQE.AF)
6	reserved
7	run request terminated (DQE.RRT)

DQE.IPUF:

Bit	Meaning if Set
0	IPU inhibit flag (DQE.IPUH)
1	IPU bias flag (DQE.IPUB)
2	CPU only (DQE.IPUR)
3	OS execution direction flag (set when PSD is in user area) (DQE.OSD)
4	base register task (DQE.BASE)
5	Ada task (DQE.ADA)
6	PTRACE debugger task (DQE.PDBG)
7	H.PTRAC task association control bit (DQE.PTRA)

DQE.SOPO:

Bit	Meaning if Set
0	SWGQ state priority-based swapping (DQE.GQPO)
1	swap inhibit due to bit map access (DQE.BMAP)
2	inhibit swap device while accessing MDT (DQE.MDTA)
3	user swap inhibit flag (DQE.USWI)
4	user swap on priority only flag (DQE.USPO)
5-7	reserved

Dispatch Queue Entry (DQE)

Byte (Hex)	Symbol	Description
40	DQE.CQC	Current quantum count; Field length = 1W; Used by the scheduler to accumulate elapsed execution time for the task to compare the level unique stage one and stage two time-distribution values.
44	DQE.SH	Used by J.SWAPR to swap shadow memory; Field length = 1B.
	DQE.SHF	Shadow memory flag; Field length = 1B;

Bit	Meaning if Set
0	task requests shadow memory (DQE.SHAD)
1	IPU shadow memory requested (DQE.SHI)
2	IPU/CPU Common Shadow Memory requested (DQE.SHB).

Byte (Hex)	Symbol	Description
	DQE.TIFC	Timer function code; Field length = 1B;

Value	Meaning
00	not active
01	request interrupt
02	resume program from suspend (SUSP) queue
03	resume program from any-wait (ANYW) queue
04	resume program from run-request-wait (RUNW) queue
05	resume program from generalized (SWGQ) queue
06	resume program from peripheral device (SWDV) queue
07	resume program from disk space (SWDC) queue

Byte (Hex)	Symbol	Description
	DQE.RILT	Request Interrupt (RI) level for timer; Field length = 1B; Identifies the interrupt level to be requested upon timer expiration.
48	DQE.UTS1	User timer slot word 1; Field length = 1W; Current timer value; Contains negative number of timer units before time out.

Byte (Hex)	Symbol	Description
4C	DQE.UTS2	User timer slot word 2; Field length = 1W; Reset timer value; Contains negative number of timer units; Used to reset the current timer value when it expires.
50	DQE.DSW	Base mode debugger status word (PCALL); Field length = 1W.
54	DQE.PRS	Peripheral requirement specification; Field length = 1W;

Bit	Description
0-7	reserved
8-15	device type code
16-23	channel address
24-31	subchannel address or contains first word of SWGQ ID.

Byte (Hex)	Symbol	Description
58	DQE.PRM	Peripheral requirements mask; Field length = 1W;

Value	Meaning
X'00FF0000'	any device of this type code
X'00FFFF00'	any device of the specified type code on the specified channel
X'00FFFFFF'	the specified device as described by type code, channel, and subchannel address, or contains second word of SWGQ ID.

Byte (Hex)	Symbol	Description
5C	Reserved	Field length = 1B
	DQE.TSKF	Task flags; Field length = 1B;

Bit	Meaning if Set
0	real-time accounting disabled (DQE.RTAC)
1-2	reserved for RMSS
3	task is running with MPX-32 mapped out (DQE.MAPO)
4	reserved for MPX-32
5	task is demand paged (DQE.DPG)
6	inhibit page out (DQE.NPGO)
7	reserved

Dispatch Queue Entry (DQE)

Byte (Hex)	Symbol	Description
	DQE.MSPN	TSA maps required to span MIDLs and MEMLs; Field length = 1B.
	DQE.MST	Static memory type specification; Field length = 1B;

Value	Memory Class
01	E
02	H
03	S
04	H1
05	H2
06	H3

This field is used to specify the type of memory required for in-swap.

Byte (Hex)	Symbol	Description
60	DQE.PSSF	Pre-emptive system service head cell string forward linkage address; Standard head cell format; Field length = 1W; Contains address of next (top-to-bottom) entry in chain.
64	DQE.PSSB	Pre-emptive system service head cell string backward linkage address; Standard head cell format; Field length = 1W; Contains address of next (bottom-to-top) entry in chain.
68	DQE.PSPR	Pre-emptive system service head cell dummy priority (always zero); Standard head cell format; Field length = 1B.
	DQE.PSCT	Pre-emptive system service head cell number of entries in list; Standard head cell format; Field length = 1B.
	DQE.ILN	Interrupt level number; Field length = 1B; Identifies associated interrupt level for interrupt connected tasks.
	DQE.RESU	Reserved usage index; Field length = 1B.

Byte (Hex)	Symbol	Description
6C	DQE.TISF	Task interrupt head cell string forward linkage address; Standard head cell format; Field length = 1W; Contains address of next (top-to-bottom) entry in chain.
70	DQE.TISB	Task interrupt head cell string backward linkage address; Standard head cell format; Field length = 1W; Contains address of next (bottom-to-top) entry in chain.
74	DQE.TIPR	Task interrupt head cell dummy priority (always zero); Standard head cell format; Field length = 1B.
	DQE.TICT	Task interrupt head cell number of entries in list; Standard head cell format; Field length = 1B.
	DQE.SWIF	Swapping inhibit flags; Field length = 1B;

Bit	Task Meaning if Set
0	resident (DQE.RESP)
1	locked in memory (DQE.LKIM)
2	unbuffered I/O in progress (DQE.IO)
3	outswapped (DQE.OTSW)
4	leaving system (DQE.TLVS)
5	forced unswappable during terminal output (DQE.FCUS)
6	forced unswappable because swap file has not been allocated for it (DQE.FCRS)
7	imbedded in the operating system (DQE.INOS)

Byte (Hex)	Symbol	Description
	DQE.UBIO	Number of unbuffered I/O requests currently outstanding; Field length = 1B.
78	DQE.RRSF	Run receiver head cell string forward linkage address; Standard head cell format; Field length = 1W; Contains address of next (top-to-bottom) entry in chain.
7C	DQE.RRSB	Run receiver head cell string backward linkage address; Standard head cell format; Field length = 1W; Contains address of next (bottom-to-top) entry in chain.
80	DQE.RRPR	Run receiver head cell dummy priority (always zero); Standard head cell format; Field length = 1B.

Dispatch Queue Entry (DQE)

Byte (Hex)	Symbol	Description
	DQE.RRCT	Run receiver head cell number of entries in list; Standard head cell format; Field length = 1B.
	DQE.NSCT	Number of map blocks outswapped; Field length = 1H.
84	DQE.MRSF	Message receiver head cell string forward Linkage address; Standard head cell format; Field length = 1W; Contains address of next (top-to-bottom) entry in chain.
88	DQE.MRSB	Message receiver head cell string backward Linkage address; Standard head cell format; Field length = 1W; Contains address of next (bottom-to-top) entry in chain.
8C	DQE.MRPR	Message receiver head cell dummy priority (always zero); Standard head cell format; Field length = 1B.
	DQE.MRCT	Message receiver head cell number of entries in list; Standard head cell format; Field length = 1B.
	DQE.NWRR	Number of no-wait mode run requests outstanding; Field length = 1B.
	DQE.NWMR	Number of no-wait mode message requests outstanding; Field length = 1B.
90	DQE.RTI	Requested task interrupt flags; Field length = 1B;

Bit	Meaning if Set
0	reserved
1	priority one end action request. Used for pre-emptive system services. (DQE.EA1R)
2	debug break request (DQE.DBBR)
3	user break request (DQE.UBKR)
4	priority two end action request (DQE.EA2R)
5	message interrupt request (DQE.MSIR)
6-7	reserved

Byte (Hex)	Symbol	Description
	DQE.NWLM	No-wait run request limit. Field length = 1B.
	DQE.ATI	Active task interrupt flags; Field length = 1B;

Bit	Meaning if Set
0	reserved
1	priority one active end action (DQE.AEA1)
2	active debug break (DQE.ADM)
3	active user break (DQE.AUB)
4	priority two active end action (DQE.AEA)
5	active message interrupt (DQE.AMI)
6-7	reserved

Byte (Hex)	Symbol	Description
	Reserved	Field length = 1B.
94	DQE.SAIR	System action task interrupt request;

Bit	Meaning if Set
0	request for delete of this task (DQE.DELR)
1	reserved
2	hold task request (DQE.HLDR)
3	abort task request (DQE.ABTR)
4	exit task request (DQE.EXTR)
5	suspend task request (DQE.SUSR)
6	run receiver mode request (DQE.RRRQ)
7	reserved

Byte (Hex)	Symbol	Description
	DQE.TAD	TSA address (logical); Field length = 1W; Byte zero contains DQE.SAIR.
98	DQE.ABC	Abort code; Field length = 3W.
A4	DQE.TSAP	Physical address of the TSA
A8-AC	DQE.SRID	If DQE.DPG is reset; Used swap space linked list; Field length = 2W.
	DQE.PGOL	If DQE.DPG is set; Forward pointer to MPTL (MAP.SF); Field length = 1HW Backward pointer to MPTL (MAP.SB) Field length = 1HW
	DQE.PGOC	Number of pages queued for pageout Field length = 1HW
	Reserved	Field length = 1HW

Dispatch Queue Entry (DQE)

Byte (Hex)	Symbol	Description
B0	DQE.CDIR	Load module RID at activation; Field length = 8W.
	DQE.CVOL	Current working volume at activation; Field length = 8W.
D0	DQE.GID	Group swap identification; Field length = 1B.
D1	Reserved	1 Byte
D2	DQE.ASH	Number of shadow memory blocks currently allocated Field length = 1H.
D4	DQE.ACX2	Advance communication word; Field length = 1W.
D8	DQE.MRQ	Memory request doubleword; Reserved field length = 1B.
	DQE.MEM	Type of memory requested; Field length = 1B;

Value	Memory Class
01	E
02	H
03	S

Byte (Hex)	Symbol	Description
	DQE.MEMR	Number of memory blocks required; Field length = 1H.
DC	DQE.MRT	Memory request type code; Field length = 1B;

Value	Meaning
00	in-swap only
01	preactivation request
02	activation request
03	memory expansion request
04	IOCS buffer request
05	shared memory request
06	system buffer request
07	release swap file space

If DQE.MRT equals 05, the next three bytes will contain the address of the shared memory table entry.

Byte (Hex)	Symbol	Description
	Reserved	Field length = 1B.
	DQE.RMMR	Map register for requested memory; Field length = 1H.

Byte (Hex)	Symbol	Description
E0	DQE.MAPN	Inclusive span of maps in use; Field length = 1H.
	DQE.CME	Number of swappable class E map blocks currently allocated; For resident tasks, if not zero, reflects the total number of map blocks in use. Field length = 1H.
E4	DQE.CMH	Number of swappable class H map blocks currently allocated; For resident tasks, if not zero, reflects the total number of map blocks in use. Field length = 1H.
	DQE.CMS	Number of swappable class S map blocks currently allocated; For resident tasks, if not zero, reflects the total number of map blocks in use. Field length = 1H.
E8	Reserved	Reserved for MPX-32

L.5 File Control Block (FCB), 16 Word

Word	0 7	8 12	13 31
0	**Opcode (FCB.OPCD)**	Logical file code (FCB.LFC)	
1	**Reserved**		
2	General control flags (FCB.GCFG)	Special flags (FCB.SCFG)	**Reserved**
3	**Status flags (FCB.SFLG)**		
4	**Actual transfer quantity (FCB.RECL)**		
5	**Reserved**	**I/O queue address (FCB.IOQA)**	
6	**Special Status (FCB.SPST)**	Wait I/O error return address (FCB.ERRT)	
7	**Index to FPT (FCB.FPTI)**	**FAT address (FCB.FATA)**	
8	**Reserved**	I/O buffer address (FCB.ERWA)	
9	Transfer quantity (bytes) (FCB.EQTY)		
10	Random access address (FCB.ERAA)		
11	**Extended I/O status word one (FCB.IST1)**		
12	**Extended I/O status word two (FCB.IST2)**		
13	**Reserved**	No-wait I/O normal end-action service address (FCB.NWOK)	
14	**Reserved**	No-wait I/O error end-action service address (FCB.NWER)	
15	Number of buffers (FCB.BBN)	Address of blocking buffer (FCB.BBA)	

Shaded areas are set by the system.

T1FCB

Word 0

Bit 0	Reserved
Bits 1-7	Operation code (FCB.OPCD) — type of function requested of the device handler. This field is set by IOCS as a function of the executed service.
Bits 8-31	Logical file code (FCB.LFC) — any combination of three ASCII characters is allowed. The LFC must match the previously assigned LFC of the I/O resource being accessed.

