MPX-32 ™

Unsupported Software

Revision 3.5

Reference Manual Volume IV

April 1990

323-001554-600

ENCORE

Limited Rights

Proprietary Information

Restricted Rights

Encore Computer Corporation
6901 West Sunrise Boulevard
Fort Lauderdale, Florida 33313

History

The MPX-32 Software Release Notes, Publication Order Number **324-004400-100**, (Release 3.2C) was printed December, 1985.

The attachments were removed from the MPX-32 Software Release Notes to form Volume IV of the MPX-32 Release 3.3 Reference Manual, Publication Order Number **323-001554-300**, (Revision 3, Release 3.3), printed December, 1986.

Publication Order Number **323-001554-400**, (Revision 4, Release 3.4) was printed January, 1988.

Publication Order Number **323-001554-500**, (Revision 3.4U03) was printed October, 1989.

Publication Order Number **323-001554-600**, (Revision 3.5) was printed April, 1990.

This manual contains the following pages:

Title page
Copyright page
iii/iv through xi/xii
1-1 through 1-16
2-1 through 2-60
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

Page

Contents

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
```

Documentation Conventions

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 MPX-32 Demonstration Package

1.1 Introduction

The MPX-32 demonstration package illustrates the major features of MPX-32 and establishes the baseline viability of the computer system MPX-32 is running on. The production and operational facilities of MPX-32 are emphasized, as opposed to its program development facilities. The major features exercised in the demonstration are:

- priority CPU scheduling
- swapping
- I/O facilities
  - blocked/unblocked I/O
  - wait/no-wait I/O
  - I/O to regular and extended memory
  - short and long transfers
  - multiple simultaneous I/O from a single task
  - various I/O functions – read, write, backspace, advance, rewind, write EOF
  - various devices – disk, magnetic tape, terminals, line printer, floppy disk
- shared memory (between tasks)
- message sending/receiving
- software interrupt system (abort receivers, break receivers, etc.)
- file locks
- timed scheduling and suspensions
- task activation

Many of the features in this demonstration are not observable, such as message sending between tasks. At the completion of the demonstration, a listing is available which includes a customized banner page, the line printer test messages, and a log of the demonstration activity.

1.2 Demonstration Structure

The MPX-32 demonstration utilizes the test executive called MPXDEMO to run several test tasks according to a predefined script. A complete description of MPXDEMO and its capabilities are described in the MPXDEMO Test Executive section.

The test tasks include IBNNR, IOMUL, IOTST, ITERM, and MIXER. IBNNR produces a banner page for the line printer output of the test session. IOMUL performs multiple no-wait I/O, permanent file creation and allocation, and file locking. IOTST performs most combinations of I/O to a device or file including wait/no-wait, blocked/unblocked, regular memory/extended memory, and variable transfer length I/O. ITERM displays a variety of patterns and banners on a CRT. MIXER provides general system load. Each of these test tasks are described in the MPX-32 Demonstration Test Tasks section in this chapter.

1.3 Hardware Considerations

The MPX-32 demonstration requires the following minimum hardware configuration:

- CPU
- 128K memory
- disk
- floppy disk or magnetic tape
- operator's console
- line printer

The following considerations apply when running the demonstration on systems with additional hardware:

- Additional hardware cannot be used in the demonstration unless it is SYSGENed in.
- An IPU makes the demonstration run slightly faster; however, the accounting information produced in the log is inaccurate and misleading. The demonstration can be altered to illustrate the presence of the IPU more dramatically by adjusting the parameters to MIXER so that MIXER is heavily compute bound.
- A scientific accelerator does not affect the demonstration beyond the performance increase due to the presence of the high speed floating point unit. However, the compute loop in MIXER is comprised primarily of calls to subroutines accelerated by the scientific accelerator; therefore, if the source of MIXER is available, recompiling MIXER with the scientific accelerator option would make it possible to demonstrate its effectiveness.
- Additional memory makes the demonstration run faster, primarily by reducing swapping.
- ITERM must not be used with a hard copy terminal; the demonstration would take hours to complete.
- A copy of IOTST must be run with each terminal, disk drive, magnetic tape drive, floppy disk drive, and line printer configured in the system.
- A copy of ITERM must be run for each CRT terminal configured in the system.
- A copy of IBNNR must be run for each line printer configured in the system.
- The timings produced by the demonstration rely on both the real-time clock and the interval timer. These timers must be properly jumpered and SYSGENed.
- The IOTST test has a map block granularity dependency. See the IOTST documentation.
- The demonstration can run out of disk space if the available disk space is limited and the temporary disk file assignments are large. If the demonstration ceases activity (swap device inactive) and the log is not produced, enter OPCOM and examine the execution queue for tasks in the SWDC state. If the five MPX-32 demonstration test tasks are all in the SWDC state, reboot the system and create additional disk space or reduce the demonstration's requirements.

1.4 Software Considerations

The MPX-32 demonstration can be run on a minimum MPX-32 configuration, such as a starter system. Specific requirements are as follows:

- All devices used in the demonstration must be SYSGENed in.
- There must be approximately 80K of memory available for use by the demonstration. If the demonstration is run with less memory, excessive swapping occurs.
- The demonstration requires an 8K global common partition named GLOBAL80. This partition can be either static or dynamic. Explicit directions on setting up a dynamic partition are given in the GLOBAL80 section in this chapter. If a static partition is desired, see the Volume Manager (VOLMGR) chapter in MPX-32 Reference Manual Volume II.
- Two assumptions are made concerning the real-time clock SYSGEN parameters. First, the MTIM parameter is assumed to be 60. If MTIM is not 60, the fractions of seconds in the times produced in the log are incorrect. Second, the NTIM parameter is assumed to be the same as the MTIM. If NTIM and MTIM differ, the time unit is not one second. The values entered in the /TIMEOUT directive and in the STAGGER and CYCLE fields of the task scheduling directive are actually time units, not seconds.
- It is possible to exhaust the available dispatch queue entries. If during the demonstration MPXDEMO reports it is unable to schedule a test task because of error 5, increase the number of dispatch queue entries using the SYSGEN DISP directive or decrease the demonstration size.
- Devices other than disk should not be SYSGENed with SHR (shared) specified in the DEVICE directive. If other devices are shared, improper displays appear on line printers and terminals and cause tests of floppy disk and magnetic tape to fail.
- The NOCMS SYSGEN directive cannot be used with the MPXDEMO software. MPXDEMO aborts with an SV02.

1.5 MPX-32 Demonstration Program (MPXDEMO)

The MPX-32 demonstration program, MPXDEMO, is included with other optional files on the system distribution tape (SDT). The load modules required for executing MPXDEMO are in the SYSTEM directory and the source files are in the DEMO directory prefixed by the characters US (unsupported source). The load modules required for executing MPXDEMO are as follows:

MPXDEMO — The MPX-32 demonstration executive test tasks

IOTST
IOMUL
ITERM
IBNNR
MIXER

BACKROUN — The background task

TESTSCR3 — minimum CONCEPT/32 system test script
TESTSCR4 — typical large CONCEPT/32 system test script
MPXDTSM — TSM macro for running the MPX-32 demonstration
MPXDBAT — batch stream for running the MPX-32 demonstration

1.6 MPXDEMO Test Executive

MPXDEMO is an MPX-32 test executive which performs the following functions:

- provides a mechanism for scheduling test tasks for execution at various time intervals
- changes the operating parameters for a test task, such as logical file code assignments
- sends a message to the test task
- monitors the progress of a test task
- aborts a test task that continues past a specified time-out value
- collects the results of a test task execution, including a pass/fail indication, execution times, and a message (if any)
- formats the results of a test task execution into a test session log

Only tasks coded for execution with MPXDEMO can be scheduled.

MPXDEMO schedules test tasks and monitors test execution and results. Task scheduling capabilities include:

- multiple activations
- message passing to test task
- alteration of user name, owner name, and priorities
- I/O file and device allocation

Monitoring capabilities include:

- timing facilities
- event recording
- breaking and aborting test tasks that do not respond before the time-out value expires
- test result collection
- test session log

1.7 GLOBAL80

MPXDEMO requires a static or dynamic global memory partition. The partition must be 8K words and be mapped into the end of the regular memory address space. To create an appropriate dynamic global memory partition, enter the Volume Manager and use the directive:

```
CREATE C GLOBAL80 16 240 BRI=N
```

To create an appropriate static global memory partition, see the Volume Manager (VOLMGR) chapter in MPX-32 Reference Manual Volume II.

1.8 MPXDEMO Directives

There are two types of MPXDEMO directives: task and group. Directives consist of up to four input lines, with each line containing up to 80 characters.

To continue a directive to the next line, use a hyphen (-) the end of the line to be continued. Any characters following the hyphen are ignored. A directive cannot be continued from within a parenthetical value, such as a message. Following is an example of an invalid directive:

```
TASKNAME=TEST,MESSAGE=(ABC-
DEF)
```

Directives may not contain embedded blanks except in messages. For example:

```
TASKNAME=TEST, MESSAGE=(AB)
```

is not allowed, but

```
TASKNAME=TEST,MESSAGE=(A B)
```

is allowed.

1.8.1 Task Directives

A task directive causes a test task to be scheduled for activation one or more times during the test session. A task directive has the form:

keyfield [,*keyfield*] . . .

which indicates one or more keyfields where *keyfield* has the form:

keyword=value

The keywords, their meaning, valid values, and special considerations are listed below.

TASKNAME=*taskname*
: *taskname* is a 1- to 8-character test task name. *taskname* is the only required keyfield in a task directive. The test taskname can be specified without TASKNAME= if it appears as the first keyfield in the test directive.

STAGGER=*seconds*
: *seconds* is a 1- to 4-digit decimal integer specifying the number of seconds before the test task is scheduled for the first activation. The default is 0.

REPEAT=*repetitions*
: *repetitions* is a 1- to 2-digit decimal integer specifying the number of times to schedule the test task for activation after the first time. The default is 0.

CYCLE=*seconds*
: *seconds* is a 1- to 4-digit decimal integer specifying the number of seconds between the repeat activations. If not specified and the repeat count is greater than 0, an error occurs.

MESSAGE=(*message*)
: *message* is a 1- to 72-character message to be sent to the test task when the test task is activated. Interpretation of the message is the responsibility of the test task.

PRIORITY=*priority*
: *priority* is a 1- to 2-digit decimal integer from 1 to 64 that specifies the execution priority of the test task.

OWNER=*ownername*
: *ownername* is a 1- to 8-character owner name to be associated with the test task.

USERNAME=*username*
: *username* is a 1- to 8-character user name to be associated with the test task during execution. This feature is important when using the assignment keyfield.

OPTION=(*option*[,*option*]. . .)
: *option* is a 1- to 2-character decimal integer from 1 to 32 specifying the options to be set during test task execution. Interpretation of the options is the responsibility of the test task.

ASSIGN*n*=(*lfc*=*info*)
n is a number from 1 to 4, *lfc* is a 1- to 3-character logical file code, and *info* is identical to the corresponding MPX-32 assignments. For example,