Word 1

Bits 0-31	Reserved

Word 2

Bits 0-7	General control flags (FCB.GCFG) — these eight bits enable the user to specify the manner in which an operation is to be performed by IOCS. The interpretation of these bits is shown as follows:

Bit	Meaning if Set	Definition
0	NWT	IOCS returns to the user immediately after the I/O operation is queued. If reset, IOCS exits to the calling program only when the requested operation has been completed.
1	NER	error processing is not performed by either the device handler or IOCS. An error return address is ignored and a normal return is taken to the caller; however, the device status is posted in the FCB unless bit 3 is set. If reset, normal error recovery is attempted. Normal error processing for disk and magnetic tape is automatic error retry. Error processing for unit record devices except the system console is accomplished by IOCS typing the message INOP to the console, which allows the operator to retry or abort the I/O operation. If the operator aborts the I/O operation, or if automatic error retry for disk or magnetic tape is unsuccessful, an error status message is typed to the console and the error return address is taken if provided. Otherwise, the task is aborted.
2	DFI	data formatting is inhibited. Otherwise, data formatting is performed by the appropriate device handler. See Table L-1 for more explanation.
3	NST	device handlers perform no status checking and no status information is returned. All I/O appears to complete without error. Otherwise, status checking is performed and status information is returned as necessary.
4	RAN	file accessing occurs in the random mode. Otherwise, sequential accessing is performed. Note: This bit is set if word 2 bit 12 is set.
5		reserved (M.FILE)
6	EXP	must be 1 for 16-word FCB.
7	IEC	this bit is reserved for internal IOCS use.

Bits 8-12 Special Control Specification (FCB.SCFG). — This field contains device control specifications unique to certain devices. Interpretation and processing of these specifications are performed by the device handlers. A bit setting is meaningful only when a particular type of device is assigned as indicated in Table L-1.

Bits 13-31 reserved for extended control specifications

Bit	Meaning if Set	Definition
13	RXON	software read flow control required for 8-Line ACM (FCB.RXON)

Table L-1
Special Control Flags

Device	Bit 2=0	Bit 2=1	Bit 8=0	Bit 8=1	Bit 9=0	Bit 9=1
Line Printer (LP)	Interpret first character as carriage control	Interpret first character as data See bit 8	Form control	No form control		
Discs, (DM,DF, FL)	Report EOF if X'0FE0FE0F' encountered in word 0 of 1st block during read of unblocked file	X'0FE0FE0F' in word 0 not recognized as EOF				
8-Line Asynchronous Communications Multiplexer (TY)	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**
	Perform special character formatting	No special character formatting	ASCII control passed as data	ASCII control character detect	Echo by controller	No echo by controller
	M.WRIT	**M.WRIT**	**SVC 1,X'3E'**	**SVC 1,X'3E'**	**M.WRIT**	**M.WRIT**
	Interpret first character as carriage control	Interpret first character as data	Stop transmitting break	Start transmitting break	Normal write	Initialize device (load UART parameters)

Device	Bit 10=0	Bit 10=1	Bit 11=0	Bit 11=1	Bit 12=0	Bit 12=1
Line Printer (LP)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Discs, (DM,DF, FL)					Normal read	Read with byte granularity (word 2 bit 4 set)
8-Line Asynchronous Communications Multiplexer (TY)	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**
	(If bit 2=0) convert lower case character to upper case	Inhibit conversion	No special character detect	Special character detect	Do not purge type ahead buffer	Purge type ahead buffer
	M.WRIT	**M.WRIT**	**M.WRIT**	**M.WRIT**	**M.WRIT**	**M.WRIT**
			Normal write	Write with input sub-channel monitoring plus software flow control		

Continued on next page

Table L-1
Special Control Flags (*Continued*)

Device						Bit 10	Bit 11	Bit 12
ALIM (Asynchronous Line Interface Module) Terminals (TY)	Read: receive data (bytes) defined for transfer count	**Bit 2**	**Bit 8**	**Bit 9**	**Read**	On Read:		
		0	1	0	=Blind mode reset	1= Inhibit conversion of lower case characters to upper case		
		0	0	1	=Echo on read			
		1	N/A	N/A	=Receive data			
		0	0	0	=Receive data			
					Write			
	Write: formatted	0	N/A	0	=Formatted write	0= Convert		
		0	N/A	1	=Initialize device			
		1	N/A	N/A	=Unformatted write			

Word 3

Bits 0-31 Status word (FCB.SFLG) — 32 indicator bits are set by IOCS to indicate the status, error, and abnormal conditions detected during the current or previous operation. The assignment of these bits is shown as follows:

Bits	Meaning if Set	Definition
0	OP	operation in progress. Request has been queued. (Note: Reset after post I/O processing complete.)
1	ERR	error condition found
2	BB	invalid blocking buffer control pointers have been encountered during file blocking or unblocking
3	PRO	write protect violation
4	INOP	device inoperable
5	BOM	beginning-of-medium (BOM) (load point) or illegal volume number (multivolume magnetic tape)
6	EOF	end-of-file
7	EOM	end-of-medium (end of tape, end of disk file)
8-9		reserved
10	TIME	last command exceeded time-out value and was terminated
11-15		reserved
16	ECHO	echo
17	INT	post program-controlled interrupt
18	LEN	incorrect length
19	PROG	channel program check
20	DATA	channel data check
21	CTRL	channel control check
22	INTF	interface check
23	CHAI	chaining check
24	BUSY	busy
25	ST	status modified
26	CTR	controller end
27	ATTN	attention
28	CHA	channel end
29	DEV	device end
30	CHK	unit check
31	EXC	unit exception

Word 4

Bits 0-31 Record length (FCB.RECL) — this field is set by IOCS to indicate the actual number of bytes transferred during read/write operations.

Word 5

Bits 0-7	Reserved
Bits 8-31	I/O queue address (FCB.IOQA) — this field is used by IOCS to point to the I/O queue for an I/O request initiated from this FCB

Word 6

Bits 0-7 Special status bits (FCB.SPST). The interpretation of these bits is shown below:

Bits	Definition
0	no-wait normal end action not taken
1	no-wait error end action not taken
2	request killed, I/O not issued
3	if set, exceptional condition has occurred in the I/O request
4	if set, software read flow control required
5-7	reserved

Bits 8-31 Wait I/O error return address (FCB.ERRT) — this field is set by the user and contains the address to which control is to be transferred in the case of an unrecoverable error when control bits 1 and 3 of word 2 are reset. If this field is not initialized and an unrecoverable error is detected under the above conditions, the requesting task is aborted.

Word 7

Bits 0-7	Index to FPT (FCB.FPTI) — this field is set by IOCS to index into the associated entry in the file pointer table (FPT)
Bits 8-31	FAT address (FCB.FATA) — this field is set by IOCS to point to the associated file assignment table (FAT) entry.

Word 8

Bits 0-7	Reserved
Bits 8-31	Data buffer address (FCB.ERWA) — start address of data area for read or write operations. (24 bit pure address)

Word 9

Bits 0-31	Quantity (FCB.EQTY)— number of bytes of data to be transferred

Word 10

Bits 0-31	Random access address (FCB.ERAA) — this field contains a block number (zero origin) relative to the beginning of the disk file. It is the start address for the current read or write operation with word 2 bit 4 set and word 2 bit 12 reset. or For disk read requests with word 2 bits 4 and 12 set (read with byte granularity), this word defines the byte offset relative to the beginning of the file. Note: If word 9 is zero, the file retains its position prior to the call.

Word 11

Bits 0-31	Status word one (FCB.IST1) — these are the first 32 bits of status returned by the sense command

Word 12

Bits 0-31	Status word two (FCB.IST2) — these are the second 32 bits of status returned by the sense command

Word 13

Bits 0-7	Reserved
Bits 8-31	No-wait I/O (FCB.NWOK) — normal completion return address. This user routine must be exited by calling the M.XIEA service.

Word 14

Bits 0-7	Reserved
Bits 8-31	No-wait I/O (FCB.NWER) — error completion return address. This user routine must be exited by calling the M.XIEA service.

Word 15 (Applicable only to volume resource.)

Bits 0-7	(FCB.BBN) — Number of 192 word buffers for user supplied blocking buffers. A value of one or zero in this field specifies one blocking buffer.
Bits 8-31	Blocking buffer address (FCB.BBA) — starting address of a contiguous area of memory FCB.BBN buffers long

Table L-2
Device Functions (Standard Devices)

Operation	IOCS Op Code	Line Printer (LP)	Mag Tape (M9/MT)	Disk (DM/DF/ DC/Floppy)	Handler=F8XIO (8-Line)
Open (M.FILE)	0	IOCS opens	IOCS opens	IOCS opens	Initialize IOP channel if necessary
Rewind (M.RWND)	1	Eject,set BOM bit word 3 bit 5 in FCB	Rewind Tape	Set current block address to zero (FAT)	SENSE operation
Read Record (M.READ)	2	Spec error	Read to data buffer	Read to data buffer	Read to data buffer
Write record (M.WRIT)	3	Write from data buffer	Write from data buffer. If blocked writes *n* data buffers to blocking buffer before output	Write from data buffer. If blocked IOCS writes *n* data buffers to blocking buffer before output	Write record to terminal
Write EOF (M.WEOF)	4	NOP*	Write EOF	If blocked, IOCS writes EOF. If unblocked writes X'0FE0FE0F'	NOP*
Execute Channel	5	Spec error	Execute Channel Program	Execute Channel Program	Execute channel Program

*NOP — No operation performed

Continued on next page

Table L-2
Device Functions (Standard Devices) *(Continued)*

Operation	IOCS Op Code	Line Printer (LP)	Mag Tape (M9/MT)	Disk (DM/DF/ DC/Floppy)	Handler=F8XIO (8-Line)
Advance Record (M.FWRD)	6	Spec error	Advance record	If blocked, advance record. If unblocked, advance one 192W block.	Set data terminal ready
Advance File (M.FWRD)	7	Spec error	Advance file (past EOF)	Spec error	Reset data terminal ready
Backspace Record (M.BACK)	8	Spec error	Backspace record	If blocked, backspace record. If unblocked backspace one 192W block	Used by J.TINIT to initialize terminals
Backspace File (M.BACK)	9	Spec error	Backspace file to previous EOF	Spec error	Reset request to send command
Upspace (M.UPSP)	A	Upspace	Multivolume only. If BOT, writes volume record. If EOT, performs ERASE, writes EOF, and issues MOUNT message.	Spec error on F-class disks. For floppy only: format diskette. New diskettes must be formatted prior to normal usage.	Set request to send command
Erase or Punch Trailer Not user IOCS/handler provides call automatically	B	NOP	Multivolume only. Same as upspace above. Erases 4" of tape before writing	NOP	Set/reset break (depends on flags in FCB)

Continued on next page

Table L-2
Device Functions (Standard Devices) *(Continued)*

Operation	IOCS Op Code	Line Printer (LP)	Mag Tape (M9/MT)	Disk (DM/DF/ DC/Floppy)	Handler=F8XIO (8-Line)
Eject/ Punch Leader (M.EJECT)	C	Eject to top of form	Write dummy record with eject control character as first character	NOP	Define special character
Close (M.CLSE)	D	IOCS closes	IOCS closes	IOCS closes	NOP
Reserve FHD Port	E	Spec error	Spec error	Reserve port-4MB disk only. Else, spec error Reserve Dual Ported Disk	Set single-channel operation (default) command
Release FHD Port	F	Spec error	Spec error	Release port-4MB disk only. Else, spec error Reserve Dual Ported Disk	Set dual-channel operation

Table L-3
Device Functions (Terminals, Handler Action Only)

Operation	IOCS Op Code	Handler = H.ASMP (ALIM)
Open M.FILE	0	NOP*
Rewind M.RWND	1	NOP*
Read record M.READ	2	Read to data buffer
Write record M.WRIT	3	Write record to terminal
Write EOF M.WEOF	4	NOP*
Execute channel	5	Execute channel
Advance record M.FWRD	6	Connect communications channel
Advance file M.FWRD	7	Disconnect communications channel
Backspace record M.BACK	8	Initialize device and set time-out value
Backspace file M.BACK	9	Clear break status flag word
Upspace M.UPSP	A	Spec error**
Erase/punch trailer	B	Transmit break
Eject/punch leader M.EJECT	C	Spec error**
Close M.CLSE	D	NOP*
Reserve FHD port	E	Spec error**
Release FHD port	F	Spec error**

* NOP = No operation performed

** Spec Error = Illegal operation code

Table L-4
Standard Carriage Control Characters and Interpretation

Control Character	Hexa-decimal Value	Result on a Terminal	Result on Directly Allocated Printer (Serial or parallel)	SLO
Blank	20	One linefeed, one carriage return before write	Single space before print	Single space before print
0	30	Two linefeeds, one carriage return before write	Double space before print	Double space before
1	31	Five linefeeds, one carriage return before write	Page eject (slew) before print	Page eject (slew) before print
+	2B	No linefeed, no carriage return before write (line append)	No space before print (overprint)	No space before print (overprint)
-	2D	Five linefeeds, one carriage return before write	Single space before print	Page eject, save and print up to three user supplied title lines. See Note 1.
<	3C	One linefeed, one carriage return before write	Single space before print	Set inhibit spooler title line in this file.
>	3E	One linefeed, one carriage return before write	Single space before print	Set enable spooler title line in this file.
=	3D	One linefeed, one carriage return before write	Single space before print	Page eject and clear up to three user-supplied title lines in this file.