```
$ASSIGN3 OUT=MT10,SAVE,,U
```

is a typical MPX-32 assignment. The corresponding task directive assignment is:

```
...,ASSIGN3=(OUT=MT10,SAVE,,U),...
```

A task directive has up to 5 logical file code (LFC) assignments. Use of the LFCs assigned is the responsibility of the test task.

1.8.2 Group Directives

A group directive affects the operation of a group of directives corresponding to a test session. All group directives start with a forward slash (/) in the first character position. The group directives are listed below.

/END
terminates a directive group. An end-of-file terminates the final directive group.

/TIMEOUT=*seconds*
seconds is a 1- to 5-digit decimal integer that specifies the amount of time to wait after activating the last task in the test session for all test tasks to respond. If this value expires, MPXDEMO aborts all outstanding test tasks.

/BACKGROUND
indicates that the background task is to be run during the test session. The background task can be used to measure system performance.

1.8.3 Sample Directive Stream

Session	Command Input
1	Task directives
	/TIMEOUT=*seconds*
	/BACKGROUND
	/END
2	Task directives
	/TIMEOUT=*seconds*
	/BACKGROUND
	/END
.	.
.	.
.	.
.	.

1.8.4 Command Summary

Task Directives	Group Directives
TASKNAME=*taskname*	/TIMEOUT=*seconds*
STAGGER=*seconds*	/BACKGROUND
REPEAT=*repetitions*	/END
CYCLE=*seconds*	
MESSAGE=(*message*)	
PRIORITY=*priority*	
OWNER=*ownername*	
USER=*username*	
OPTION=(*option*[,*option*]. . .)	
ASSIGN*n*=(*lfc*=*info*)	

1.9 MPXDEMO Special Considerations

MPXDEMO documentation assumes the time unit (as specified during SYSGEN) is one second. If it is not, then wherever the documentation reads seconds, it should be read as time units.

The Text Editor and TSM put characters into columns 73 through 80 of a line. MPXDEMO may be unable to interpret the directives if this occurs.

1.10 MPXDEMO Operation

MPXDEMO schedules test tasks and collects and prints the results. The following logical file codes are used for input and output:

Unit	Usage	Default Assignment	I/O Unit Default Assignment
5	directive input to MPXDEMO	SYC	ASSIGN2 5=SYC
6	directive echo and diagnostics	SLO,100	ASSIGN2 6=SLO,100
7	log of test session	SLO,1000	ASSIGN2 7=SLO,1000

The demonstration begins when MPXDEMO is activated and given a set of directives. MPXDEMO is a standard MPX-32 task, and can be activated interactively, batch, with a command file, or with the OPCOM ACTIVATE directive. The demonstration package provides two methods of activation using the files MPXDBAT and MPXDTSM. The directive input is from TESTSCR3 or TESTSCR4.

MPXDEMO can read its directives from disk files, cards, or terminals.

MPXDEMO accepts directives specifying which test tasks to run, when and how often to run them, what messages to send to them, and which resources, priority, owner name, and user name to associate with them. Additional directive lines set the time-out value, determine whether the background task BACKROUN is to be run, and terminate the directive input. The time-out value determines how long MPXDEMO lets the test session run before terminating it by aborting the outstanding test tasks. The background task uses the available CPU time and reports how much CPU time it used in the test session log.

MPXDEMO operates in response to a directive stream. The directive stream is divided into groups of directives, with each directive group describing a test session. The /END directive, or end of file on the directive input unit, terminates a directive group.

Each directive group is used to make entries in a test task timing table, store a message in a message table, and build a parameter task activation (M.PTSK) block. The timing table has a fixed time increment of 350 seconds and a fixed time span of 4200 seconds. Therefore, activation requests that do not fall on an exact timing table increment are rounded to the nearest one, and activation requests beyond the timing table span are ignored. When all directives in a group are processed, the test session begins. At each time increment, starting at zero, MPXDEMO activates the test tasks scheduled for that time, and then suspends itself until it reaches the next timing table increment containing at least one test task activation.

As test tasks complete, test results are written in GLOBAL80, a static or dynamic global partition. Once MPXDEMO has activated all test tasks in a test session, it waits for their results. If any test tasks have not responded when the time-out value expires, MPXDEMO removes them from the system by sending a break interrupt, and then kills them.

If all test tasks are terminated, MPXDEMO formats and prints the information contained in GLOBAL80 as a test session log, and then processes the next available directive group. One additional capability of MPXDEMO is to activate a background task for the duration of a test session that utilizes all unused CPU time and report the amount of CPU time used in the log. The background task interferes with the test session if any of the test tasks are run at time-distribution priority levels.

Each input file provides a script of directives for MPXDEMO. These scripts can be user customized by editing the MESSAGE keyfield contained in the scripts. Directive usage is described in the MPXDEMO Directives section in this chapter.

To run the MPX-32 demonstration program, complete the following:

1. Install the MPX-32 demonstration package.
2. Create the dynamic Global Common definition, GLOBAL80, using the following Volume Manager directive:

   ```
   VOL>CREATE C GLOBAL80 16 240 BRI=N
   ```

3. Mount scratch tapes with write rings inserted on all configured tape drives and mount scratch floppy disks in all configured floppy disk drives. Depending on the test script, some tape drives and floppy disk drives may not be used.

4. To run the demonstration in the interactive mode, enter TSM and respond:

   TSM>MPXDTSM *testscript*

   testscript is TESTSCR3 or TESTSCR4. See section 1.5 in this chapter. When running the MPX-32 demonstration in this manner, the originating terminal is unavailable to the test tasks.

5. To run the demonstration in the batch mode, complete the following:

   a) Enter the Text Editor and transfer the contents of MPXDBAT into a workfile.

   b) Change the line that reads:

      $ASSIGN1 5=TESTSCR1

      to

      $ASSIGN1 5=*testscript*

      where *testscript* is TESTSCR3 or TESTSCR4. See section 1.5 in this chapter.

   c) Store the file as MPXDBAT with the editor command:

      EDT>STORE MPXDBAT SYS UNN

   d) Enter MPXDBAT into the batch stream with:

      TSM>!BATCH @SYSTEM(SYSTEM)MPXDBAT

      or

      ??BATCH @SYSTEM(SYSTEM)MPXDBAT

   e) Log off the terminal so it can be used by the test tasks, if necessary.

The demonstration proceeds. Lines print on the line printers. Various patterns and banners are displayed on the terminals. Tape drives, floppy disk drives and disk drives advance, backspace, and rewind. The demonstration terminates after producing a log of the test task activity on the line printer.

The duration of the demonstration varies widely depending on configuration and demonstration complexity. A minimum test script (TESTSCR3) run on a minimum configuration takes approximately 30 minutes.

1.11 Monitoring the MPX-32 Demonstration

During the demonstration, there is minimal evidence of activity beyond the physical movements of the devices.

1.12 Error Conditions

MPXDEMO provides error diagnostic facilities for the following types of error: directive, activation, and overflow.

1.12.1 Directive Errors

Directive errors occur when the syntax is incorrect. If MPXDEMO cannot decode a directive, the directive is displayed with continuation characters and the remaining characters removed. The following message is displayed:

```
PROBLEM IN DIRECTIVE:
```

Any directive containing an error is not processed.

1.12.2 Activation Errors

Activation errors occur when MPXDEMO activates the tasks as requested using the parameter task activation service, M.PTSK. If an error is detected, the following message is displayed:

```
**** ERROR **** TASK taskname
COULD NOT BE RUN BECAUSE n
```

taskname is the name of the task and *n* is the error code returned from M.PTSK. These error codes are documented in Chapter 6 of the MPX-32 Reference Manual Volume I in the section describing M.PTSK.

1.12.3 Overflow Errors

Overflow errors occur because the communication region, GLOBAL80, has a limited capacity. The total count of the test task activations must not exceed 60. Each activation attempt after the limit is reached causes the following message to be displayed:

```
**** ERROR **** TASK taskname
NOT RUN DUE TO EXCESSIVE TASKS SCHEDULED
```

taskname is the name of the task.

If MPXDEMO encounters any other error conditions, descriptive messages are produced. The test tasks can also encounter error conditions. The log produced by MPXDEMO includes a field which indicates whether the test task passed or failed. If the test task failed, a variety of other messages can appear. Some possibilities are as follows:

- If an error condition such as an I/O or a system service error is detected, some tasks display a message that might include an event code. Beyond retrying the demonstration, there is little that can be done by the user to correct the problem. The message and the event code are intended solely as debugging aids for Development.

Error Conditions

- If the test task receives a break interrupt, the following message is displayed:

  `BREAK RECEIVED EVENT CODE` *n*

  n is the event code. In most cases, this condition occurs when MPXDEMO times out and purges the test tasks from the system. Check the log to see if a time out occurred. If MPXDEMO is timing out, either the /TIMEOUT value needs to be increased or test tasks are hung up and corrective action requires further investigation. If MPXDEMO has not timed out and the break received message is displayed, it indicates a serious problem that requires further investigation.

- If the test task aborts, the following message is displayed:

  `ABORT CODE` *abortcode extended abortcode* `EVENT CODE` *n*

  abortcode is a 4-character abort code. *extended abort code* is an abort message. *n* is the event code. The following may be helpful in analyzing abort codes:

  - RT*xx* abort messages may result when an incorrect message is passed to the test task.
  - AL*xx* abort messages may result when logical file code assignments are incorrect.
  - TIMEOUT and BREAK INTRPT abort messages may result when MPXDEMO times out and sends a break to the test task which does not respond correctly.

See the MPX-32 Reference Manual Appendix C for further descriptions on abort codes.

1.13 MPX-32 Demonstration Test Tasks

1.13.1 IBNNR

IBNNR produces a banner page for the line printer output of the test session. The form of the banner page is:

```
*****************************************************
*                                                   *
*                      MPX-32                       *
*                                                   *
*                   DEMONSTRATION                   *
*                                                   *
*                      RUN FOR                      *
*                                                   *
*                   customer name                   *
*                                                   *
*                        OF                         *
*                                                   *
*                      company                      *
*                                                   *
*****************************************************
```

The message keyfield specified in an MPXDEMO task directive to be sent to IBNNR has the form:

MESSAGE=(*customer name/company*)

customer name
is an alphanumeric string that can include blanks and most special characters. However, it may not include a forward slash (/).

company
is an alphanumeric string that can include blanks and most special characters

Both *customer name* and *company* are restricted to 20 characters.

I/O units are as follows:

Unit	Usage	Default Assignment	Suggested Assignment
DEV	banner page is written to this unit	None	ASSIGN3 DEV=LP

1.13.2 IOMUL

IOMUL tests multiple no-wait I/O operations from a single task, synchronization file locks, and the creation and access of permanent files.

The message keyfield specified in an MPXDEMO task directive to be sent to IOMUL has the form:

MESSAGE=(*filename*)

filename is an 8-character name for the disk file to be created (if necessary). If synchronization file locks are to be fully tested, 2 copies of IOMUL must be activated simultaneously with the same filename specified.

I/O units are as follows:

Unit	Usage	Default Assignment	Suggested Assignment
DV1	30 records written to unit	None	ASSIGN3 DV1=DC,100
DV2	30 records written to unit	None	ASSIGN3 DV2=DC,100
DV3	30 records written to unit	None	ASSIGN3 DV3=DC,100
DV4	30 records written to unit	None	ASSIGN3 DV4=LP
DV5	30 records written to unit	None	ASSIGN3 DV5=TY

1.13.3 IOTST

IOTST performs most combinations of I/O to a device or file, including wait/no-wait, blocked/unblocked, regular memory/extended memory, and variable transfer length I/O.

The message keyfield specified in an MPXDEMO task directive to be sent to IOTST has the following form.

MESSAGE=(*io* ,*files* , *records* , *incr* , *maxrecordsize* , *mapblocksize*)

io is a 1-digit decimal integer that specifies whether input, output, or both input and output is to be performed as follows:

0 - input and output
1 - input only
2 - output only

files is a 4-digit decimal integer indicating the number of files to read and write. A value of 0 means the default is 3.

records is a 4-digit decimal integer indicating the number of records per file. A value of 0 means the default is 2.

incr is a 4-digit decimal integer indicating the record size increment. A value of 0 means the default is 100.

maxrecordsize
is a 5-digit decimal integer indicating the maximum record size. A value of 0 means the default is 310.

mapblocksize
is a 4-digit decimal integer indicating the map block size in words. Specify 2048 for a CONCEPT/32 computer.

I/O units are as follows:

Unit	Usage	Default Assignment	Suggested Assignment
DEV	I/O device	None	All configured devices

1.13.4 ITERM

ITERM displays a banner and a variety of patterns on a CRT.

The message keyfield specified in an MPXDEMO task directive to be sent to ITERM has the following form.

MESSAGE=(*type*, *banner*)

type indicates the terminal type. A Hazeltine is indicated by an H and an ADM-3 is indicated by an A. Other terminals are not supported.

banner specifies the words formatted as a banner. Each word can contain alphanumeric characters and most special characters. Blanks are used to separate words. Each word consists of up to 12 characters and the banner consists of up to 3 words.

I/O units are as follows:

Unit	Usage	Default Assignment	Suggested Assignment
DEV	CRT device used to display banner and patterns	None	ASSIGN3 DEV=TY or ASSIGN3 DEV=CA

1.13.5 MIXER

MIXER provides general system load by emulating the execution behavior of a typical application. MIXER reads, computes, and writes in a loop.

The message keyfield specified in an MPXDEMO task directive to be sent to MIXER has the following form.

MESSAGE=(*cycles*, *cpuseconds*, *recordsize*, *recordcount*)

cycles is a 5-digit decimal integer specifying the number of times to repeat the loop

cpuseconds is a 5-digit decimal integer specifying the number of seconds to compute during each loop

recordsize is a 5-digit decimal integer specifying the size in words of each record read or written *recordcount*

is a 5-digit decimal integer specifying the number of records to read or write in each loop

I/O units are as follows:

Unit	Usage	Default Assignment	Suggested Assignment
SIN	read at beginning of each loop	None	ASSIGN3 SIN=DC,200,
SOT	written at end of each loop	None	ASSIGN3 SOT=DC,200,U

Note: The files for SIN or SOT must be large enough to contain the records specified by *recordsize* and *recordcount.*

2 Unsupported Software

2.1 Introduction

The following unsupported software is supplied in this release of MPX-32:

- ANALYZE – Crash Dump Analyzer
- ARTS – Display Allocated Resource Table
- AUTOJOB – Automatic Batch Job Submission on Boot-up
- CHECKDEF – TERMDEF Utility
- CONFIG – Processor and Interrupt Configuration Module
- CSWI – Context Switch Timing for M.SURE
- DD – Disk Dump by Sector
- DDUMP – Disk Dump by File
- DEVI – Display Device Information
- DISCERR – Display Disk Error History
- DISPTWO – Display Two ASCII Files
- DOWNDATE – Source Compare Program
- DOWNDAT2 – Source Compare/Source Update Output
- DSPC1, DSPC2 – Display Disk Space Usage Utility
- FOLAP – File Overlap Detection Utility
- GM.CHHST – State Chain History
- GM.CLS – TERMDEF Demo
- GM.FLOW – Assembly Source Code Flowchart Tool
- GM.SEAR – Source Search Tool
- GM.TDEMO – TERMDEF Demo
- INTR – Indirectly Connected Interrupt Response Timing
- LASER – Laser Printer Utility
- LMINFO – Load Module Information
- LOGMDT – Log Contents of Rapid File Allocation MDT
- LOGOFF – Remote Terminal Logoff
- LOOK – String Search
- MIPSMON – Instruction Sequence Timing Tool
- NEWCOPY – Floppy Disk Duplication Tool
- POOLSCAN – Memory Pool Monitor
- PORTPROT – Dial-up Port Protection
- RNVOL – Rename Volume
- RT_DEBUG – Realtime Debugger
- SDUTIL – Small Computer System Interface (SCSI) Disk Utility
- SPRINT – Serial Printer Formatter/Spooler
- SRCH – Interactive String Search

- SW.CHART – Swapper Percentage Active Monitor
- SW.MODI – Interactive Swap Parameters Modifier Program
- SW.MON – Swap Monitor Program Version 2
- SWAPMON – Swap Monitor Program Version 1
- SYSINFO – Display Communications Region
- TABS – Change Tabs
- TERMLOAD – Terminal Initializer/Loader
- TSCAN – TSM Scanner Demo
- UDTS – Display UDT
- US.ERR – Unsupported Software Error Codes

To access an individual tool, enter the keyword specified in capital letters at the TSM prompt. For example,