Notes:

1. User-supplied title lines have the same effect as this character. Supplying a fourth title line clears the first three, but only one page is ejected. User-supplied titles are retained by the spooler and are repeated at the top of each page until cleared or the spool file ends.

L.6 File Control Block, Compatible Mode 8 word (FCB)

Word	0 7	8 12	13 31
0	Opcode (FCB.OPCD)	Logical file code (FCB.LFC)	
1	Transfer control word (FCB.TCW)		
2	General control flags (FCB.GCFG)	Special flags (FCB.SCFG)	Random access address (FCB.CBRA)
3	Status flags (FCB.SFLG)		
4	Actual transfer quantity (FCB.RECL)		
5	Reserved	I/O queue address (FCB.IOQA)	
6	Special Status (FCB.SPST)	Wait I/O error return address (FCB.ERRT)	
7	Index to FPT (FCB.FPTI)	FAT address (FCB.FATA)	

Shaded areas are set by the system.

A.L8W.FCB

Word 0

Bit 0	Reserved
Bits 1-7	Operation code (FCB.OPCD) — type of function requested of the device handler. This field is set by IOCS as a function of the requested service.
Bits 8-31	Logical file code (FCB.LFC) — any combination of three ASCII characters is allowed.

Word 1 (FCB.TCW)

This word supplies a transfer control word (TCW) that describes a data buffer and transfer quantity. If no TCW definition is supplied, the transfer buffer defaults to location zero of the task's logical address space and is 4096 words long.

Bits 0-11	Quantity — 12 bit field specifying the number of data items to be transferred. This quantity must include the carriage control character, if applicable. The transfer quantity is in units determined by the address in bits 12 to 31.

Bits 12-31 Format code and buffer address— bits 12, 30 and 31 specify byte, halfword, or word quantities for data transfers. They are interpreted as follows:

Type of Transfer	F (12)	C (30,31)	Address
Byte	1	xx	13-31
Halfword	0	x1	13-30
Word	0	00	13-29

Word 2

Bits 0-7 General control flags (FCB.GCFG) — these eight bits enable the user to specify the manner in which an operation is to be performed by IOCS. The interpretation of these bits is shown below:

File Control Block, Compatible Mode 8 word (FCB)

Bit	Meaning if Set	Definition
0	NWT	IOCS returns to the user immediately after the I/O operation is queued. If reset, IOCS exits to the calling program only when the requested operation has been completed.
1	NER	error processing is not performed by either the device handler or IOCS. An error return address is ignored and a normal return is taken to the caller; however, the device status is posted in the FCB unless bit 3 is set. If reset, normal error recovery is attempted. Normal error processing for disk and magnetic tape is automatic error retry. Error processing for unit record devices except the system console is accomplished by IOCS typing the message INOP to the console, which allows the operator to retry or abort the I/O operation. If the operator aborts the I/O operation, or if automatic error retry for disk or magnetic tape is unsuccessful, an error status message is typed to the console and the error return address is taken if provided. Otherwise, the task is aborted.
2	DFI	data formatting is inhibited. Otherwise, data formatting is performed by the appropriate device handler. See Table L-5 for more explanation.
3	NST	device handlers perform no status checking and no status information is returned. All I/O appears to complete without error. Otherwise, status checking is performed and status information is returned as necessary.
4	RAN	file accessing occurs in the random mode. Otherwise, sequential accessing is performed.
5		reserved (M.FILE)
6	EXP	must be 0 for 8 word FCB.
7	IEC	this bit is reserved for internal IOCS use.

Bits 8-12	Special Control Specification (FCB.SCFG). — This field contains device control specifications unique to certain devices. Interpretation and processing of these specifications are performed by the device handlers. A bit setting is meaningful only when a particular type of device is assigned as indicated in Table L-2.
Bits 13-31	Random access address (FCB.CBRA) — This field contains a block number (zero origin) relative to the beginning of the disk file, and specifies the base address for read or write operations.

Table L-5
Special Control Flags (8 Word FCB)

Device	Bit 2=0	Bit 2=1	Bit 8=0	Bit 8=1	Bit 9=0	Bit 9=1
Line Printer (LP)	Interpret first character as carriage control	Interpret first character as data See bit 8	Form control	No form control		
Discs, (DM,DF, FL)	Report EOF if X'0FE0FE0F' encountered in word 0 of 1st block during read of unblocked file	X'0FE0FE0F' in word 0 not recognized as EOF				
8-Line Asynchronous Communications Multiplexer (TY)	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**
	Perform special character formatting	No special character formatting	ASCII control passed as data	ASCII control character detect	Echo by controller	No echo by controller
	M.WRIT	**M.WRIT**	**SVC 1,X'3E'**	**SVC 1,X'3E'**	**M.WRIT**	**M.WRIT**
	Interpret first character as carriage control	Interpret first character as data	Stop transmitting break	Start transmitting break	Normal write	Initialize device (load UART parameters)

Device	Bit 10=0	Bit 10=1	Bit 11=0	Bit 11=1	Bit 12=0	Bit 12=1
Line Printer (LP)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Discs, (DM,DF, FL)						
8-Line Asynchronous Communications Multiplexer (TY)	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**
	(If bit 2=0) convert lower case character to upper case	Inhibit conversion	No special character detect	Special character detect	Do not purge type ahead buffer	Purge type ahead buffer
	M.WRIT	**M.WRIT**	**M.WRIT**	**M.WRIT**	**M.WRIT**	**M.WRIT**
			Normal write	Write with input sub-channel monitoring plus software flow control		

Continued on next page

Table L-5
Special Control Flags (8 Word FCB) (*Continued*)

Device						Bit 10	Bit 11	Bit 12
ALIM (Asynch-ronous Line Interface Module) Terminals (TY)	Read: receive data (bytes) defined for transfer count	**Bit 2**	**Bit 8**	**Bit 9**	**Read**	On Read:		
		0	1	0	=Blind mode reset	1= Inhibit conversion of lower case characters to upper case		
		0	0	1	=Echo on read			
		1	N/A	N/A	=Receive data			
		0	0	0	=Receive data			
					Write			
	Write: formatted	0	N/A	0	=Formatted write	0= Convert		
		0	N/A	1	=Initialize device			
		1	N/A	N/A	=Unformatted write			

Word 3

Bits 0-31 Status word (FCB.SFLG) — 32 indicator bits are set by IOCS to indicate the status, error, and abnormal conditions detected during the current or previous operation. The assignment of these bits is shown as follows:

Bits	Meaning if Set	Definition
0	OP	operation in progress. Request has been queued. (Note: Reset after post I/O processing complete.)
1	ERR	error condition found
2	BB	invalid blocking buffer control pointers have been encountered during file blocking or unblocking
3	PRO	write protect violation
4	INOP	device inoperable
5	BOM	beginning-of-medium (BOM) (load point) or illegal volume number (multivolume magnetic tape)
6	EOF	end-of-file
7	EOM	end-of-medium (end of tape, end of disk file)
8-9		reserved
10	TIME	last command exceeded time-out value and was terminated
11-15		reserved
16	ECHO	echo
17	INT	post program-controlled interrupt
18	LEN	incorrect length
19	PROG	channel program check
20	DATA	channel data check
21	CTRL	channel control check
22	INTF	interface check
23	CHAI	chaining check
24	BUSY	busy
25	ST	status modified
26	CTR	controller end
27	ATTN	attention
28	CHA	channel end
29	DEV	device end
30	CHK	unit check
31	EXC	unit exception

File Control Block, Compatible Mode 8 word (FCB)

Word 4

Bits 0-31	Record length (FCB.RECL) — this field is set by IOCS to indicate the actual number of bytes transferred during read/write operations.

Word 5

Bits 0-7	Reserved
Bits 8-31	I/O queue address (FCB.IOQA) — this field is used by IOCS to point to the I/O queue for an I/O request initiated from this FCB

Word 6

Bits 0-7 Special status bits (FCB.SPST). The interpretation of these bits is shown below:

Bits	Definition
0	no-wait normal end action not taken
1	no-wait error end action not taken
2	kill command, I/O not issued
3	if set, exceptional condition has occurred in the I/O request
4	if set, software read flow control required
5-7	reserved

Bits 8-31 Wait I/O error return address (FCB.ERRT) — this field is set by the user and contains the address to which control is to be transferred in the case of an unrecoverable error when control bits 1 and 3 of word 2 are reset. If this field is not initialized and an unrecoverable error is detected under the above conditions, the user is aborted.

Word 7

Bits 0-7	Index to FPT (FCB.FPTI) — this field indexes into the appropriate entry in the file pointer table (FPT)
Bits 8-31	FAT address (FCB.FATA) — this field points to the file assignment table (FAT) entry associated with all I/O performed for this FCB. This field is supplied by IOCS.

L.7 File Control Block (FCB), High Speed Data

The following section details the 16 words that make up the FCB for the HSD.

Word	0 … 7	8 … 15	16 … 23	24 … 31
0	Opcode (FCB.OPCD)	Logical file code (FCB.LFC)		
1	Reserved			
2	General control flags (FCB.GCFG)	Special flags (FCB.SCFG)	Reserved	UDDCMD of IOCD if bit 11 of word 2 is set
3	Status flags (FCB.SFLG)			
4	Record Length in bytes (FCB.RECL)			
5	Reserved	I/O queue address (FCB.IOQA)		
6	Special Status (FCB.SPST)	Wait I/O error return address (FCB.ERRT)		
7	Index to FPT (FCB.FPTI)	FAT address (FCB.FATA)		
8	Reserved	Data address (FCB.ERWA)		
9	Transfer quantity (bytes) (FCB.EQTY)			
10	Device command for non-EXCPM (FCB.ERAA)			
11	Reserved			
12	Extended I/O status word two (FCB.IST2)			
13	Reserved	No-wait I/O normal end-action service address (FCB.NWOK)		
14	Reserved	No-wait I/O error end-action service address (FCB.NWER)		
15	Reserved			

Shaded areas are set by the system.

T2FCB

Word 0

Bit 0	Reserved
Bits 1-7	Contain the operation code, set by IOCS that specifies the type of function requested of H.HSDG.
Bits 8-31	Contain the logical file code associated with the device for the I/O operation.

Word 1

This word is reserved and should be set to zero.

Word 2

Bits 0-7 Contain control flags that enable the user to specify how an operation is to be performed by IOCS. Following is the meaning of these bits when they are set:

Bit	Meaning When Set
0	IOCS returns to the user immediately after the I/O operation is queued (no wait I/O). If reset, IOCS exits to the calling program only when the HSD completes the requested operation (wait I/O).
1	H.HSDG and IOCS do not perform error processing. IOCS ignores the error return address and takes a normal return to the caller. H.HSDG posts device status in the FCB (unless bit 3 is set). If reset, H.HSDG and IOCS perform error processing.
2	specifies physical execute channel program. If reset, specifies logical channel program or non-execute channel program I/O request.
3	IOCS performs no status checking and does not return status information. All I/O appears to complete without error. If reset, IOCS performs status checking and returns status information.
4, 5	Reserved, should be zero.
6	specifies 16 word FCB. Must be set to 1.
7	reserved for internal IOCS use.

Bits 8-23 contain the following special flags:

Bit	Meaning When Set
8	specifies request device status after a transfer. H.HSDG adds an IOCB to the IOCL to retrieve device-specific status after the data transfer completes.
9	specifies send device command prior to data transfer. H.HSDG prefixes the transfer with an IOCB that sends a device command word to the device. The value sent is the 32-bits contained in word 10 of the FCB.
10	specifies disable time out for this request. This bit indicates the operation will take an indeterminable period of time. In most cases this applies only to read operations.
11	specifies set UDDCMD from the least significant byte of word 2. This bit indicates that the UDDCMD byte in the data transfer IOCB must be set to the least significant byte of FCB word 2. This allows the user to pass additional control information to the device without modifying the device driver.
12	specifies disable asynchronous status notification during no-wait I/O.
13	specifies the execute channel program request INIT. By setting this bit, all preliminary I/O data structures are set up based on the I/O command list address provided in word 8 of the FCB. When set, this bit prepares for future cyclic I/O requests but does not issue any I/O.
14	specifies the execute channel program request GO. This bit issues an SIO for the most recently processed INIT execute channel program request (see bit 13).
15-23	reserved
Note:	For further information on the HSD FCB please see the H.HSDG section in the MPX-32 Technical Manual Volume II.

Bits 24-31 if bit 11 is set, these bits define the UDDCMD field of the generated IOCB, overriding the default value from a handler table. This field applies only to FCB format.