```
TSM>DD
```

enters the program Disk Dump by Sector.

2.2 ANALYZE – Crash Dump Analyzer

ANALYZE is an interactive task for examining the contents of system tables and queues after a system crash.

Initially, ANALYZE must be supplied with a tape or floppy disk produced by H.DMPMT or an equivalent routine, such as the DUMP entry point of H.DEBUG2. The H.DMPMT data is read and stored on a named disk file. The disk file is used in subsequent activations for analyzing the crash. The display from ANALYZE is directed to either the SLO device or the terminal; the default is to the terminal.

ANALYZE first prompts for the pathname of the disk file. Then, it prompts for tape, floppy, or disk. If the reply is disk, it uses the named disk file which was previously produced by ANALYZE. If the reply is tape or floppy, it uses the appropriate device. If a disk file exists and a tape or floppy is being used, ANALYZE prompts to delete the disk file (reply Y or N). If the reply is Y or if a disk file does not exist, ANALYZE writes to the disk file. ANALYZE creates the file dynamically. If the reply is N, ANALYZE exits.

System tables and queues sometimes change for each MPX-32 release. ANALYZE checks the revision number of the system where the dump was produced. If the revision number does not match the current MPX-32 revision, it displays a continue option prompt (reply Y or N).

All numbers displayed and accepted by ANALYZE are hexadecimal except error codes.

Following are ANALYZE directives and descriptions:

Directive	Description
CDT	display controller definition tables
COMM	display communications region variables and/or dispatch queue entries
DQE	display dispatch queue entries
DUMP	display contents of memory or a task
EXIT	exit ANALYZE
HELP	display useful information
INDEX	display addresses of data structures
POOL	display memory pool
PRINT	divert most output to SLO device
SEARCH	search for occurrences of a value
SET SAME	set or reset ANALYZE logic
SMT	display shared memory table entries
TSA	display task service areas
UDT	display unit definition tables
VIEW	divert all output to user terminal

2.2.1 CDT Directive

The CDT directive displays a text description of the controller definition table (CDTs). If no parameters are specified, the CDT directive displays all CDTs. The following information is contained in the display:

- physical address of this CDT
- CDT index (CDT.INDX)
- UDT index for each device on the controller
- class (CDT.CLAS)
- device type (CDT.DTC)
- mnemonic, channel, and subaddress (CDT.CHAN, CDT.SUBA)
- interrupt priority level (CDT.IPL)
- number of units on controller (CDT.NUOC)
- number of outstanding requests (CDT.IORO)
- flags (CDT.FLGS, CDT.FLG2)
- I/O status (CDT.IOST)
- interrupt handler address (CDT.SIHA)

The I/O queue (IOQ) summary consists of the number of entries in the IOQ list (CDT.IOCT) and the information listed below. When this information is displayed, the previous IOQ (BIOQ) is listed first, followed by the current IOQ (FIOQ) and any other IOQs linked in the UDT entry's IOQ list. The IOQ summary includes:

- indices of the associated CDT and UDT
- status and flags (IOQ.STAT, IOQ.FLGS)
- FCB or TCPB address (IOQ.FCBA)
- program number (IOQ.PRGN)
- handler function words (IOQ.FCT1, IOQ.FCT2, IOQ.FCT3, IOQ.FCT4)
- number of bytes transferred (IOQ.UTRN)
- number of words in the OS buffer (IOQ.WOSB) if the device is nonextended I/O
- buffer addresses (IOQ.FBUF, IOQ.TBUF)
- I/O return status (IOQ.IOST, IOQ.IST1, IOQ.IST2)
- FCB control information (IOQ.CONT)

Syntax

CDT [IOQ] [*devmnc* | *index*] [,*devmnc* | ,*index*] ...

[IOQ] displays an IOQ summary for each CDT listed

[*devmnc* | *index*]
devmnc is the device type mnemonic, such as TY. *index* is the CDT index. If *devmnc* and *index* are omitted, the CDT directive displays all CDTs.

2.2.2 COMM Directive

The COMM directive displays a text description of the communications region and all active DQEs. See the DQE directive. If no parameters are specified, the COMM directive displays a text description of all DQEs in those states.

The communication region information displayed by default is:

- system name (C.SYSTEM)
- revision number (C.REV)
- patch level (C.UPDT)
- status and system configuration flags (C.BIT)
- count of outstanding interrupts and traps (C.GINT)
- count of memory release events (C.RRUN)
- configuration flags (C.CONF)
- counts of memory modules available (C.TMAC, C.EMAC, C.HMAC, C.SMAC)
- counts of memory modules configured (C.TMCC, C.EMCC, C.HMCC, C:SMCC)
- address of memory pool (C.SBUF)
- number of words in memory pool for each queue (FREE and USED):
  - head cell address
  - number of entries in the queue
  - maximum cell size
  - minimum cell size
  - total number of words in the queue

Syntax

COMM [*qmnc*] [,*qmnc*] ...

[*qmnc*] is a queue mnemonic, such as CURR or SWGQ. If *qmnc* is omitted, COMM displays all DQEs.

2.2.3 DQE Directive

The DQE directive dumps the specified active DQEs in text format. If no parameters are specified, all active DQEs are dumped.

If an invalid parameter is specified, an error message is displayed on the user terminal.

Note: In the case of J.SWAPR, if the DQE state is SUSP, it is not necessarily linked into the SUSP chain.

The following information is provided:

- physical address of this DQE
- DQE entry number
- load module name

- base mode task declaration
- current user priority (DQE.CUP)
- base priority of user task (DQE.BUP)
- I/O priority (DQE.IOP)
- task state (DQE.US)

If the task is in the MRQ, the memory request type (DQE.MRT) is displayed.

If the task is in the SWGQ, the following information is displayed:

- general queue identity (DQE.PRS, DQE.PRM)
- general queue function code (DQE.GQFN). If this is queued for volume resource or queued for dual-port lock or queued for synchronous resource lock, then a description of the associated allocated resource table (ART) is displayed:
  - UDT index (AR.UDTI)
  - resource descriptor block address (AR.BLOCK)
  - current access mode (AR.CACM)
  - resource allocation flags (AR.FLAGS)
  - Resource information:

Resource	Display
volume	MVT entry pointer (AR.MTVA)
segment definition	number of blocks in definition (AR.NBLKS)
memory partition	SMT entry pointer (AR.SMTA)
device	UDT entry pointer (AR.UDTA)

  Else volume name:
  - DQE index of the exclusive lock owner (AR.XRL)
  - DQE index of the synchronous lock owner (AR.SRL)
  - number of active assignments (AR.ASSNS)
  - number of resource users (AR.USERS)
  - number of dual-processor requests queued (AR.QUE)
  - number of readers currently on this resource (AR.RDRS)
  - current EOF position in this file (AR.EOF)
  - current EOM position in this file (AR.EOM)
- task activation number (DQE.TAN)
- owner name (DQE.ON)
- pseudonym (DQE.PSN)
- user status word (DQE.USW)
- scheduling flags (DQE.USHF)
- abort code (DQE.ABC) (when an abort is in progress)
- swapping inhibit flags (DQE.SWIF)

- number of no-wait I/O requests (DQE.NWIO)
- number of unbuffered I/O requests currently outstanding (DQE.UBIO)
- number of no-wait mode run requests outstanding (DQE.NWRR)
- number of no-wait mode message requests outstanding (DQE.NWMR)
- number of swappable E-class map blocks currently allocated (DQE.CME)
- number of swappable H-class map blocks currently allocated (DQE.CMH)
- number of swappable S-class map blocks currently allocated (DQE.CMS)
- inclusive span of maps in use (DQE.MAPN)
- map span required for MIDLs and MEMLs (DQE.MSPN)
- shadow memory flags (DQE.SHF)

Syntax

DQE [*index* | *lmn*] [,*index* | ,*lmn*] ...

[*index* | *lmn*]
: *index* is the DQE entry number. *lmn* is the load module name. If *index* and *lmn* are omitted, the DQE directive dumps all active DQEs.

2.2.4 Dump Data Directive

The dump data directive writes the specified data in hexadecimal/ASCII format – four words per line on the user terminal, or eight words per line preceded by an index of data structures on the SLO device. See the INDEX directive. When no parameters are specified, all data is dumped.

Syntax

DUMP [*staddr*] [,*endaddr*]

[*staddr*] is the physical address where the dump begins. The default is zero.

[,*endaddr*] is the physical address where the dump ends. The default is the last data dumped by H.DMPMT.

2.2.5 Dump Load Module Directive

ANALYZE displays the DQE index followed by a side-by-side ASCII/hexadecimal dump of the load module – four words per line on the user terminal, or eight words per line on the SLO device.

Note: The tasks with hand-built TSAs cannot be dumped in this manner.

Syntax

DUMP #,*index*[,**R**] [,*staddr*] [,*endaddr*]

#	indicates that a load module is to be dumped
index	is the DQE entry number
[,**R**]	denotes addresses relative to the start of DSECT. If R is not specified, addresses are logical and start at the beginning of TSA.
[,*staddr*]	is the address where the dump begins
[,*endaddr*]	is the address where the dump ends

2.2.6 EXIT Directive

The EXIT directive exits ANALYZE. A prompt is displayed asking whether to delete the disk file; reply Y or N.

Syntax

EXIT

2.2.7 HELP Directive

The HELP directive displays information about the topic. If a topic is not specified, a list of topics is displayed.

Syntax

HELP [*topic*]

[*topic*]	is the directive to be described

2.2.8 INDEX Directive

The INDEX directive displays on the terminal the physical addresses of various data structures regardless of print or view execution. The addresses displayed are:

- allocated resource table (AR.)
- channel definition table (CHT.)
- communication region (C.)
- controller definition table (CDT.)
- CPU scratchpad image
- device type table (DTT.)
- dispatch queue (DQE.)
- end of data read in
- end of operating system
- interrupt timer table (ITT.)
- map image list for operating system
- master process list
- memory allocation table (MEM.)
- module address table
- mounted volume table (MV.)
- resourcemark table
- shared memory table (SMT.)
- unit definition table (UDT.)

Syntax

INDEX

2.2.9 POOL Directive

The POOL directive displays the start and end physical addresses and the number of free or used cells in the memory pool. It then displays the forward and backward links, the address of the cell header, and the number of words for each cell followed by a side-by-side hexadecimal/ASCII dump of the cell's data area.

Syntax

POOL {FREE|USED}

FREE	specifies the free cells in memory pool
USED	specifies the used cells in memory pool

2.2.10 PRINT Directive

The PRINT directive prints all requested data on the SLO device until a VIEW directive is executed.

Syntax

PRINT

2.2.11 SEARCH Directive

The SEARCH directive searches within the specified addresses for a match to the supplied value.

Syntax

SEARCH [*staddr*] , [*endaddr*] ,*value* [, *mask*]

[*staddr*]	is the address where the search begins. The default is 0.
[*endaddr*]	is the address where the search ends. The default is the last address dumped by H.DMPMT.
value	is the hexadecimal value to be matched
[, *mask*]	is a hexadecimal word mask. The default is 'XFFFFFFFF'.

2.2.12 SET SAME Directive

The SET SAME directive sets or resets the SAME logic for a hexadecimal/ASCII dump. The SAME logic compares the last word of the previous line to the entire current line. If they are the same, the current line is not printed and SAME is displayed. For all consecutive lines that match, SAME is displayed once. SAME logic remains set or reset until it is changed by the SET SAME directive.

Syntax

SET SAME={Y|N}

Y	enables SAME logic. Default.
N	disables SAME logic

2.2.13 SMT Directive

The SMT directive displays a text description of the shared memory table (SMT) for the specified partition, owner name, or task number. If no parameters are specified, all SMT entries are displayed.

The following information is provided:

- physical address of the SMT
- partition name (SMT.NAME)
- owner name or task number associated with the partition, if any (SMT.TNUM)
- owner name of partition creator (SMT.OWNR)
- project group name associated with partition (SMT.PROJ)
- entry index (SMT.IND)
- flags (SMT.FLAG)
- number of tasks sharing this memory that are not outswapped (SMT.UCNT)
- number of map blocks (SMT.MAPN)
- starting map register number (SMT.MAPS)
- memory type (SMT.MTY) (if dynamic memory partition)
- total number of pages (SMT.PTOT)
- start address of map image descriptor list (SMT.MIDL)

Syntax

SMT [*owner* | *pname*] [,*owner* | ,*pname*] ...

[*owner* | *pname*]
: *owner* is an owner name or task number. *pname* is a partition name. If *owner* and *pname* are omitted, the SMT directive displays all SMT entries.

2.2.14 TSA Directive

The TSA directive dumps TSAs in text format. If no parameters are specified, all active TSAs are dumped. If an invalid parameter is specified, a message is displayed on the terminal.

Note: The TSA for J.SWAPR cannot be dumped.

The following information is displayed:

- task's load module name
- task's DQE index
- contents of the pushdown stack (registers and PSD for each level)
- logical start address of TSA
- physical start address of TSA
- address of task's file assignment table (T.FATA)
- address of task's file pointer table (T.FPTA)
- address of task's segment definition table (T.SEGA)
- number of FAT/FPT pairs associated with task (T.FILES)
- start address of task's DSECT area (T.BIAS)
- end address of task's DSECT area (T.END)
- transfer address of task's main segment (T.TRAD)
- end address of the TSA (T.TEND)
- status bits (T.BIT1, T.BIT2, T.BIT3, T.BIT4, T.BIT5)
- map image descriptor list and memory attribute list (T.MIDL, T.MEML)
- DSECT origin within T.MEML/T.MIDL (T.DSOR)
- DSECT size in map blocks (T.DSSZ)
- CSECT origin within T.MEML/T.MIDL (T.CSOR)
- CSECT size in map blocks (T.CSSZ)
- extended address origin in T.MEML/T.MIDL (T.EAOR)
- extended address size in map blocks (T.EASZ)

If the task was a base mode task, the following is also provided:

- task's prelocation delta (T.PREL)
- start address of the shared image descriptors (T.SHIMDA)
- number of shared image descriptors (T.NSI)

Syntax

TSA [*index* | *lmn*] [,*index* | ,*lmn*] ...

[*index* | *lmn*]
index is the DQE entry number. *lmn* is a load module name. If *index* and *lmn* are omitted, the TSA directive dumps all active TSAs.

2.2.15 UDT Directive

The UDT directive displays a text description of the UDTs. If no parameters are specified, descriptions of all UDTs are displayed.

The following information is displayed:

- physical address of the UDT
- UDT index (UDT.UDTI)
- CDT index (UDT.CDTI)
- unit status (UDT.STAT, UDT.STA2)
- If the device is allocated (see flags):
  - DQE index
  - load module name
  - associated ART data (see DQE directive)
- device type code (UDT.DTC)
- mnemonic, channel, and subaddress (UDT.CHAN, UDT.SUBA)
- flags (UDT.FLGS, UDT.BIT2)
- service interrupt handler address (UDT.SIHA)

The IOQ summary consists of the number of entries on the list (UDT.IOCT) plus any IOQ data supplied by the CDT directive.

Syntax

UDT [**IOQ**] [*devmnc* | *index*] [,*devmnc* | ,*index*] ...

[**IOQ**] displays an IOQ summary for each UDT listed

[*devmnc* | *index*]
devmnc is a device type mnemonic, such as TY. *index* is the UDT index. If *devmnc* and *index* are omitted, the UDT directive displays all UDTs.

2.2.16 VIEW Directive

The VIEW directive displays all requested data on the user terminal until the PRINT directive is executed.

Syntax

VIEW

Examples

Using the tape for the first time

`TSM>ANALYZE`	
`PLEASE ENTER THE DISC FILE PATHNAME`	
`ANA>@TB00(SYSTEM)DUMP`	
`USE TAPE, FLOPPY OR DISC FILE?`	
`ANA>TAPE`	Reads tape to above disk file.
`DISC FILE ALREADY EXISTS`	Only displayed if the
`SHALL I DELETE THE DISC FILE?`	disk file exists.
`ANA>Y`	
`THE SUPPLIED DATA IS FROM A`	
`REVISION` *xxxx* `SYSTEM.`	
`CONTINUE ? (Y/N)`	
`ANA>Y`	
`ANA>TSA`	Displays all active TSAs to terminal.
`ANA>DUMP 180000,`	Displays memory from 180000 hex to the end of the data from H.DMPMT to the terminal.
`ANA>TSA J.TSM,A,J.SOUT`	Displays the TSAs of J.TSM, the task with DQE entry number A and J.SOUT to the user terminal.
`ANA>DUMP #,5`	Displays the load module with DQE entry 5 to the user terminal.
`ANA>PRINT`	Diverts display to the SLO device
`ANA>CDT IOQ 5,TY`	Prints the CDTs for CDT entry 5 and all teletypewriters with IOQ summaries to the SLO device.
`ANA>SMT GLOBAL01`	Prints the SMT of GLOBAL01 to the SLO device.
`ANA>EXIT`	
`SHALL I DELETE THE DISC FILE?`	
`ANA>N`	
`TSM>`	

Using the previously produced disk file