Word 3

IOCS uses this word to indicate status, error, and abnormal conditions detected during the current or previous operation. Following is the meaning of the bits when they are set:

Bit	Meaning When Set
0	operation in progress. Request has been queued. This bit is reset after post I/O processing completes.
1	error condition found
2, 3	not applicable, should never be returned
4	device inoperable, HSD not present or offline
5-15	not applicable, should never be returned
16	a time-out occurred and a CD terminate was issued.
17, 18	not applicable, should never be returned
19	there was data remaining in the HSD fifo when the transfer count equaled zero.
20	a parity error occurred during the current data transfer.
21	a non-present memory error occurred during the current data transfer.
22	program violation. An invalid operation code was detected.
23	device inoperative
24	HSD data buffer overflow. Some data from the device was lost.
25	external termination
26	IOCB address error
27	error on TI address fetch
28	device EOB
29	Non-device access errors precluded request queuing. For a list of the errors, see word 12.
30, 31	non-execute channel program type of IOCB in error as follows:

Value	Meaning
00	data transfer
01	device status
10	command transfer

Word 4

This word specifies the record length. For non-execute channel program I/O, IOCS sets this word to indicate the number of bytes transferred during read or write operations.

Word 5	
Bits 0-7	reserved
Bits 8-31	specify the IOQ address. IOCS sets this field to point to the IOQ entry initiated from this FCB.
Word 6	
Bits 0-7	specify special status as follows:

Bit	Meaning When Set
0	no-wait normal end action not taken
1	no-wait error end action not taken
2	kill command, I/O not issued
3	an exception condition has occurred in the I/O request
4	not used
5-7	reserved

Bits 8-31	contain the wait I/O error return address. The user sets this field to the address where control is to be transferred for unrecoverable errors when bits 0, 1, and 3 of word 2 are reset. If this field is not initialized and an unrecoverable error is detected under the above conditions, the user task is aborted.
Word 7	
Bits 0-7	set by the I/O control system (IOCS), contains an index to the file pointer table (FPT) entry for this I/O operation.
Bits 8-15	supplied by the IOCS, points to the file assignment table (FAT) entry associated with this FCB.

Word 8

Bits 0-7	reserved
Bits 8-31	these bits are used as the data address, a logical IOCL address, or a physical IOCL address as follows:

Data address – This is the starting address of the data area for FCB format I/O operations. This address must be a word address.

Logical IOCL address – This is a logical, doubleword address that points to a user-supplied IOCL for SIO format I/O operations. For more information about SIO format, refer to Reference Manual Volume I, Chapter 3. The execute channel program entry point (H.IOCS,10) must be used and bit 2 of word 2 of the FCB is reset. All addresses within the IOCL are assumed to be logical and map block boundary crossings need not be resolved.

Physical IOCL address – This is a physical, doubleword address that points to a user-supplied IOCL for SIO format I/O operations. The execute channel program entry point (H.IOCS,10) must be used and bit 2 of word 2 of the FCB is set. All addresses within the IOCL are assumed to be physical and all map block boundary crossings are assumed to be resolved.

Word 9
This word specifies the number of bytes of data to be transferred.

Word 10
For nonexecute channel program format, this word defines a device command.

Word 11
Reserved — should be set to zero.

Word 12
This word contains status sent from the user's device or if bit 29 of word 3 is set, this word defines the opcode processor (EP5) detected errors as follows:

Value	Explanation
1	request made with non-expanded FCB
2	FCB format transfer count was zero
3	FCB format, byte transfer count was not a multiple of 4 bytes
4	SIO format with a physical IOCL request by an unprivileged caller
5	SIO format with a physical IOCL request by a nonresident caller
6	first IOCB in caller's IOCL is a transfer in channel
7	caller's IOCL not on a doubleword boundary
8	SIO format IOCL contains an IOCB with a zero transfer count
9	infinite transfer in channel loop
10	consecutive SOBNZ's in IOCL

11	SOBNZ target is not in the IOCL
12	the transfer address is not on a word boundary
13	unprivileged caller's input buffer includes protected locations
14	unprivileged caller's input buffer is unmapped either in MPX-32 or below DSECT
15	cyclic I/O request was made for which no cyclic IOQ is current
16	cyclic I/O request was made and permanent IOQ support was not sysgened into the system

Word 13

Bits 0-7 reserved

Bits 8-31 contain the address of the user-supplied routine to branch to for no-wait I/O normal completion. This routine must be terminated by calling H.IOCS,34 (no-wait I/O end action return). If word 2 bit 12 is reset, this address plus one word is the location where control is transferred on asynchronous status notification.

Word 14

Bits 0-7 reserved

Bits 8-31 contain the address of the user-supplied routine to branch to for no-wait I/O error completion. This routine must be terminated by calling H.IOCS,34 (no-wait I/O end action return).

Word 15

Reserved — should be set to zero.

L.8 File Pointer Table (FPT)

The file pointer table (FPT) provides the linkage between the file control block (FCB) and the file assignment table (FAT). It also allows for multiple logical file code assignments to be made equivalent to the same FAT. The linkage to the FAT is performed at assignment. The linkage to the FCB is performed at open and is re-established if necessary for every operation at opcode processing time. The FPT resides in the task's service area.

FPT entries one to six are reserved for the system as follows:

Entry 1 - System LFC *s*
Entry 2 - Load module LFC *LM
Entry 3 - H.VOMM resource descriptor LFC (1)
Entry 4 - H.VOMM directory LFC (2)
Entry 5 - H.VOMM DMAP/SMAP LFC (3)
Entry 6 - H.VOMM modify resource descriptor LFC X'FFFEE'

Each FPT entry has the following format:

	0 7	8 15 16 23 24 31
Word 0	Reserved	Logical file code (FPT.LFC)
1	Flags (FPT.FLGS). See Note 1.	FCB address (FPT.FCBA)
2	Reserved	FAT address (FPT.FATA)

Notes:

1. Bits in FPT.FLGS are assigned as follows:

Bit	Meaning if Set
0	reserved
1	multiple FPT entries exist that point to the same FAT (i.e., $ASSIGN4 or $ASSIGN *lfc* TO LFC = *lfc* statements)
2	FPT busy flag
3	FPT open
4	this FPT entry is not in use
5	pseudo-SYC assignment (used by TSM)
6	pseudo-FPT for unassigned tempory file
7	reserved

L.9 Parameter Task Activation Block

The following is the structure of the expanded parameter task activation block:

Byte	Word	0–7	8–15	16–23	24–31
0	0	PTA.FLAG	PTA.NRRS	PTA.ALLO	PTA.MEMS
4	1	PTA.NBUF	PTA.NFIL	PTA.PRIO	PTA.SEGS
8	2-3	PTA.NAME			
10	4-5	PTA.PSN			
18	6-7	PTA.ON			
20	8-9	PTA.PROJ			
28	10	PTA.VAT	PTA.FLG2	PTA.EXTD	
2C	11	PTA.PGOW			
30	12	PTA.TSW			
34	13	PTA.RPTR			
38	14	PTA.PGO2			
3C	15	PTA.FSIZ		PTA.RSIZ	
40	16-19	Reserved (zero)			
50-*nn*	20-*nn*	RRS List			

Byte (Hex)	Symbol	Description
0	PTA.FLAG	contains the following:

Bit	Contents
0	reserved
1	job oriented (PTA.JOB)
2	terminal task (PTA.TERM)
3	batch task (PTA.BTCH)
4	debug overlay required (PTA.DOLY)
5	resident (PTA.RESD)
6	directive file active (PTA.DFIL)
7	SLO assigned to SYC (PTA.SLO)

For unprivileged callers, bits 0-3 are not applicable. These characteristics are inherited from the parent task.

Byte (Hex)	Symbol	Description
1	PTA.NRRS	number of resource requirements or zero if same as summary entries in the load module or executable image preamble

Parameter Task Activation Block

Byte (Hex)	Symbol	Description
2	PTA.ALLO	memory requirement: number of 512-word pages exclusive of TSA, or zero if same as the preamble
3	PTA.MEMS	memory class (ASCII E, H or S) or zero if memory class is to be taken from the preamble. If the memory class is to be taken from the preamble, the caller has the option of specifying the task's logical address space in this field as follows: **Bits** / **Contents** 0-3 — hexadecimal value 0 through F representing the task's logical address space in megabytes where zero is 1MB and F is 16MB 4-7 — zero
4	PTA.NBUF	the number of blocking buffers required or zero if same as the preamble
5	PTA.NFIL	the number of FAT/FPT pairs to be reserved or zero if same as the preamble
6	PTA.PRIO	the priority level at which the task is to be activated or zero for the cataloged load module priority. See the Parameter Send Block section in Chapter 2 of the MPX-32 Reference Manual Volume I, for more details.
7	PTA.SEGS	the segment definition count or reserved (zero)
8	PTA.NAME	contains the load module or executable image name, left justified and blank filled, or word 2 is zero and word 3 contains a pathname vector or RID vector
10	PTA.PSN	contains the 1- to 8-character ASCII pseudonym, left justified and blank filled, to be associated with the task or zero if no pseudonym is desired. For unprivileged callers, this attribute is inherited from the parent task if zero is supplied or the parent is in a terminal or batch job environment.
18	PTA.ON	contains the 1- to 8-character ASCII owner name, left-justified and blank-filled, to be associated with the task or zero if the task to default to the current owner name. Valid only when task has system administrator attribute.
20	PTA.PROJ	contains the 1- to 8-character ASCII project name, left-justified and blank-filled, to be associated with files referenced by this task, or zero if same as LMIT
28	PTA.VAT	the number of volume assignment table (VAT) entries to reserve for dynamic mount requests or zero if same as the preamble

Byte (Hex)	Symbol	Description
29	PTA.FLG2	contains the following flags:

Bit	Meaning if Set
0	debug activating task (PTA.DBUG)
1	Command Line Recall and Edit is in effect for the task (PTA.CLRE)
2	NOTSA option (PTA.NTSA)
3	TSA option (PTA.TSA)
4	expanded PTASK block flag (must be set to use options 33-64) (PTA.EBLK)
5	reserved (zero)
6	enables NOMAPOUT option (PTA.NMAP)
7	enables MAPOUT option (PTA.MAP)

Byte (Hex)	Symbol	Description
2A	PTA.EXTD	contains the following values:

Bit	Meaning if Set
-1	maxaddr of extended MPX-32 and TSA
-2	minaddr of extended MPX-32 and TSA
0	invalid with PTA.TSA or PTA.NTSA option
n	a positive number representing a map block of MPX-32 and TSA

Byte (Hex)	Symbol	Description
2C	PTA.PGOW	contains the initial value of the task option word or zero
30	PTA.TSW	contains the initial value of the task status word or zero
34	PTA.RPTR	contains a pointer to the resource requirement summary list or, if an expanded PTASK block is not used, the RRS list begins here (see resource requirement summary list description below)
38	PTA.PGO2	contains the initial value of the second task option word
3C	PTA.FSIZ	contains the length of the fixed portion of the PTASK block in bytes
3E	PTA.RSIZ	contains the number of bytes of the resource requirement summary
40	Reserved	

Byte (Hex)	Symbol	Description
50		resource requirement summary list. Each entry contains a variable length RRS. The RRS list has up to 384 words. Each entry must be doubleword bounded. Each entry is compared with the RRS entries in the LMIT. If the logical file code currently exists, the specified LFC assignment will override the cataloged assignment, otherwise the special assignment will be treated as an additional requirement and merged into the list. If MPX-32 Revision 1.*x* format of the RRS is specified, it is converted to the format acceptable for assignment processing by the Resource Management Module (H.REMM). See MPX-32 Revision 1.*x* Technical Manual for format of the RRS.

L.10 TSM Procedure Call Block (PCB)

The PCB contains the information necessary for the service to complete a procedure call. The format of the PCB is as follows:

	0 7	8 15	16 23	24 31
Word 0		Send buffer address (PCB.SBA)		
1	Send quantity (PCB.SQUA)			
2		Return buffer address (PCB.RBA)		
3	Actual return length (PCB.ACRP)		Return buffer length (PCB.RPBL)	

Send buffer address	is the address of a character string that represents a valid TSM procedure call directive
Send quantity	contains the length in bytes of the TSM procedure call directive
Return buffer address	is the address of a buffer to contain either valid return information or an error message if CC1 is set and R7 contains a value of 1
Actual return length	is the number of bytes returned from the procedure call
Return buffer length	is the size in bytes of the supplied return buffer

L.11 Pathname Blocks (PNB)

The pathname block (PNB) is an alternative form of a pathname that can be used interchangeably with pathnames. Because of its structure, it can be parsed faster than a pathname. The PNB is a doubleword bounded, variable length ASCII character string which H.VOMM can distinguish from a pathname since the PNB always starts with an exclamation point.

H.VOMM provides a service to convert a pathname to a PNB. The examples which follow illustrate common pathnames and their corresponding PNB.

Example 1

```
@VOL1(DIR1)FILE1
```

Word 0	! V O L
1	blank
2	V O L 1
3	blank
4	blank
5	blank
6	! D I R
7	R O O T
8	D I R 1
9	blank
10	blank
11	blank
12	! R E S
13	blank
14	F I L E
15	1 b̸ b̸ b̸
16	blank
17	blank

Example 2

```
FILE1
```

Word 0	! V O L
1	W O R K
2	! D I R
3	W O R K
4	! R E S
5	blank
6	F I L E
7	1 b̸ b̸ b̸
8	blank
9	blank

Example 3

`(DIRECTORY)MYFILE`

Word 0	! V O L
1	W O R K
2	! D I R
3	R O O T
4	D I R E
5	C T O R
6	Y ƀ ƀ ƀ
7	blank
8	! R E S
9	blank
10	M Y F I
11	L E ƀ ƀ
12	blank
13	blank

Example 4

`@SYSTEM(SYSTEM)LOADMOD`

Word 0	! V O L
1	S Y S T
2	! D I R
3	S Y S T
4	! R E S
5	blank
6	L O A D
7	M O D ƀ
8	blank
9	blank

L.12 Post Program-Controlled Interrupt Notification Packet (PPCI)

If a task sets up a PPCI end-action receiver to check status during execution of its channel program, the status is returned in a notification packet. The address of the notification packet is contained in register three upon entering the task's PPCI end-action receiver. The notification packet is described below.