```
TSM>ANALYZE
PLEASE ENTER THE DISC FILE PATHNAME
ANA>@TB00(SYSTEM)DUMP
USE TAPE, FLOPPY OR DISC FILE?
ANA>DISC
THE SUPPLIED DATA IS FROM A
REVISION xxxx SYSTEM.
CONTINUE ? (Y/N)
ANA>Y
directives for ANALYZE
ANA>X
SHALL I DELETE THE DISC FILE?
ANA>Y                                  If you have finished
                                       with the disk file
                                       enter Y, otherwise, enter N.
TSM>
```

2.3 ARTS – Display Allocated Resource Table

ARTS displays the first six words of each allocated resource table (ART) entry that is used on the terminal.

Syntax

ARTS

Default Assignments

```
$AS LO TO LFC=UT
$AS SI TO LFC=UT
```

2.4 AUTOJOB – Automatic Batch Job Submission on Boot-up

AUTOJOB is activated after J.TSM by the SYSGEN directive SEQUENCE=(AUTOJOB). The program searches the system volume and directory for the file M.BATCH. If the file is not found, AUTOJOB exits. If it is found, AUTOJOB scans each line and processes the directives which are syntactically correct. The OPTION directive can be used to prevent error messages from being displayed at the console and to abort AUTOJOB if an error occurs.

Note: This program is written in FORTRAN 77+.

Syntax

AUTOJOB

2.4.1 Command File Syntax

cmd=arg [*com*]

or

[-] *path* [*com*]

cmd — is the command keyword terminated by the equal sign

arg — is the command argument

[*com*] — is the comment field. Comments are ignored by AUTOJOB.

\- — inhibits the message which indicates that the job was sent to the batch stream

path — is the full pathname of the file to be sent to the batch stream at boot-up. The files are preprocessed by J.SSIN and can be edit stored or saved, Toolkit card image, or default format with trailing blanks removed.

2.4.2 Command Summary

* [*com*] — *com* is a comment. Any line starting with the asterisk (*) character is ignored by AUTOJOB.

NOTE=[*com*]

com is a text message up to column 72 which is sent to the console in the format AUTOJOB:*com*. This command is used primarily to produce milestone messages for M.BATCH file processing.

OPTION=[MESSAGE | NOMESSAGE] [*com*]

MESSAGE sends messages to the console (default). NOMESSAGE inhibits message output to the console. *com* is an optional comment. The OPTION flags can be enabled or disabled at any time to control processing.

OPTION=[ABORT | NOABORT] [*com*]

ABORT aborts command processing if an error is detected. NOABORT continues command processing if an error is detected (default). *com* is an optional comment. The OPTION flags can be enabled or disabled at any time to control processing.

[-] *path* [*com*]

\- inhibits the message which indicates that the job was sent to the batch stream. *path* is the pathname of the file submitted to the batch stream. The pathname must be followed by a blank. *com* is an optional comment.

2.4.3 Resource Summary

Unit CT is the LFC used to write messages to the console. It is dynamically allocated and should not be reassigned.

Unit BAT is the LFC used to read the M.BATCH file. It is dynamically allocated and should not be reassigned.

2.4.4 M.BATCH File Example

```
*
**Prepare the system for application
*
OPTION=ABORT                         If the following job fails, abort
NOTE=Start up application program
@USER1(SETUP)BUILD                   Build application environment
OPTION=NOABORT                       Do not abort if errors occur
@USER1(SETUP)COPY                    Copy files to application disk
@USER1(SETUP)START                   Start application
-@USER1(SETUP)CLEANUP                Clean up everything except messages
OPTION=NOMESSAGE                     Do not send messages to the console
*
** Process System Administrator tasks
*
@SYSTEM(SYSTEM)CLEANUP               Clean up workfiles
@SYSTEM(SYSTEM)BUILDMD               Initialize and mount memory disk
@SYSTEM(SYSTEM)MONITOR               Activate background monitor
OPTION=MESSAGE                       Send messages to the console
NOTE=Log all the disk files
@SYSTEM(SYSTEM)LOGDISC               Log all the files on the system
NOTE=The job is complete
```

Notes:

All directives must start in column one. They are processed up to column 72.

The M.BATCH file must be edit stored or Toolkit card image.

The jobs submitted must mount required volumes by the JCL in the particular batch job or must refer to volumes mounted by the M.MOUNT file.

The OPTION=NOMESSAGE directive does not disable error messages. The hyphen (-) option is not required if this directive is used.

The following defaults are in effect at start-up:

```
OPTION=NOABORT
OPTION=MESSAGE
```

2.5 CONFIG – Processor and Interrupt Configuration Module

CONFIG displays and/or prints the system's current hardware configuration. It is run interactively from TSM. The source file US.CONFG contains the job stream and source to create the load module CONFIG.

This program is written in FORTRAN 77+.

Syntax

CONFIG

The following prompt appears:

```
DO YOU WANT A HARDCOPY OUTPUT? (Y/N):
```

Sample Output

The following is a sample of processor configuration output.

```
================================================================

              PROCESSOR CONFIGURATION


PROCESSOR TYPE : 32/9780

 CPU = 97    (REV: F)                  IPU = 97    (REV: F)
 MACC = ON                             MACC = ON
 FPA = N/A                             FPA = N/A
 WCS = N/A                             WCS = N/A
 CACHE =      0 KB (0 BOARD)           CACHE =      0 KB (0 BOARD)
 SHADOW = 4096 KB (1 BOARD)            SHADOW = 4096 KB (1 BOARD)
    SLOT      PHYSICAL ADDR               SLOT      PHYSICAL ADDR
     0      (000000 - 3FFFFF)              0      (000000 - 3FFFFF)
     1      (NOT  CONFIGURED)              1      (NOT  CONFIGURED)
     2      (NOT  CONFIGURED)              2      (NOT  CONFIGURED)
     3      (NOT  CONFIGURED)              3      (NOT  CONFIGURED)

================================================================
```

The following is a sample of interrupt configuration output.

```
=================================================================

                    INTERRUPT CONFIGURATION

LEVEL   CHANNEL  TYPE     HANDLER   LOW RATE(SEC)     HIGH RATE(SEC)
-----   -------  ----     -------   -------------     --------------
00      0000      I/O     H.MDXIO       N/A               N/A
05      0900      I/O     H.DCXIO       N/A               N/A
06      0800      I/O     H.DCXIO       N/A               N/A
07      0400      I/O     H.DCXIO       N/A               N/A
08      0C00      I/O     H.DCXIO       N/A               N/A
09      1000      I/O     H.MTXIO       N/A               N/A
0A      1800      I/O     H.DPXIO       N/A               N/A
0B      0E00      I/O     N.ENXIO       N/A               N/A
0C      0600      I/O     H.DCXIO       N/A               N/A
10      7600      I/O     H.DCSCI       N/A               N/A
13      7E00      I/O     H.CTXIO       N/A               N/A
18      7F06    NON I/O    H.IPCL       N/A               N/A
1C      7F03    NON I/O      ITLB       N/A               N/A
20      6904      UNK        ITLB     ***DEVICE NOT PRESENT***
5E      7904     RTOM     H.IPUIT   .0000384          .0000003
5F      7F04    IOP/MFP    H.IPIT   .0000384          .0000384

=================================================================
```

2.6 CHECKDEF – TERMDEF Utility

CHECKDEF verifies new terminal definitions added to the TERMDEF facility. CHECKDEF operates in interactive mode and cannot be run in batch mode.

CHECKDEF prompts for a two-character control code and calls M.GETDEF using the terminal function as the requested type. (The control codes are upper/lower case sensitive.) It then displays the returned block and data based on the type of terminal that CHECKDEF is being run on.

If the TERMDEF facility has been excluded with the SYSGEN NOTDEF directive, CHECKDEF displays the following message on the terminal:

```
TERMDEF HAS NOT BEEN INSTALLED
```

Note: This program is written in FORTRAN 77+.

Syntax

CHECKDEF

2.7 CSWI – Context Switch Timing for M.SURE

CSWI is comprised of real-time test modules that measure the performance of the fast context switch logic module H.SURE (Suspend/Resume). H.SURE and a real-time option module (RTOM) must be configured into the current system image. (The RTOM must be jumpered for a high frequency select of 300 nanoseconds.)

The CSWI modules test that the M.SURE service is optimized for real-time tasks of the same priority by measuring the time that M.SURE takes to execute the first instruction of the CSWI.B task. It repeats this 50 times and places the results in the permanent disk file CSWI.OUT.

The CSWI environment consists of the following:

File	Description
BLD.CSWI	macro to build CSWI performance tests
CLEAN_UP	macro to delete unnecessary files from the SYSTEM directory after the tests are run
COPY_FILES	macro to copy necessary files to the SYSTEM directory before the tests are run
CSWI.A	M.SURE caller load module
CSWI.B	M.SURE target load module
CSWI.I	timing report load module
J.CSWI.A	source for M.SURE caller
J.CSWI.B	source for M.SURE target
J.CSWI.I	source for the time report program
J.STPK	source for the FORTRAN statistical package
JT.SURE	JCL to run CSWI performance tests
SG.II	example SYSGEN directives

The CSWI modules run in E-class memory from the SYSTEM directory in a stand-alone environment for consistent results. The standard configuration is 32KB cache, two-way interleaved memory on a CPU-only system. The RTOM must be SYSGENed at priority 20 with subaddress 04 as shown in the following sample SYSGEN statement:

```
PRIORITY=20,RTOM(79,04),INTV
```

For a more detailed example of the SYSGEN statements required for running the CSWI modules, refer to the SG.II file.

2.7.1 Running the CSWI Modules

Following are the instructions for running the CSWI modules:

1. Sign on to MPX-32 as ownername SYSTEM.
2. Ensure that the CSWI modules are located in directory CSWI on volume SYSTEM. (If these files do not exist on the system, restore them from the SDT tape.)
3. Change directory to `@SYSTEM^CSWI`.

4. Copy the required files to the SYSTEM directory with the COPY_FILES macro.
5. Change directory to `@SYSTEM^SYSTEM`.
6. Execute the CSWI modules with the following statement:

```
TSM>BATCH JT.SURE
```

7. Upon completion, the file CSWI.OUT in the SYSTEM directory contains the results. The CSWI modules produce results for all combinations of the following: M.SUME/M.SUSP, M.SURE, real-time accounting on (ONRA) or off (OFRA) and in cache and cache bust. Each combination has the following columns of data:

   raw ticks — the number of clock ticks from the RTOM during one context switch

   raw nsec — the number of nanoseconds for one context switch

   adj nsec — the adjusted time for one context switch. (The overhead is subtracted from the raw nanoseconds to read the RTOM interval timer.)

   C.INTC — the clock tick counter value

   A statistical analysis is provided after each combination. The number of values and the percentage of those values falling between the average (plus or minus one standard deviation and those outside one standard deviation) are shown. The context switch values are in microseconds.

8. Remove unnecessary files from the SYSTEM directory with the CLEAN_UP macro in the CSWI directory. This macro does not delete CSWI.OUT.

Notes:

To recompile the CSWI modules, use the BLD.CSWI macro. Pass as a parameter the name of the FORTRAN compiler and the run-time library files to be used as shown in the following example:

```
TSM>BLD.CSWI FORT45NH SRTL45NH SRTD45NH
```

The CSWI modules provided on the SDT tape have been built with FORTRAN 77+ 4.4 and Macro Assembler 3.0.

2.8 DD – Disk Dump by Sector

DD creates a side by side hexadecimal/ASCII dump of any system-configured disk and allows for disk sector modification.

Syntax

DD *addr* [,*qty* [,*dev* [,*str*]]]

addr is the absolute sector to start the dump (hexadecimal)

[,*qty* [,*dev* [,*str*]]]

qty is the number of sectors to dump. The value is hexadecimal if preceded by X (e.g., X1F or X'1F'); otherwise decimal. If *qty* is omitted, DD dumps all sectors from *addr* to the end of the disk.

dev is the device to be dumped (e.g., DM0800). If *dev* is omitted, DD dumps the system disk.

str is a 1- to 8-ASCII character search string. When a match is found, the block numbers containing *str* are displayed. DD searches from *addr* to the end of the disk, unless *qty* is also specified.

The disk can be modified if the owner is SYSTEM and option 13 is set prior to activating DD. To modify the disk, press any key other than <ret> at the TSM `ENTER CR FOR MORE` prompt. The following prompt is displayed:

```
TO PATCH DISC FILE
ENTER BLK #, WORD #, VALUE1 [, VALUE2...]
```

`BLK #` specifies the hexadecimal disk sector to be modified

`WORD #` specifies the hexadecimal word within the specified disk sector where modification is to begin

`VALUE1 [, VALUE2...]`

specifies the desired values in hexadecimal. Specifying only one value modifies only the word specified. Multiple values modify the specified word with the first value and sequentially subsequent words with the subsequently specified values. The number of values specified is limited by what can be specified on one line. Re-enter modification mode to continue modifications.

`ZAP` can be specified instead of a word and value to zero all 192 decimal words of the specified sector.

After modification, DD displays the sector with the modified values.

Modification Examples

The following example modifies words 57, 58 and 59 of sector 4B7 with the values `00000069`, `00000021`, and `00001290` respectively:

```
4B7, 57, 00000069, 00000021, 00001290
```

The following example zeros all 192 decimal words of hexadecimal sector 121:

```
121, ZAP
```

Optional Assignments

`$ASSIGN OT TO LFC=UT`	(default – 4 word wide dump)
`$ASSIGN OT TO SLO`	(8 word wide dump)

Aborts

`RM02 - ACCESS MODE NOT ALLOWED`
Someone other than owner SYSTEM attempted to modify the disk.

2.9 DDUMP – Disk Dump by File

DDUMP creates a side by side hexadecimal/ASCII dump of a disk file.

A file can be patched by pressing any key except return at the TSM `ENTER CR FOR MORE` message.

Syntax

DDUMP *file* [,*stad* [,*enda*]]

file — is a 1- to 16-character file name of a file in the current working directory or the system directory that is to be dumped

[,*stad* [,*enda*]]
stad is a hexadecimal starting address relative to the first block of the file (0) where the dump starts. *enda* is a hexadecimal ending address relative to the first block of the file where the dump stops.

Optional Assignments

`$ASSIGN OT TO LFC=UT`	(default – 4 word wide dump)
`$ASSIGN OT TO SLO`	(8 word wide dump)

Aborts

Various system aborts. No internal aborts.

2.10 DEVI – Display Device Information

DEVI displays the following device information about the requested device: UDT, CDT, DCA, and IOQ if present.

Syntax

DEVI [*-opt*] *devmnc*

[*-opt*]	*opt* specifies the following options: T is terminal output (default) and P is printer output (SLO)
devmnc	specifies the device mnemonic

Default Assignment

`$AS LO TO SLO` (for P option and/or abort messages)

2.11 DISCERR – Display Disk Error History

DISCERR displays how many recoverable and unrecoverable disk errors occurred for all disk drives. This information is cumulative from the time the system was booted.

Syntax

DISCERR

2.12 DISPTWO – Display Two ASCII Files

DISPTWO displays two uncompressed, blocked, ASCII files side by side for visual review. The user can interactively select options to control the display.

Syntax

DISPTWO

Default Assignment

`$AS OT TO SLO`

2.13 DOWNDATE – Source Compare Program

DOWNDATE identifies changes between program source files.

Note: This program is written in FORTRAN 77+.

Syntax

DOWNDATE

or

$EXECUTE DOWNDATE (batch mode)

Default Assignments

`$ASSIGN 1 TO ORIGINAL`	(old filename – static assignment required)
`$ASSIGN 2 TO UPDATED`	(new filename – static assignment required)
`$ASSIGN 5 TO DEV=NU`	(synchronization – normally not used)
`$ASSIGN 6 TO SLO`	(output from program – optional)

Aborts

Various FORTRAN run-time aborts. If there are too many changes, the program loses sync and is not able to resync.

2.14 DOWNDAT2 - Source Compare/Source Update Output

DOWNDAT2 is an enhanced version of DOWNDATE that identifies changes between source files. The output format is slightly different from DOWNDATE. Generated output can be processed by Source Update.

Note: This program is written in FORTRAN 77+.

Syntax

DOWNDAT2

or

$EXECUTE DOWNDAT2 (batch mode)

Default Assignments

`$ASSIGN 1 TO ORIGINAL`	(old filename – static assignment required)
`$ASSIGN 2 TO UPDATED`	(new filename – static assignment required)
`$ASSIGN 5 TO DEV=NU`	(synchronization – normally not used)
`$ASSIGN 6 TO SLO`	(output from program – optional)
`$ASSIGN 7 TO DEV=NU`	(for source update processing – optional)

Aborts

Various FORTRAN run-time aborts.

Note: If there are too many changes, the program loses sync and is not able to resync.

2.15 DSPC1, DSPC2 – Display Disk Space Usage Utility

This utility displays information about disk space usage. There are three routines that must be called to use this utility: DSPC1, FSORT2, and DSPC2. DSPC1 extracts the disk space usage information, FSORT2 sorts the information, and DSPC2 formats the information into report form. For more information about FSORT2, refer to the Sort/Merge chapter in Volume II of the MPX-32 Reference Manual. For more details about the steps necessary to run DSPC1, FSORT2, and DSPC2, refer to the sample TSM macro file US.DSPC.

Syntax for DSPC1

DSPC1 SYS [,U]
DSPC1 DSC, *devmnc* **[,U]**

SYS specifies system volume

DSC specifies disk

devmnc specifies the device mnemonic, channel address, and device subaddress. For example, DM0804 specifies moving head disk at channel address 08 and device subaddress 04.

[,U] specifies update access. The default is read.

Note: The disk usage information is output to a file called DS.USAGE. This file is allocated dynamically and is created in the user's current working volume and directory.

Syntax for DSPC2

DSPC2

Default Assignments