	0 7	8 15	16 23	24 31
Word 0	String forward address (NOT.SFA)			
1	String backward address (NOT.SBA)			
2	Link priority (NOT.PRI)	NOT.TYPE See Note 1.	Reserved	
3	FCB address (NOT.CODE)			
4	PSD 1 of task's PPCI receiver (NOT.PSD1)			
5	PSD 2 of task's PPCI receiver (NOT.PSD2)			
6	Number of PPCIs received since last buffer clear (NOT.STAR)		Number of status doublewords in status buffer (NOT.STAS)	
7	Address of PPCI status buffer (NOT.STAA)			
8	Address of buffer storing next status doubleword (NOT.STPT)			
9	Reserved			
10-*n*	PPCI status buffer			

Notes:

1. NOT.TYPE - Set to 1 for asynchronous notification.
2. Words 0-9 are updated by the operating system and must not be changed by the user.

L.13 Parameter Receive Block (PRB)

The parameter receive block (PRB) is used to control the storage of passed parameters into the receiver buffer of the destination task. The same format PRB is used for message and run requests. The address of the PRB must be presented when the M.GMSGP or M.GRUNP services are invoked by the receiving task.

	0 7	8 15	16 23	24 31
Word 0	Status (PRB.ST)	Parameter receiver buffer address (PRB.RBA)		
1	Receiver buffer length (PRB.RBL)		Number of bytes actually received (PRB.ARQ)	
2	Owner name of sending task, word one (PRB.OWN)			
3	Owner name of sending task, word two			
4	Task number of sending task (PRB.TSKN)			

Notes:

1. Status (PRS.ST) contains the status-value encoded status byte:

Code	Definition
0	normal status
1	invalid PRB address (PRB.ER01)
2	invalid receiver buffer address or size detected during parameter validation (PRB.RBAE)
3	no active send request (PRB.NSRE)
4	receiver buffer length exceeded (PRB.RBLE)

2. Parameter receiver buffer address (PRB.RBA) contains the word address of the buffer where the sent parameters are stored.
3. Receiver buffer length (PRB.RBL) contains the length of the receiver buffer (0 to 768 bytes).
4. Number of bytes received (PRB.ARQ) is set by the operating system and is clamped to a maximum equal to the receiver buffer length.
5. Owner name of sending task (PRB.OWN) is a doubleword that is set by the operating system to contain the owner name of the task that issued the parameter send request.
6. Task number of sending task (PRB.TSKN) is set by the operating system to contain the task activation sequence number of the task that issued the parameter send request.

L.14 Parameter Send Block (PSB)

The parameter send block (PSB) describes a send request issued from one task to another. The same PSB format is used for both message and run requests. The address of the PSB (word bounded) must be specified when invoking the M.SMSGR or M.SRUNR services, but is optional when invoking the M.PTSK service.

When a load module name is supplied in words 0 and 1 of the PSB, the operating system searches the system directory only. For activations in directories other than the system directory, a pathname or RID vector must be supplied.

When activating a task with the M.SRUNR or M.PTSK service, the value specified in byte 0 of PSB word 2 (PSB.PRI) is used to determine the task's execution priority. This value overrides the cataloged priorities of the sending and receiving tasks and the priority specified in the PTASK block. However, priority clamping is used to prevent time-distribution tasks from using this value to execute at a real-time priority, and real-time tasks from executing at a time-distribution priority. Values that can be specified in PSB.PRI are 1-64 (to be the task priority), zero (to use the base priority of the sending task), and X'FF' (to ignore the PSB priority field).

A PSB can be specified as a parameter for the M.PTSK service, along with the required task activation (PTASK) block. The PTASK block also contains a priority specification field. The PSB priority value always overrides the PTASK block priority value.

	0 7	8 15	16 23	24 31
Word 0	Load module or executable image name (PSB.LMN) or zero if activation (or task number (PSB.TSKN) if message or run request to multicopied task)			
1	Load module or executable image name, pathname vector, or RID vector if activation (or zero if message or run request to multicopied task)			
2	Priority (PSB.PRI)	Reserved	Number of bytes to be sent (PSB.SQUA)	
3	Reserved	Send buffer address (PSB.SBA)		
4	Return parameter buffer length in bytes (PSB.RPBL)		Number of bytes actually returned (PSB.ACRP)	
5	Reserved	Return parameter buffer address (PSB.RBA)		
6	Reserved	No-wait request end action address (PSB.EAA)		
7	Completion status (PSB.CST)	Processing start status (PSB.IST)	User status (PSB.UST)	Options (PSB.OPT)

Parameter Send Block (PSB)

Word 0

Bits 0-31 — Load module or executable image name — contains characters 1 through 4 of the name of the load module or executable image to receive the run request or

Task number — contains the task number of the task to receive the message or the task number of the multicopied load module or executable image to receive the run request.

Word 1

Bits 0-31 — Load module or executable image name — contains characters 5 through 8 of the name of the load module or executable image to receive the run request, or zero if the message or run request is sent to multicopied load module or executable image.

Word 2

Bits 0-7 — Contains the priority at which the receiver task is expected to be activated. Valid values are 1-64, zero, (for base priority of the sending task) and X'FF', which generates activation priority based on a combination of values that can be specified during task activation.

The following tables show how the priority of a receiver task is determined when activated with M.SRUNR or with M.PTSK.

When Activating with M.SRUNR

Send Task	Cataloged Priority of Receive task	Priority in PSB	Activates Receive task at
1-54	1-54	0	Send task cat. priority
1-54	55-64	0	55 (time-dist. clamp)
55-64	1-54	0	54 (real-time clamp)
55-64	55-64	0	Send task cat. priority
*	1-54	1-54	PSB priority
*	1-54	55-64	54 (real-time clamp)
*	55-64	1-54	55 (time-dist. clamp)
*	55-64	55-64	PSB priority
*	*	X'FF'	Receive task cat. priority

* not specified

When Activating with M.PTSK

Cataloged Priority of Send Task	Cataloged Priority of Receive task	Priority in PTASK block	Priority in PSB	Activates Receive task at
1-54	1-54	0	0	Send task cat. priority
1-54	55-64	0	0	55 (time-dist. clamp)
1-54	*	1-54	0	Send task cat. priority
1-54	*	55-64	0	55 (time-dist. clamp)
55-64	1-54	0	0	54 (real-time clamp)
55-64	55-64	0	0	Send task cat. priority
55-64	*	1-54	0	54(real-time clamp)
55-54	*	55-64	0	Send task cat. priority
*	1-54	0	1-54	PSB priority
*	1-54	0	55-64	54 (real-time clamp)
*	55-64	0	1-54	55 (time-dist.clamp)
*	55-64	0	55-64	PSB priority
*	*	1-54	1-54	PSB priority
*	*	1-54	55-64	54 (real-time clamp)
*	*	1-54	X'FF'	PTASK block priority
*	*	55-64	1-54	55 (real-time clamp)
*	*	55-64	55-64	PSB priority
*	*	55-64	X'FF'	PTASK block priority
*	*	0	X'FF'	Receive task cat. priority

* not specified

Bits 8-15	reserved
Bits 16-31	Number of bytes to be sent — specifies the number of bytes to be passed (0 to 768) with the message or run request.

Word 3

Bits 0-7	reserved
Bits 8-31	Send buffer address — contains the word address of the buffer containing the parameters to be sent.

Word 4

Bits 0-15	Return parameter buffer length — contains the maximum number of bytes (0 to 768) that may be accepted as returned parameters.
Bits 16-31	Number of bytes actually returned — set by the send message or run request service upon completion of the request.

Word 5

Bits 0-7	reserved
Bits 8-31	Return parameter buffer address — contains the word address of the buffer where any returned parameters are stored.

Word 6

Bits 0-7	reserved
Bits 8-31	No-wait request end-action address — contains the address of a user routine to be executed at a software interrupt level upon completion of the request.

Word 7

Bits 0-7 Completion status — contains completion status information posted by the operating system as follows:

Bit	Meaning if Set
0	operation in progress (PSB.OIP)
1	destination task was aborted before completion of processing for this request (PSB.DTA)
2	destination task was deleted before completion of processing for this task (PSB.DTD)
3	return parameters truncated — attempted return exceeds return parameter buffer length (PSB.RPT)
4	send parameters truncated — attempted send exceeds destination task receiver buffer length (PSB.SPT)
5	user end-action routine not executed because of task abort outstanding for this task (can be examined in abort receiver to determine incomplete operation) (PSB.EANP)
6-7	reserved

Bits 8-15	Processing start (initial) status — contains initial status information posted by the operating system as follows:

Bit	Meaning if Set
0	normal initial status (PSB.IST)
1	message request task number invalid (PSB.TSKE)
2	run request load module or executable image name not found (PSB.LMNE)
3	reserved
4	file associated with run request load module or executable image name does not have a valid load module or executable image format (PSB.LMFE)
5	dispatch queue entry (DQE) space is unavailable for activation of the load module or executable image specified by a run request (PSB.DQEE)
6	an I/O error was encountered while reading the directory to obtain the file definition of the load module or executable image specified in a run request (PSB.SMIO)
7	an I/O error was encountered while reading the file containing the load module or executable image specified in a run request (PSB.LMIO)
8	memory unavailable
9	invalid task number for run request to module or executable image in RUNW state
10	invalid priority specification. An unprivileged task can not specify a priority which is higher than its own execution priority (PSB.PRIE).
11	invalid send buffer address or size (PSB.SBAE)
12	invalid return buffer address or size (PSB.RBAE)
13	invalid no-wait mode end action routine address (PSB.EAE)
14	memory pool unavailable (PSB.MPE)
15	destination task receiver queue is full (PSB.DTQF)

Bits 16-23	User status — defined by the destination task.

Bits 24-31 Options — contains user-request control specification as follows:

Bit	Meaning if Set
24	request is to be issued in no-wait mode (PSB.NWM)
25	do not post completion status or accept return parameters. This bit is examined only if bit 24 is set. When this bit is set, the request was issued in the no call back mode. (PSB.NCBM).

L.15 Resource Create Block (RCB)

Each H.VOMM entry point that creates a permanent file, a temporary file, a memory partition, or a directory may receive a resource create block (RCB) in order to fully define the attributes of the resource that is created. RCB formats are described in the next three tables. RCBs must be doubleword bounded.

If an RCB is not supplied by the caller, the resource is created with the default attributes described in the MPX-32 Reference Manual Volume I, Chapter 4.

Permanent and Temporary File Resource Create Block (RCB)

	0 7 8 15 16 23 24 31
Word 0 1	File owner name (RCB.OWNR)
2 3	File project group name (RCB.USER)
4	Owner rights specifications (RCB.OWRI). See Note 1.
5	Project group rights specifications (RCB.UGRI). See Note 1.
6	Other's rights specifications (RCB.OTRI). See Note 1.
7	Resource management flags (RCB.SFLG). See Note 2.
8	Maximum extension increment (RCB.MXEI). See Note 3.
9	Minimum extension increment (RCB.MNEI). See Note 4.
10	Maximum file size (RCB.MXSZ). See Note 5.
11	Original file size (RCB.OSIZ). See Note 6.
12	File starting address (RCB.ADDR). See Note 7.
13	File RID buffer (RCB.FAST). See Note 8.
14	Option flags (RCB.OPTS). See Note 9.
15	Default override (RCB.FREE). See Note 10.

Notes:

1. Rights specifications are optional:

Bit	Description
0	read access allowed (RCB.READ)
1	write access allowed (RCB.WRIT)
2	modify access allowed (RCB.MODI)
3	update access allowed (RCB.UPDA)
4	append access allowed (RCB.APPN)
9	delete access allowed (RCB.DELE)

2. Resource management flags. For any bit not set, system defaults apply and, in some cases, the default is the equivalent of the bit being set (optional):

Bit	Description
0-7	resource type, equivalent to file type code, interpreted as two hexadecimal digits, 0 - FF (RCB.FTYP)
8-10	reserved
11	file EOF management required (RCB.EOFM)
12	fast access (RCB.FSTF)
13	do not save (RCB.NSAV)
14	reserved for MPX-32 usage
15	file start block requested (RCB.SREQ)
16	file is executable (RCB.EXEC)
17	owner ID set on access (RCB.OWID)
18	project group ID set on access (RCB.UGID)
19	reserved
20	maximum file extension increment is zero. System default value not used. (RCB.MXEF)
21	minimum file extension increment is zero. System default value not used (RCB.MNEF)
22	reserved
23	zero file on creation/extension (RCB.ZERO)
24	file automatically extendible (RCB.AUTO)
25	file manually extendible (RCB.MANU)
26	file contiguity desired (RCB.CONT)
27	shareable (RCB.SHAR) (owner rights spec only)
28	link access (RCB.LINK)
29-30	reserved
31	file data initially recorded as blocked (RCB.BLOK)

3. Maximum extension increment is the desired file extension increment specified in blocks (optional). Default is 64 blocks.

4. Minimum extension increment is the minimum acceptable file extension increment specified in blocks (optional). Default is 32 blocks.

5. Maximum file size is the maximum extendible size for a file specified in blocks (optional).

6. Original file size is the original file size specified in blocks (optional). Default is 16 blocks.

7. File starting address is the disk block where the file should start, if possible. If the space needed is currently allocated, an error is returned (optional).
8. File RID buffer is the address within the file creator's task where the eight word resource identifier (RID) is to be returned. If this parameter is not supplied (i.e., is zero), the RID for the created file is not returned to the creating task.
9. Option flags bits are as follows:

Bit	Description
0	owner has no access rights (RCB.OWNA)
1	project group has no access rights (RCB.USNA)
2	others have no access rights (RCB.OTNA)

10. Default override - If set, these bits override any corresponding bit set in RCB.SFLG and the system defaults (optional):

Bit	Description
0-7	must be zero
8-10	reserved
11	file EOF management not required
12	fast access not required
13	resource can be saved
14-22	reserved
23	do not zero file on creation/extension
24	file is not automatically extendible
25	file is not manually extendible
26	file contiguity is not desired
27	resource is not shareable
28-30	reserved
31	file data initially recorded as unblocked

Directory Resource Create Block (RCB)

Word	0 7 8 15 16 23 24 31
Word 0-1	Directory owner name (RCB.OWNR)
2-3	Directory project group name (RCB.USER)
4	Owner rights specifications (RCB.OWRI). See Note 1.
5	Project group rights specifications (RCB.UGRI). See Note 1.
6	Other's rights specifications (RCB.OTRI). See Note 1.
7	Resource management flags (RCB.SFLG). See Note 2.
8-10	Reserved
11	Directory original size (RCB.OSIZ). See Note 3.
12	Directory starting address (RCB.ADDR). See Note 4.
13	Directory RID buffer (RCB.FAST). See Note 5.
14	Option flags (RCB.OPTS). See Note 6.
15	Default override (RCB.FREE). See Note 7.

Notes:

1. Rights specifications bits are as follows:

Bit	Description
0	read access allowed (RCB.READ)
8	directory may be traversed (RCB.TRAV)
9	directory may be deleted (RCB.DELE)
10	directory entries may be deleted (RCB.DEEN)
11	directory entries may be added (RCB.ADEN)

2. Resource management flags are optional:

Bit	Description
13	do not save (RCB.NSAV)
27	shareable (RCB.SHAR)

3. Directory original size is the number of entries required (optional).
4. Directory starting address is the disk block number where the directory should start, if possible. If the space needed is currently allocated, an error is returned (optional).
5. Directory RID buffer is the address within the directory creator's task where the eight word resource identifier (RID) is to be returned. If this parameter is not supplied (i.e., is zero), the RID for the created directory is not returned to the creating task.

6. Option flags are as follows:

Bit	Description
0	owner has no access rights (RCB.OWNA)
1	project group has no access rights (RCB.USNA)
2	others have no access rights (RCB.OTNA)

7. If default override is set, these bits override any corresponding bit set in RCB.SFLG and the system defaults (optional).

Bit	Description
0-7	must be zero
13	resource can be saved
27	resource is not shareable

Memory Partition Resource Create Block (RCB)

	0 7 8 15 16 23 24 31
Word 0-1	Partition owner name (RCB.OWNR)
2-3	Partition project group name (RCB.USER)
4	Owner rights specifications (RCB.OWRI). See Note 1.
5	Project group rights specifications (RCB.UGRI). See Note 1.
6	Other's rights specifications (RCB.OTRI). See Note 1.
7	Resource management flags (RCB.SFLG). See Note 2.
8-9	Reserved
10	Starting word page number (RCB.PPAG)
11	Partition original size (RCB.OSIZ). See Note 3.
12	Partition starting address (RCB.ADDR). See Note 4.
13	Partition RID buffer (RCB.FAST). See Note 5.
14	Option flags (RCB.OPTS). See Note 6.
15	Default override (RCB.FREE). See Note 7.

Notes:

1. Rights specifications are optional:

Bit	Description
0	read access allowed (RCB.READ)
1	write access allowed (RCB.WRIT)
9	delete access allowed (RCB.DELE)

2. Resource management flags are optional:

Bit	Description
13	do not save (RCB.NSAV)

3. Partition's original size is the number of protection granules required.
4. Partition's starting address is a 512-word protection granule number in the user's logical address space where the partition is to begin.
5. Partition's RID buffer is the address within the partition creator's task where the eight word resource identifier (RID) is to be returned. If this parameter is not supplied (i.e., is zero), the RID for the created partition is not returned to the creating task.
6. Option flags are optional:

Bits	Description
0	owner has no access rights (RCB.OWNA)
1	project group has no access rights (RCB.USNA)
2	others have no access rights (RCB.OTNA)
9	defines a static partition (RCB.STAT)
24-31	define memory class (RCB.MCLA). Values are:

Value	Memory Class
0	S (default)
1	E
2	H
3	S

7. If set, these bits override any corresponding bit set in RCB.SFLG and the system defaults (optional):

Bits	Description
0-7	must be zero
13	resource can be saved

L.16 Resource Identifiers (RID)

The fastest means of locating a volume resource (once created) is by its resource identifier (must be on a doubleword boundary). The resource identifier has the following format:

	0 7 8 15	16 23 24 31
Word 0-3	Volume name	
4	Creation date	
5	Creation time	
6	Volume address of resource descriptor	
7	Must contain zero	Resource type

Since the resource identifier contains the volume address of the resource descriptor, the resource descriptor (which points to and describes the resource) can be accessed directly without going through the various directories which would otherwise have to be traversed.

Given a valid pathname defining a resource, the corresponding resource descriptor may be retrieved by the H.VOMM locate resource service. The first eight words of a resource descriptor consist of the resource identifier.

L.17 Resource Logging Block (RLB)

The resource logging block (RLB) is a word-bounded data structure used to pass information between H.VOMM and the caller. The information is used to locate a directory entry and resource descriptor for a single resource or for all resources defined in a particular directory.

	0 7	8 15 16 23 24 31
Word 0	Pathname vector or RID address (RLB.TGT)	
1	Resource directory buffer address (192W) (RLB.BUFA). See Note 1.	
2	Associated mounted volume table entry address (RLB.MVTE)	
3	Parent directory RD block address (RLB.RDAD)	
4	Type (RLB.TYPE). See Note 2.	Buffer offset (RLB.BOFF)
5	Length. See Note 3.	Return buffer address (RLB.DIRA)
6	User FCB address (RLB.FCB)	
7	Flags. See Note 4.	Reserved (RLB.INT)

Notes:

1. Optional. If not specified, a resource directory is not returned.

2. Bits in RLB.TYPE are assigned as follows:

Bits	Meaning if Set
0	indicates recall (RLB.RECA)
1-7	reserved

3. This word contains the address of a buffer and its length in words (the buffer can be up to 16 words long).

4. Bits in the flags byte are assigned as follows:

Bits	Meaning if Set
0-1	reserved
2	directory entry and resource descriptor for specified directory are returned
3	root directory
4	resource is located
5-7	reserved

L.18 Resource Requirement Summary (RRS) Entries

The resource requirement summary (RRS) is a doubleword bounded data structure used to identify the resources required by a task to the resource manager. Resources are statically allocated using the information in the RRS entry. The RRS is generally built by processors requiring static allocation of resources, such as TSM, cataloger, etc., or supplied as an argument for dynamic allocation.

For compatibility purposes, revision 1.*x* RRS formats can be used. The details of these formats can be found in Chapter 2 of a revision 1.*x* Technical Manual.

Type 1 - Assign by Pathname

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Plength (RR.PLEN)	Reserved. See Note 2.
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4-*n*	Pathname (variable length) (RR.NAME1)			

Type 2 - Assign to Temporary File

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Initial file size (RR.PLEN)	
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4-7	Volume name (16 characters; left-justified, blank-filled) (RR.NAME1) (Volume name is optional)			

Type 3 - Assign to Device

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Density (RR.DENS). See Note 5.	Zero
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4	Device type (RR.DT3). See Note 6.	Volume number (RR.VLNUM)	Channel number See Note 7. (RR.CHN3)	Subchannel number (RR.SCHN3)
5	Unformatted ID (1-4 characters) (RR.UNFID)			

Type 4 - Assign to LFC

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Zero	
2	Zero	Logical file code (RR.SFC)		
3	Options (RR.OPTS). See Note 4.			

Type 5 - Assign by Segment Definition

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	UDT index (RR.UDTI)	Reserved
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4	Starting block number (RR.STBLK)			
5	Number of blocks (RR.NBLKS)			

Resource Requirement Summary (RRS) Entries

Type 6 - Assign by Resource ID

	0 — 7	8 — 15	16 — 23	24 — 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Zero	Reserved
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4-7	Volume name (16 characters; left-justified, blank-filled) (RR.NAME1)			
8	Binary creation date (RR.DATE)			
9	Binary creation time (RR.TIME)			
10	Resource descriptor block address (RR.DOFF)			
11	Reserved		Resource type (RR.RTYPE)	

Type 7 - Reserved for Future Use

Type 8 - Reserved for Future Use

Type 9 - Mount by Device Mnemonic

	0 — 7	8 — 15	16 — 23	24 — 31
Word 0	Zero	System ID (RR.SYSID). See Note 11.		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Zero	
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4-7	Volume name (16 characters; left-justified, blank-filled) (RR.NAME1)			
8	Device type (RR.DT9). See Note 8.	Reserved	Channel number (RR.CHN9). See Note 9.	Subchannel number (RR.SCHN9)
9	Zero			

Type 10 - Assign to ANSI Tape

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Format (RR.FORM)	Protect (RR.PROT)
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4	Record length (RR.RECL)		Block size (RR.BSIZE)	
5	Generation number (RR.GENN)			
6	Generation version number (RR.GENV)			
7	Absolute termination date (RR.EXPIA)			
8	Relative termination date (RR.EXPIR)		Logical volume identifier (RR.LVID)	
9	RR.LVID (cont.)			
10-13	17-character file identifier (RR.AFID)			
14	RR.AFID (cont.)	Reserved		
15	Reserved			

Type 11 - Assign to Shadow Memory

	0 7	8 15	16 23	24 31
Word 0	Zero			
1	Type (RR.TYPE) See Note 1.	Size (RR.SIZE)	Shadow flags (RR.SHAD). See Note 10.	
2	Start address (RR.SADD)			
3	End address (RR.EADD)			

Notes:

1. Bits in RR.TYPE are assigned as follows:

Value	Meaning
1	assign by pathname (RR.PATH)
2	assign to temporary file (RR.TEMP)
3	assign to device (RR.DEVC)
4	assign to secondary LFC (RR.LFC2)
5	assign to segment definition (RR.SPACE)
6	assign by resource ID (RR.RID)
7	reserved for future use
8	reserved for future use
9	mount by device mnemonic (RR.MTDEV)
10	assign to ANSI labeled tape (RR.ANS)
11	assign to shadow memory (RR.SHTYP)
12-255	reserved

2. Byte 3 is zero. This field is used by MPX-32 for big blocking buffers.
3. Bits in RR.ACCS are assigned as follows:

Bits	Meaning if Set
0	read access allowed (RR.READ)
1	write access allowed (RR.WRITE)
2	modify access allowed (RR.MODFY)(not valid for ANSI tapes)
3	update access allowed (RR.UPDAT)
4	append access allowed (RR.APPND)
5-15	reserved
16	explicit shared use requested (RR.SHAR)
17	exclusive use requested (RR.EXCL)
18	assign as volume mount device (RR.MNT)
19-31	reserved

4. Bits in RR.OPTS are assigned as follows:

Bits	Meaning if Set
0	treat as SYC file (RR.SYC) (TSM/JOB only)
1	treat as SGO file (RR.SGO) (TSM/JOB only)
2	treat as SLO file (RR.SLO)
3	treat as SBO file (RR.SBO)
4	explicit blocked option (RR.BLK)
5	explicit unblocked option (RR.UNBLK)
6	inhibit mount message (RR.NOMSG)
7	reserved for system use
8	automatic open requested (RR.OPEN)
9	user-supplied blocking buffer address in FCB (RR.BUFF)
10-11	reserved for system use
12	mount with no-wait (RR.NOWT)
13	mount as public volume (RR.PUBLC)
14	set by H.VOMM for special case handling of VOMM assignments (RR.VOMM)
15	file is spooled when deallocated (RR.SEP)
16	ANSI labeled tape on RRS type 3 (RR.ANSI)
17-31	reserved

5. RR.DENS contains the density specification for XIO high speed tape units. When specified, this field has the following bit significance:

Bits	Meaning if Set
0	indicates 800 bpi nonreturn to zero inverted (NRZI)
1	indicates 1600 bpi phase encoded (PE)
6	indicates 6250 bpi group coded recording (GCR)

If this field is zero, 6250 BPI is set by default.

6. RR.DT3 specifies whether or not a channel is present and specifies the device type:

Bits	Meaning if Set
0	channel present
1-7	device type

7. RR.CHN3 specifies whether or not a subchannel is present and specifies the channel number:

Bits	Meaning if Set
0	subchannel is present. Examined only if bit zero of RR.DT3 is set.
1-7	channel number

8. RR.DT9 specifies whether or not a channel is present and specifies the device type:

Bits	Meaning if Set
0	channel present
1-7	device type

9. RR.CHN9 specifies whether or not a subchannel is present and specifies the channel number:

Bits	Meaning if Set
0	subchannel is present. Examined only if RR.DT9 is set.
1-7	channel number

10. RR.SHAD contains the shadow flags that qualify the start and end addresses, or specify what portions of the task are to be shadowed:

Bits	Meaning if Set
0-7	reserved
8	shadow the task (RR.SHTSK)
9	shadow the TSA (RR.SHTSA)
10	shadow the stack (RR.SHST)
11	shadow memory is required (RR.SHRQ)
12	shadow the entire task (RR.SHALL)
13	absolute address (RR.ABS)
14	relative to the code section origin (RR.CREL)
15	relative to the data section origin (RR.DREL)

11. RR.SYSID is the ID for mounting a multiprocessor volume. Valid IDs are:

Multiported (MP)	0 through F
Dual Ported (DP)	0 or 1

For more information on mounting multiprocessor volumes see the MPX-32 Reference Manual Volume I, Chapter 4, Mounting Multiprocessor Volumes.

L.19 Receiver Exit Block (RXB)

The receiver exit block (RXB) is used to control the return of parameters and status from the destination (receiving) task to the task that issued the send request. It is also used to specify receiver exit options. The same format RXB is used for both messages and run requests. The address of the RXB must be presented as an argument when either the M.XMSGR or M.XRUNR services are called.

	0 7	8 15	16 23	24 31
Word 0	Return status (RXB.ST)	Return parameter buffer address (RXB.RBA)		
1	Options (RXB.OPT)	Reserved	Number of bytes to be returned (RXB.RQ)	

Notes:

1. Return status (RXB.ST) contains status as defined by the receiver task. Used to set the user status byte in the parameter send block (PSB) of the task which issued the send request.
2. Return parameter buffer address (RXB.RBA) contains the word address of the buffer containing the parameters which are to be returned to the task which issued the send request.
3. Options (RXB.OPT) contains receiver exit control options. It is encoded as follows:

Value	Exit Type	Meaning
0	M.XRUNR	wait for next run request.
	M.XMSGR	return to point of task interrupt.
1	M.XRUNR	exit task, process any additional run requests. If none exist, perform a standard exit.
	M.XMSGR	N/A

4. Number of bytes to be returned (RXB.PQ) contains the number of bytes (0 to 768) of information to be returned to the sending task.

L.20 Type Control Parameter Block (TCPB)

The type control parameter block (TCPB) allows I/O to and from the system console by setting up task buffer areas for messages output by a task and optional reads back from the console. If no input is desired, word one of the TCPB must be zero.

See the MPX-32 Reference Manual Volume I, Chapter 5 for further details on the TCPB.

	0 11	12	13 31
Word 0	Output quantity (TCP.OQ)	See Note 1.	Output data address (TCP.OTCW)
1	Input quantity (TCP.IQ)	See Note 1.	Input data address (TCP.ITCW)
2	Console Teletype Flags (TCP.FLGS). See Note 2.		

Notes:

1. Bit 12 is set to 1.
2. Bits in TCP.FLGS are assigned as follows:

Bits	Meaning if Set
0	no-wait I/O
31	operation in progress. This bit is reset after post-I/O processing completes.

Type Control Parameter Block (TCPB)

Type Control Parameter Block (TCPB) using 24-bit address:

	0 7	8 15 16 23 24 31
Word 0	Output quantity (TCP.OQ)	Output data buffer address (TCP.OTCW)
1	Input quantity (TCP.IQ)	Input data buffer address (TCP.ITCW)
2	Console device flags (TCP.FLGS) See Note 1.	

Notes:

1. Bit interpretations for TCP.FLGS are:

Bits	Meaning if Set
0	no-wait I/O
1	data buffer addresses are 24-bit addresses (TCP.LAD) Note: This bit must be set.
31	operation in progress. This bit is reset after post-I/O processing completes.

L.21 Unit Definition Table (UDT)

The unit definition table (UDT) is a system resident structure that identifies device-dependent information required by a handler for a specific device. The UDT is built by the SYSGEN process, one for each device configured in the system. During SYSGEN, each UDT is linked to its corresponding controller definition table (CDT) and its associated controller and handler.

	0 7	8 15	16 23	24 31
Word 0	UDT index (UDT.UDTI)		CDT index (UDT.CDTI)	
1	Unit status (UDT.STAT). See Note 1.	Device type code (UDT.DTC). See Note 2.	Logical channel number (UDT.CHAN)	Logical subaddress (UDT.SUBA)
2	Reserved	Address of dispatch queue entry of task which has device allocated if device is not shared (UDT.DQEA)		
3	Physical channel number (UDT.PCHN)	Physical subaddress (UDT.PSUB)	Sectors per block (UDT.SPB) or number of characters per line (UDT.CHAR). See Note 3.	Sectors per allocation unit (UDT.SPAU) or number of lines per screen (UDT.LINE). See Note 4.
4	Flags (UDT.FLGS). See Note 5.	Number of sectors per track on disk or global line counter if a terminal (UDT.SPT)	Maximum byte transfer (UDT.MBX)	
5	Number of sectors on disk or tab setting if a terminal (UDT.SECS)			
6	Sector size, on disk or a tab setting if a terminal (UDT.SSIZ)		Number of heads on disk or a tab setting if a terminal (UDT.NHDS)	
7	Serial number if tape or removable disk (UDT.SERN). See Note 6.			
8	Peripheral time-out value (UDT.PTOV)			
9	Reserved	Address of device context area (UDT.DCAA) or handler name at initialization (UDT.HNAM)		
10	Bit flags (UDT.BIT2). See Note 7.		Associated allocated resource table index if assigned (UDT.ARTI)	
11	Service interrupt handler address (UDT.SIHA)			
12	Reserved (UDT.CXR) See Note 8.	Secondary flags (UDT.BIT3) See Note 9.	Reserved (UDT.SHFL)	Reserved (UDT.DQEN)
	or UDT.HIST. See Note 10			
13	Address of first IOQ linked to this device (UDT.FIOQ)			
14	Address of last IOQ linked to this device (UDT.BIOQ)			
15	Link Priority (UDT.LPR1)	Link Count (UDT.IOCT)	Unit Status byte 2 (UDT.STA2). See Note 11.	

Notes:

1. Bits in UDT.STAT are assigned as follows:

Bit	Meaning if Set
0	online (UDT.ONLI)
1	dual-portd XIO disk (UDT.DPDC)
2	allocated (UDT.ALOC)
3	terminal in use and not in wait (UDT.USE)
4	system output unable to allocate (UDT.NOAL)
5	shared device (UDT.SHR)
6	premounted (UDT.PREM)
7	terminal (TSM) device (UDT.TSM)

2. For example, 01 for any disk, 04 for any tape, etc. Valid device type codes are listed in Appendix A.
3. For disks, contains the number of sectors per block (UDT.SPB). For terminals, contains the number of characters per line (UDT.CHAR).
4. For disks, contains the number of sectors per allocation unit (UDT.SPAU). For SLO or terminals, contains the number of lines per page or screen (UDT.LINE).
5. Bits in UDT.FLGS are assigned as follows:

Bit	Meaning if Set
0	extended I/O device (UDT.FCLS)
1	I/O outstanding (UDT.IOUT)
2	removable disk pack (UDT.RMDV)
3	a break has been requested for this device (UDT.LOGO)
4	autoselectable for batch SLO (UDT.BSLO)
5	autoselectable for batch SBO (UDT.BSBO)
6	autoselectable for real-time SLO (UDT.RSLO)
7	autoselectable for real-time SBO (UDT.RSBO)

6. If the device is a terminal or console, the first halfword is the current terminal type for TERMDEF (UDT.CTDF) and the second halfword is the default terminal type (UDT.DTDF).
7. Bits in UDT.BIT2 are assigned as follows:

Bits	Meaning if Set
0	port is private; else switched (UDT.DIAL)
1	port is connected to modem (UDT.MODM)
2	port has graphic capability (UDT.GRFC)
3	port is full duplex (UDT.FDUX)
4	port is configured multidrop (UDT.MDRA)
5	volume mounted on device (UDT.VOL)
6	echo by computer (UDT.ECHO)
7	device has failed. Log off TSM (UDT.DEAD)
8	cache device (UDT.CAC)
9	inhibit automatic line wrap (UDT.NRAP)
10	spool device requires form feed after printing rather than before; initial form feed is inhibited (UDT.FEOP)

Bits	Meaning if Set
11	quarter inch cartridge tape drive (UDT.QITD)
12	software read flow control required (UDT.RXON)
13	software write flow control required (UDT.WXON)
14	hardware read flow control required (UDT.RHWF)
15	hardware write flow control required (UDT.WHWF)

8. For switched port, contains the value specified in the LOGONFLE CXR = option (UDT.CXR)
9. Bits in UDT.BIT3 are assigned as follows:

Bits	Meaning if Set
0	SCSI device (UDT.SCSI)
1-7	reserved

10. UDT.HIST is used as an address save area by pseudo device handlers, such as ON.IPXIO
11. Bits in UDT.STA2 are assigned as follows:

Bits	Meaning if Set
0	IOQ linked from UDT (UDT.IOQ)
1	IOP device (initialized by SYSGEN) (UDT.IOP)
2	device malfunction (UDT.MALF)
3	operator intervention applicable (UDT.INTV)
4	use standard XIO interface
5	floppy disk
6	cartridge module drive
7	moving head disk with fixed head option
8	if software read flow control enabled, use DTR line; otherwise, use RTS line. (UDT.RDTR)
9	memory disk (UDT.MD) or valid command line recall and edit device (UDT.CLRE)
10	memory allocated for memory disk (UDT.MDAL)
11	start address of memory disk specified at SYSGEN (UDT.MDST)
12	multiport device is shared with an MPX-32 Revision 3.2C or earlier version (UDT.PPV)
13	device is exclusive ANSI (UDT.ANSI)
14	serial printer (UDT.SLPR)
15	port is switched and CXR=N option has been specified (UDT.DCXR)

Glossary

access method	A software package that provides the ability to access fields within records, to classify or order records according to the contents of fields, and to perform other such functions.
access mode	Defines the range of operations to be performed on a resource.
aged page	A page which has not been referenced within a predetermined frame of time during demand page processing. This page is no longer considered a part of the task's working set.
allocated resource table (ART)	A system resident table with an entry for each currently allocated resource in the system.
allocation	The process of securing a resource for a specific usage and access mode for a task.
allocation unit	A mechanism for grouping more than one block on a formatted disc, or other mass medium, at one time. Usually specified in multiples of 192-word disc blocks. See disc block.
argument	A value (string or integer) that is assigned to a parameter.
assign	To associate a resource with a logical file code used by a process.
assignment	The process of associating a logical file code with a system resource. Does not guarantee the resource for a specific use or access mode for a task.
asynchronous	Implies one entity does not wait for or otherwise acknowledge another entity when it performs an operation.
asynchronous notification	A process does not stop execution waiting for notification. It receives a software interrupt when an asynchronous operation is complete.
base mode	Implies the base register instruction set that allows executable programs of up to 4096KW (16MB).
blocked I/O	The process of packing records equal to or less than 254 bytes so that more than one record is stored in a 192-word disc block.
blocking buffers	Buffers used for packing records for blocked I/O. See blocked I/O.

caller notification packet (CNP)	A structure used to supply additional calling parameters and to control the handling of abnormal conditions that may occur during resource requests.
classes of users	A three-level grouping of users into OWNER, PROJECTGROUP and OTHER. Used to permit or limit access to a resource by 'class'.
command file	A file containing commands known to a particular operating system or process.
CONCEPT/32	A term which implies the entire line of CONCEPT/32 computers; for example, the 32/67.
configuration	Hardware: the physical hardware related to a CPU. Software: adapting the operating system to a hardware configuration with the SYSGEN processor.
data files	Files containing data or transactions that have been processed or will be processed by a task.
data management	The ability to structure data into records using buffers.
Datapool	An area of memory that contains the same functionality as Global Common but with the added flexibility of symbolic references being independent of the actual positioning of data within the memory area. See Global Common.
deallocate	To detach a resource from a process.
deassign	To remove the association between a logical file code and a resource and deallocate the resource.
dequeue	To remove from a prioritized list.
demand page	Allocation of memory when the logical page is referenced by the task on demand. The process of allocating physical memory when pages are referenced and deallocating physical memory when pages are no longer active. Pages that are no longer active are considered aged and removed from the task's working set.
device	A peripheral unit such as a card reader, a printer, a disc drive, or a tape drive. Distinguished from media used with devices.
device access	Levels are physical I/O, logical device I/O, and logical file I/O.
device-dependent I/O	Tasks perform operations to a specified device with minimal IOCS overhead.
device-independent I/O	Tasks perform I/O operations through the use of operating system calls which are independent of the device used to perform the operation.

direct I/O	Tasks perform operations bypassing IOCS and handler functions by coding its own handler and attaching it to a specific channel.
directory	A list of file names and/or memory partition names. Stored on disc like a regular file. Located via a resource descriptor for the directory. Directory names are 1 to 16 characters in length and valid characters for names are A to Z, 0 to 9, dot (.) and underscore (_).
directory descriptor	The resource descriptor for a directory.
disc block	A common unit of measurement (some number of words) used to measure file space on formatted media throughout a system. The number of words in a block is oriented to the most common sector size on discs used with the system.
DMAP	See resource descriptor allocation map.
dynamic assignment	The association of a logical file code with a system resource during task execution.
enqueue	To put into a list ordered by software priority.
exclusive use	A resource is not available for use by any other task until that resource is deallocated by the using task. Guarantees access to a resource, within the access limitations imposed by the resource creator, when logical I/O is initiated.
executable image	A file of object code produced by the LINKER/X32.
explicit shared use	A resource can be used concurrently by more than one task. Each task maintains resource integrity by establishing its own synchronization and locking mechanisms. Each task is guaranteed access to the resource, within access limitations imposed by the resource creator, when logical I/O is initiated.
extended code	That part of the operating system that has been modified to run in the extended execution space.
extended file control block	A file control block set up by the user which contains more information than the standard file control block. See file control block.
file	A set of information stored on a mass medium such as disc or tape that is given a unique identity (number and often name) and treated as a single entity for processing.
file control block (FCB)	Set up by the user to describe logical files within a task. Describes attributes of logical I/O operation.
file descriptor	A resource descriptor for a file.

file identifier	A unique identifier stored in the resource descriptor for a file when the file is created. Used to access the resource descriptor without a directory search.
file segment	Set of contiguous allocation units on a volume identifying the space associated with a file. Each file segment definition contains the absolute 192-word block volume segment address and the segment length in 192-word blocks (maximum of 32 file segment definitions per file).
file space allocation map (SMAP)	A bit map used to allocate space on a volume.
filename	A 1- to 16-character name supplied for a permanent file when it is created on a mass medium. Used in most cases thereafter to identify the file. Valid characters for filenames are the upper-case letters A to Z, the decimal numbers 0 to 9, and the special characters dot (.) and underscore (_). Filenames to be used with the compatible interfaces, for example Editor, File Manager, and Media, are limited to 1 to 8 characters.
format	Standard organization of information.
formatted volume	A disc pack or floppy disc that contains standard volume system structures established by the Volume Formatter utility.
Global Common	An area of memory accessible by using symbolic names to identify specific storage cells. Programs belonging to many independent tasks can freely access the same data and exchange control information within the Global Common area.
implicit shared use	A resource is available for concurrent use by other tasks in a compatible access mode. Does not guarantee access when logical I/O is initiated. Resource integrity is automatically maintained by the system.
job file	A command file designed to run in the batch or interactive environment.
library file	Object modules or source modules identified by name that are output to a single file. Modules on library files can be used separately and repeatedly. For example, object modules can be retrieved by name during cataloging and inserted with existing code. The ability to edit the contents of library files by name is also normally available.
load module file	A file of object code produced by the Cataloger that is ready to relocate from disc into memory and execute as a process. Load module files can be activated by name and are controlled by name or task number.

logical device I/O	I/O where the physical characteristics of a device are not determined automatically by the file management system (device and data formatting are inhibited), allowing the user to exert control over a particular physical device or device medium.
logical dismount	The action taken by MPX-32 to disassociate a volume from the requesting task. A TSM logical dismount disassociates the volume from the requesting context.
logical file code (LFC)	User defined 1- to 3-character ASCII codes identifying logical files within tasks.
logical file I/O	I/O where the physical characteristics of a device and device medium (device format control, data conversion, data formatting) are performed automatically for the user so that he gains a degree of device independence.
logical mount	The action taken by MPX-32 to associate a physically mounted volume to a task. A TSM logical mount associates the volume to the TSM context requesting the mount.
logical resource	Any entity existing only because of a mechanism provided by software. The primary logical resources are: disc volumes, directories, files, and memory partitions.
map block	A 2048-word unit of memory allocation. In demand page processing, a page is a map block.
medium (singular) media (plural	A contiguous source of input or output that is used for a particular peripheral device. For example, a disc pack is the medium mounted on a disc drive; a tape is the medium mounted on a tape drive; paper is the medium used on a printer; a deck of cards is the medium used on a card reader. The operating system distinguishes use of media from use of devices.
memory descriptor	The resource descriptor for a memory partition.
memory partitions	Named areas of physical memory that can be shared by concurrently executing tasks.
modular	Construction in independent layers. Each higher level layer builds on the layer beneath it and provides its own standard interfaces to the levels above and below it.
mounted volume table (MVT)	A system resident table with an entry for each physically mounted volume. Each entry contains information used by the system to maintain volume accounting information.

multicopied tasks	Tasks with the same name and the same concurrent load module activity, owned by a single owner or several owners. This is accomplished by cataloging a task as multicopy. Task numbers must be used to communicate with multicopied tasks. See task number.
multiprocessor volume	A specially mounted user volume that allows tasks operating in separate system environments to concurrently access any volume resource.
multivolume magnetic tape	A set of 1 through 255 maximum physical reels of magnetic tape processed as a continuous reel.
nonbase mode	Implies the nonbase register instruction set which allows executable programs of up to 128KW.
nonpublic volume	A volume assigned specifically to the tasks that mount it. Remains physically mounted until use and assign counts equal 0.
object file	A file of assembled or compiled code that can be cataloged or linked into a task.
owner	The user who has possession of and can control access to a file, device, memory partition, or directory. Usually the owner of a resource is the user who created its resource descriptor.
owner name	A 1- to 8-character name supplied at logon which remains unchangeable through logoff. The following characters cannot be used in owner names: blanks, commas, semicolons, equal signs, line feeds, dollar signs, percent signs, exclamation points, and left or right parentheses. All other characters are valid. Owner names are associated with any task or process activated on the system and noted by any process that acts in the owner's behalf. Owner name is also associated with any resources a user creates unless the user specifies otherwise. Specifying a different owner when creating a resource definition does not change the user's owner name; it only specifies the owner name associated with the resource.
page	A 512-word unit of memory protection. Also referred to as a protection granule. Four pages compose a map block. For demand page processing, a page is a map block brought into memory and removed from memory during the life of a demand page task.
page fault	The reference of a page within the logical address space which is not currently a part of the task's working set.

page in	Bringing into logical memory a page needed to satisfy an address referenced by a task.
page out	The removal of aged pages from the task's logical address space.
parameter	A symbolic name in a process or directive file that can be assigned an argument.
pathname	Variable length ASCII character strings which uniquely identify a volume resident resource by explicitly or implicitly describing the volume, one or more directories, and the resource name.
pathname block	Doubleword bounded variable length ASCII character string beginning with "!" which uniquely identifies a volume resident resource by explicitly or implicitly describing the volume, one or more directories, and the resource name.
permanent files	Files that remain defined on a volume until explicitly deleted.
physical dismount	The action taken by MPX-32 to disassociate a volume from an assigned mount device and deallocate the device.
physical mount	The action taken by MPX-32 to allocate a mount device and associate that device to the assigned volume name.
physical resource	Any physical hardware that MPX-32 supports. Tasks access the resource to perform their functions. The primary physical resources are: the CPU, computer memory (main storage), and input/output devices.
portable	Can be used on any compatible device in a single system configuration. Can also be carried to a compatible device on a different system hardware configuration. Usually describes a volume.
post program-controlled interrupt receiver	User supplied end-action receiver entered when a hardware post program-controlled interrupt is encountered.
process	A body of code scheduled for CPU time as a single entity. A load module is a process, in loadable form, stored on disc. Same as task.
project group name	A name that is specified at logon and can also be changed. Identifies a group of users that have a defined set of rights when they access a resource.
protect	To limit access to a resource. See classes of users.
protection granule	A 512-word unit of memory protection. Also referred to as a page in a non-demand page context. Four protection granules compose a map block.

public volume	A volume available for resource assignments by all tasks activated in the system.
real time task	Synonymous with time critical process.
requestor	The process which requests a function. Each process on a system has an associated owner name. The system process that requests a function for a user (e.g., in the interactive environment) keeps track of the owner name so that the user thinks of himself as the 'requestor'.
resource	Any source of support that exists external to a task and that the task needs to perform its function. A resource can be physical or logical.
resource create block (RCB)	Defines access attributes for permanent files, temporary files, memory partitions, and directories when the particular resource is created. If not supplied at resource creation, system default attributes are assumed.
resource descriptor (RD)	Contains access, accounting, and space definition information pertaining to mounted volume resources, permanent files, temporary files, directories and partitions.
resource descriptor allocation map (DMAP)	A bit map used for the allocation of resource descriptors on a volume.
resource identifier (RID)	The fastest way to locate an already created volume resource. The RID is in the first eight words of a resource descriptor and contains the volume address of the resource descriptor, which points to and describes the resource.
resource logging block (RLB)	A parameter block used as input to the M.LOGR service for logging resources.
Resource Management Module (H.REMM)	Performs allocation and assignment of all system resources and maintains access compatibility and usage rights for these resources. Also contains synchronization mechanisms for concurrent access to shared resources.
resource requirement summary (RRS)	Defines assignment requirements of a resource. Entries are variable length, doubleword bounded. There are 9 types of entries.
root directory	The directory of all directories defined on a volume.
SMAP	See file space allocation map.

source file	A file of source code to be assembled or compiled into object code.
static assignment	The association of a logical file code with a system resource during task activation.
status posting	The process of returning information that indicates whether a service was completed successfully, with errors, or denied.
swap volume	A volume used as the primary medium for swap file allocations.
symbolic	A representation of a physical resource, e.g., a name that represents an entity but is not the entity itself.
synchronous notification	A process waits on further processing until it is notified that an operation is done or that there is something inhibiting the operation (e.g., a resource is not available or other processes are in contention for the resource).
system administrator attribute (SA)	Gives an unprivileged user the ability to execute privileged SVC's, allows a user to mount public volumes, and allows a user to change his owner name. A user with the system administrator attribute is, however, restricted to resource access limitations imposed by the resource creator.
system directory	Special directory on the system volume which contains volume resources necessary for system operation.
system volume	A volume containing the system and bootstrap images from which the current system was IPLed. This volume is automatically mounted by the SYSINIT task at system initialization.
task	Synonymous with process.
task name	The name supplied when a task is cataloged or linked.
task number	An 8-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.
temporary files	Unnamed files that are referenced by resource identifiers. They are automatically deleted from the system and their volume space made available when the last task assigned to them terminates execution.
time critical process	A process which has time constraints. Same as a real time task.
traverse	To pass through a directory on the way to another directory or resource.

type control parameter block (TCPB)	Set up by the user for sending and receiving messages to/from the system console.
unformatted media	A medium (magnetic tape, disc pack or floppy disc) that does not contain valid volume format information, but must be mounted before initiation of I/O operations.
usage mode	Defines the degree to which multiple tasks can concurrently allocate a resource. Usage modes are: exclusive use, explicit shared, and implicit shared.
user	A person who uses a system. Processes and commands that activate processes are either initiated by a user or initiated on behalf of a user.
volume	A medium that has a standard format. Disc packs can be formatted as volumes.
volume assignment table (VAT)	A task resident table with an entry for each non-public volume currently assigned to the task.
Volume Management Module (H.VOMM)	Manipulates volume resident and related memory resident structures in order to allow for creation, deletion, and maintenance of user and system resources which reside on volumes; for example, provides space management for all currently mounted volumes in the system.
working set	The pages (map blocks) of a task that are actively being referenced within a predetermined frame of time.

Index

MPX-32™ Utilities & Processors
Reference Manual
No. 323-001552-600

READER'S
COMMENT
FORM

Please use this form to communicate your views about this manual. The form is preaddressed and stamped for your convenience.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy	____	____	____	____
Clarity	____	____	____	____
Completeness	____	____	____	____
Examples	____	____	____	____
Figures	____	____	____	____
Index	____	____	____	____
Organization	____	____	____	____
Retrievability of Information	____	____	____	____

Additional comments:

If you wish a reply, please print your name and mailing address:

What is your occupation/title?

Thank you for your cooperation.

Note: Copies of Encore publications are available through your Encore representative or the customer service office serving your locality.

FOLD HERE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 2356 FORT LAUDERDALE, FL

POSTAGE WILL BE PAID BY ADDRESEE

ENCORE COMPUTER CORPORATION
ATTENTION: DOCUMENTATION COORDINATOR
6901 W. SUNRISE BLVD.
P.O. BOX 409148
FT. LAUDERDALE, FL 33340-9970

FOLD HERE

UT ALONG LINE