```
$ASSIGN IN TO DS.USAGE
$ASSIGN OT TO LFC=UT
```

Note: DS.USAGE must be assigned as blocked with update access to read a 64-byte entry from the file and generate reports.

2.16 FOLAP - File Overlap Detection Utility

FOLAP scans a volume for file overlaps and displays the resource descriptors (RDs) of any overlapped files. The volume being scanned cannot have any other activity to it. FOLAP can only be used on unmounted disks.

Note: This program is written in ASSEMBLE.

Syntax

FOLAP DSC,*dev*

or

FOLAP SYS

DSC	specifies disk
dev	specifies the device to scan
SYS	specifies system volume

2.17 GM.CHHST – State Chain History

GM.CHHST records state chain history, which is used to evaluate the effectiveness of the optional CPU/IPU scheduling algorithm included in MPX-32 3.3 and later versions.

To use GM.CHHST, H.EXEC2 and H.CPU2 must be assembled with the internal flag CHAINHST set true and the object compressed and SYSGENed. Also, the SYSGEN directive DELTA=*xx* must be specified. GM.CHHST records the last 256 entries (12 words each) of scheduling information and outputs this information to a file named REPORT. For more information, see the source file US.CHHST.

Syntax

GM.CHHST

2.18 GM.CLS – TERMDEF Demo

GM.CLS clears the screen on the terminal from which it is invoked. GM.CLS can optionally delay a specified number of seconds or wait for the break key (<break>) before clearing the screen.

Notes:

It is assumed that the terminal defines the clear screen function as cl.

If the TERMDEF facility has been SYSGENed out with the SYSGEN NOTDEF directive, GM.CLS returns to TSM.

This program is written in ASSEMBLE.

Syntax

GM.CLS [*delay* | 0]

[*delay* | 0]	*delay* specifies the number of seconds to wait before clearing the screen. 0 specifies wait until <break> is pressed before clearing the screen.

2.19 GM.FLOW – Assembly Source Code Flowchart Tool

GM.FLOW reads an uncompressed, nonbase assembly source code file and produces a display of all forward and backward references. The output consists of up to 29 visible levels of forward references, 72 bytes of source code, and up to 29 visible levels of backward references. This report is followed by a summary displaying unresolved references and unreferenced symbols; for example, `BU *ADDRESS,X1` cannot be resolved. This report also contains simple statistics collected during run time; for example, forward references found and records processed.

Optionally, a sequence number requiring eight spaces is placed in the source output starting in any column from 1 to 65. When used, the sequence number overlays the source output and is relative to the first record actually processed. The record where processing starts and stops can also be optionally specified.

Note: This program is written in FORTRAN 77+.

Syntax

GM.FLOW [-S[*column*]] [*start*] [*stop*]

[-S[*column*]]
: -S indicates that sequence numbers are to be placed in the text section of the output. *column* specifies a column from 1 to 65 as the starting location of the sequence number. The default is 1.

[*start*]
: indicates the record number where processing starts. Records skipped are ignored and not listed. If not specified, the default is the first record in the file. 0 indicates that the default is used.

[*stop*]
: indicates the record number where processing stops. Records after the stop are not read. If not specified, default is the last record in the file. 0 indicates the default is used.

Resource Summary

```
$AS SI TO SOURCE_FILE    (required)
$AS SO TO SLO            (default – can be reassigned)
```

Example of Output

```
          *
          ** RUN LOOP GM.FLOW DEMO
          *
         START           LW        R1,DATA
+<<<<<<<<<<<<<<<<<<<<<<<<BZ        MERGE
!        REDO                      LI R1,1<<<<<<<<<<<<<<<<<<< +
! +<<<<<<<<<<<<<<<<<<<<<<BU        HERE                       !
! !     DATA             REZ                         1W       !
+====== MERGE            BU        LOOP                       !
! +>>>> HERE             BU        REDO===================== +
+>>>>>> LOOP             TRN       R1,R4                      !
        DO               STW       R4,DATA<<<<<<<<<<<<<<<+    !
                         BIB       R4,DO>>>>>>>>>>>>>>>>>+    !
                         BU        HERE>>>>>>>>>>>>>>>>>>>>>> +
```

2.20 GM.SEAR – Source Search Tool

GM.SEAR searches for two different strings within a blocked, uncompressed source file. It searches by columns if the string starts within a predefined column range.

Output can be displayed as follows:

- the matching line with seven lines preceding and following it
- only the matching line. This display resembles edit when used for searching.
- all lines beginning at a specified line

Additional functions include:

- echoes screen data to SLO. This can be enabled and disabled.
- online help and program status inquiry
- file movement through back record, skip record, or rewind functions
- full pathname support
- comments can be sent to SLO to document echoed messages
- support of standard terminal and ANSI mode terminal screen controls

Note: This program is written in FORTRAN 77+.

Syntax

GM.SEAR [-A]

[-A] indicates that the terminal requires ANSI screen control codes

Note: Additional documentation is resident within the program. It can be examined by requesting help at the first prompt or by selecting help from the command menu.

2.21 GM.TDEMO – TERMDEF Demo

GM.TDEMO demonstrates the following TERMDEF functions: cursor addressing, clear screen, cursor hide and show, and highlighting. If a system is set up for TERMDEF, GM.TDEMO runs on all of the defined terminals.

Notes:

It is assumed that the terminal defines the screen functions cl, cm, me, md, HC, and SC. If a terminal function is not defined, it cannot be performed by GM.TDEMO.

If the TERMDEF facility has been SYSGENed out with the SYSGEN NOTDEF directive, GM.TDEMO returns to TSM.

This program is written in FORTRAN 77+.

Syntax

GM.TDEMO

2.22 INTR – Indirectly Connected Interrupt Response Timing

INTR is comprised of test modules that measure the performance of the indirectly connected interrupt module H.ICP. H.ICP and a real-time option module (RTOM) must be configured into the current system image. (The RTOM must be jumpered for a high frequency select of 300 nanoseconds.)

The INTR modules load the RTOM with a positive number of interval timer count down units and sets the RTOM to interrupt at 0. The INTR modules then go into suspension until the RTOM timer expires. The modules measure the time from the expiration of the timer to the first instruction of the indirectly connected task.

The INTR environment consists of the following:

File	Description
BLD.INTR	macro to build INTR performance tests
CAT_RESULTS	macro to concatencate results files into one file
CLEAN_UP	macro to delete unnecessary files from the SYSTEM directory after the tests are run
COPY_FILES	macro to copy necessary files to the SYSTEM directory before the tests are run
INTRPTB	J.INTRPTB load module
INTRPTC	J.INTRPTC load module
INTRPTLB	J.INTRPTLB load module
INTRPTLC	J.INTRPTLC load module
J.INTRPTB	source for timing interrupts with cache busted, CPU is idle
J.INTRPTC	source for timing interrupts with cache busted, CPU is idle
J.INTRPTLB	source for timing interrupts with cache busted and the CPU busy
J.INTRPTLC	source for timing interrupts in cache with the CPU busy
J.LOOPRT	source for the real-time task
J.STPK	source for the FORTRAN statistical package
JT.INTR	macro to run INTR performance tests
LOOPRT	J.LOOPRT load module
MOVE_RESULTS	macro to move the current results to another directory
PRINT_RESULTS	macro to print the current statistics
SG.II	example SYSGEN directives

The INTR modules run in E-class memory from the SYSTEM directory in a stand-alone environment for consistent results. The standard configuration is 32KB cache, two-way interleaved memory on a CPU-only system. The RTOM must be SYSGENed at priority 20 with subaddress 04 as shown in the following sample SYSGEN statement:

```
PRIORITY=20,RTOM=(79,04),INTV
```

For a detailed example of the SYSGEN statements required for running the INTR modules, refer to the SG.II file.

2.22.1 Running the INTR Modules

Following are the instructions for running the INTR modules:

1. Sign on to MPX-32 as SYSTEM.
2. Ensure that the INTR modules are located in directory INTR on volume SYSTEM. (If these files do not exist on the system, restore them from the SDT tape.)
3. Change directory to `@SYSTEM^INTR`.
4. Copy any required files to the SYSTEM directory with the COPY_FILES macro.
5. Change directory to `@SYSTEM^SYSTEM`.
6. Execute the INTR modules with the following statement:

   ```
   TSM>JT.INTR
   ```

7. The INTR modules produce results for all combinations of the following options: real-time accounting on (ONRA) or off (OFRA), CPU idle or busy, and in cache or cache bust. Upon completion the results are placed in the following files:

File	Description
ON.II.C	Samples – ONRA, CPU idle, in CACHE
ON.II.CS	Statistics – ONRA, CPU idle, in CACHE
ON.II.B	Samples – ONRA, CPU idle, CACHE bust
ON.II.BS	Statistics – ONRA, CPU idle, CACHE bust
ON.II.LC	Samples – ONRA, CPU busy, in CACHE
ON.II.LCS	Statistics – ONRA, CPU busy, in CACHE
ON.II.LB	Samples – ONRA, CPU busy, CACHE bust
ON.II.LBS	Statistics – ONRA, CPU busy, CACHE bust
OFF.II.C	Samples – OFRA, CPU idle, in CACHE
OFF.II.CS	Statistics – OFRA, CPU idle, in CACHE
OFF.II.B	Samples – OFRA, CPU idle, CACHE bust
OFF.II.BS	Statistics – OFRA, CPU idle, CACHE bust
OFF.II.LC	Samples – OFRA, CPU busy, in CACHE
OFF.II.LCS	Statistics – OFRA, CPU busy, in CACHE
OFF.II.LB	Samples – OFRA, CPU busy, CACHE bust
OFF.II.LBS	Statistics – OFRA, CPU busy, CACHE bust

Each file has the following columns of data:

clock ticks — the clock tick counter value

adj nsec — The adjusted time for one interrupt-to-task dispatch. (The overhead is subtracted from the raw nanoseconds to access the RTOM interval timer.)

A statistical analysis is provided after each combination. The number of values and the percentage of those values falling between the average (plus or minus one standard deviation and those outside one standard deviation) are shown. The interrupt-to-task dispatch value are in microseconds.

The following macros can be used on the individual result files:

- PRINT_RESULTS – prints all of the individual file
- CAT_RESULTS – concatenates the individual files into one results file, RES.INTR
- MOVE_RESULTS – moves a file to another directory. (MOVE_RESULTS cannot move the RES.INTR file.) The volume and directory names where the file is to be moved must be passed as parameters to the MOVE_RESULTS macro as follows:

  `TSM>MOVE_RESULTS` *volume directory*

8. Remove unnecessary files from the SYSTEM directory with the CLEAN_UP macro in the INTR directory. This macro does not delete RES.INTR.

Notes:

To recompile the INTR modules, use the BLD.INTR macro and pass as a parameter the name of the FORTRAN compiler and the run-time library files to be used as shown in the following example:

```
TSM>BLD.INTR FORT45NN SRTL45NN SRTD45NN
```

The INTR modules provided on the SDT tape have been built with FORTRAN 77+ 4.4 and Macro Assembler 3.0.

2.23 LASER – Laser Printer Utility

LASER (US.LASER) is a conversion of the SPRINT program that provides minimal support for a laser printer (in this case, a DataProducts LZR-2600 with a board that emulates a Daisy Wheel).

LASER allows vertical or horizontal printing and can be exited (X) at most prompts. For more information, refer to the SPRINT documentation in this section.

Syntax

LASER

2.24 LMINFO – Load Module Information

LMINFO displays load module information from the preamble after receiving responses from the following prompts:

```
INPUT THE VOLUME THAT THE LOAD MODULE RESIDES ON:
INPUT THE DIRECTORY THAT THE LOAD MODULE RESIDES ON:
INPUT THE LOAD MODULE NAME:
```

Syntax

LMINFO

2.25 LOGMDT – Log Contents of Rapid File Allocation MDT

LOGMDT displays the address, size, and contents of the rapid file allocation memory resident descriptor table (MDT) area. It can include a hexadecimal dump of all active and inactive resource descriptor (RD) entries in the MDT.

LOGMDT gets and displays the MDT address and entry parameters from the MPX-32 communications area. If the MDT base address is non-zero, LOGMDT displays the pathnames and entry numbers of files with entries (active or inactive) in the MDT. Entry numbers are typically non-linear because RD entries are hashed into the MDT. LOGMDT checks all of the possible entry positions for any non-zero word, and reports any entry position that is not zeroed.

If the MDT is not configured (the base address equals zero), LOGMDT terminates after displaying the address and entry parameters.

Note: This package consists of a main program written in FORTRAN 77+ and three subroutines written in ASSEMBLE, and uses only native services. The program runs in privileged mode.

Syntax

LOGMDT [V]

[V] specifies verbose. Each entry logged includes the file's owner name, project name, file type, and a hexadecimal dump of the file's 192-word RD.

Resource Summary

Unit LO log information and error output. The default is to LFC=UT (static, reassignable). For hard-copy output, use `$ASSIGN LO TO SLO.`

Error Reporting and Termination

There are no internal aborts. The program reports error messages and ends execution for the following conditions:

- a consistency check of two MDT "C." parameters fails
- the internal buffers cross map block boundaries
- improper status from external subroutines

Example – MDT with two active entries

```
**************************************************
LOGMDT date LOG RAPID FILE ALLOCATION - MDT -

MDT INSTALLED, CURRENT PARAMETERS ARE:

MDT BEGIN ADDRESS (REAL) = 001E8000 HEX
MDT TOTAL LENGTH (BYTES) = 00006000 HEX 24576 DEC
MDT END ADDRESS (REAL)   = 001EE000 HEX

# MDT ENTRIES CURRENTLY USED  =  2
# ENTRIES NOW AVAIL.  (FREE)  = 30
TOTAL # ENTRIES (MDT SIZE)    = 32

ENTRY # 27, ACTIVE !
VOL/DIR/FIL = MPX3.x    RAPFILE        MT02

ENTRY # 28, ACTIVE !
VOL/DIR/FIL = MPX3.x    RAPFILE        MT01

ENTRIES NOT LOGGED ARE ALL ZEROS !!
A TOTAL OF   32 ENTRIES SEARCHED.

********************END OF LOGMDT*****************
```

Example – MDT with one active entry and one inactive entry

```
**************************************************
LOGMDT date LOG RAPID FILE ALLOCATION - MDT -

MDT INSTALLED, CURRENT PARAMETERS ARE:

MDT BEGIN ADDRESS (REAL) = 001E8000 HEX
MDT TOTAL LENGTH (BYTES) = 00006000 HEX 24576 DEC
MDT END ADDRESS (REAL)   = 001EE000 HEX

# MDT ENTRIES CURRENTLY USED  =  1
# ENTRIES NOW AVAIL. (FREE)   = 31
TOTAL # ENTRIES (MDT SIZE)    = 32

ENTRY # 27, *** NOT ACTIVE ***
VOL/DIR/FIL = MPX3.x      RAPFILE       MT02

ENTRY # 28, ACTIVE !
VOL/DIR/FIL = MPX3.x      RAPFILE       MT01

ENTRIES NOT LOGGED ARE ALL ZEROS !!
A TOTAL OF   32 ENTRIES SEARCHED.

*********************END OF LOGMDT****************
```

2.26 LOGOFF – Remote Terminal Logoff

LOGOFF terminates outstanding I/O at a specified TY address and logs off the current user. UDT linked devices must also be TSM devices for this program to run. The source code (US.LOGO) can be modified to specify system administrator use only. This program runs privileged only when required.

Note: This program is written in ASSEMBLE.

Syntax

LOGOFF TY*ccss*

cc	is the channel
ss	is the subchannel

Aborts

US01	FATAL ERROR DETECTED. SEE SLO OR UT FOR MORE INFORMATION.
US02	UNRECOVERABLE I/O ERROR ON TERMINAL

2.27 LOOK – String Search

LOOK searches for a string (up to 16 characters) in a file or a series of files. The files to search are read in on LFC SYC. The string to search for is read in on UT.

Syntax

LOOK

Default Assignments

```
$AS SYC TO SYC
$AS SO TO LFC=UT
```

2.28 MIPSMON – Instruction Sequence Timing Tool

MIPSMON enables a user to time individual instruction sequences with great accuracy. Instruction sequences are defined in an assembly language program using MIPSMON macros. This program is linked with the MIPSMON Executive to form a load module which is then run in a stand alone environment. The instructions are timed using an RTOM. An output file is produced. Another program, MIPREP, reduces the output file and presents the results of the measurement in terms of time (nanoseconds) and machine cycles.

MIPSMON contains the following software:

File	Description
MIPSMONX32.M	MIPSMON base object module
MIPS_LIB	MIPSMON nonbase object library
MIPS_DIR	MIPSMON nonbase object directory
MIPMACX32	base mode macro library
MIPMAC	nonbase mode macro library
MIPMSYS	MIPS system JCL
MIPREP	MIPS report generator

The user interface to MIPSMON consists of a JCL file, MIPSYS, and a load module, MIPREP. MIPSYS reads an instruction sequence file containing the instruction sequences to be timed. It outputs a load module. When the load module is run, an output file is produced which contains the results of the timing. MIPREP produces a report from this output file.

MIPSYS Syntax

MIPSYS *file*,*lm*,[*base*],[*defout*]

file	is the instruction sequence file that is input
lm	is the load module that is output
[*base*]	indicates base register instructions. The default is nonbase register.
[*defout*]	is the output file. If omitted, the timing results are output to MIP.OUT.

MIPREP Syntax

MIPREP *defout* [**>PRINT**]

defout is the output file

[**>PRINT**] indicates that output is to the printer. Default is to the terminal.

An instruction sequence file is an assembly language program that contains the instructions to be timed. The file also contains MIPSMON macros which direct the MIPSMON Executive to time the instruction sequences as specified.

Instruction Sequence File Format

```
PROGRAM prgname
MIPINIT
 .
 .
 .
one or more sequences to be measured
 .
 .
 .
M.DONE
END
```

`MIPINIT` initializes certain data structures and assembly variables. MIPINIT must appear once in the file under the PROGRAM statement. There are no parameters for this macro.

`M.DONE` must appear once in the file following the last instruction sequence to be measured

Instruction Sequence Format

```
NEW name
M.INIT
(initialization instructions)
TRSW 0
M.INITL
(initialization instructions)
TRSW 0
M.INST
(instruction sequence to be measured)
M.INSTE
```

`NEW` introduces each new measurement sequence and assigns it an identifier (*name*) which is printed in the report. *name* is a 1- to 8-character string.

`M.INIT` introduces a subroutine which is called before entering the timing loops (dummy and real). M.INIT is optional. It should contain any loop independent initialization required by the sequence being timed. The subroutine is called by a branch and link and must contain a TRSW 0 to return.

`M.INITL` introduces a subroutine which is called within the timing loops (dummy and real). M.INITL is optional. It should contain any loop dependent initialization required by the sequence being timed. The subroutine is called by a branch and link and must contain a TRSW 0 to return.

`M.INST` introduces the sequence to be timed. There are no parameters for this macro.

`M.INSTE` indicates the end of the sequence to be timed. There are no parameters for this macro.

MIPSMON inserts the user's instructions into a sequence of instructions that block interrupts and clear the instruction pipeline so the user's instructions can be timed without interference. The sequence is executed repeatedly and then averaged to obtain an accurate value.

Notes:

The timing of instruction sequences is dependent on input values. An input value of 0 should not be used because it does not represent typical instruction processing.

M.INIT and M.INITL can be used to avoid underflow and overflow conditions which lead to incorrect timing statistics.

Example

The following example is a sequence file that measures the performance of the load word (LW) instruction. The sequence contains a LW instruction, two multiply floating point word instructions, and a store word instruction. The M.INITL routine initializes the variable T to avoid the overflow which occurs from continued multiplications. By resetting T to its original value, the sequence of instructions is timed with the same input values each time.

```
                       PROGRAM LWTIME
                       MIPINIT
*
* INITIALIZATION DATA
*
T                      DATAW E'2.1'
TP                     DATAW E'2.1'
E                      DATAW E'1.877493'
J                      DATAW E'.499975'
*
* LW/MPFW/STW TIMING
*
                       NEW TIMING
*
                       M.INITL
                       LW   3,TP
                       STW  3,T
                       TRSW 0
*
                       M.INST
                       LW   3,T
                       MPFW 3,E
                       MPFW 3,J
                       STW  3,T
                       M.INSTE
*
                       M.DONE
                       END
```

MIPSMON allows general instruction sequences with the following exceptions:

- The values in R7, B4, B5, and B6 cannot be changed.
- The STRUCT label is reserved for MIPSMON.
- The value in R0 must be preserved.
- There cannot be more than 100 words between M.INST and M.INSTE.
- Branches require special handling.

2.29 NEWCOPY – Floppy Disk Duplication Tool

NEWCOPY performs the following functions:

- archives floppy disks to a hard disk
- duplicates floppy to floppy
- copies from archive to floppy disk
- performs general maintenance on archives
- formats floppy disks

Note: This program is written in FORTRAN 77+.

Syntax

NEWCOPY

Default Assignments

`$ASSIGN LO TO SLO`	(audit trail)
`$ASSIGN OT TO LFC=UT`	(prompt/message output)
`$ASSIGN IN TO LFC=UT`	(command input)

All other assignments for resources are dynamic.

Aborts

Examine source file for details.

2.30 POOLSCAN – Memory Pool Monitor

POOLSCAN monitors the current usage and state of memory pool. It can be used to maintain a historical profile of memory pool usage. The profile can be used to project memory pool requirements. There are three types of memory pool that can be selected: miscellaneous, IOQ, or MSG.

POOLSCAN uses an updating screen output method that is compatible with the following terminals: ADM-3A, Hazeltine 1500, TeleVideo 910, TeleVideo 921, and TeleVideo 9220. It can work with other terminals as well.

Syntax

POOLSCAN [*del*] **[,H|,D] [,-A] [IOQ|MISC|MSG]**

[*del*] specifies a time delay of 0 to 60 seconds between screen updates. If not specified, the default is ten seconds.

[,H|,D] H specifies that the display is hexadecimal (default). D specifies that the display is decimal.

[,-A] specifies that the terminal requires ANSI control codes, such as TeleVideo 922 or TeleVideo 970

[IOQ | MISC | MSG]
specifies the type of memory pool. IOQ is I/O queue memory pool, MISC is miscellaneous memory pool (default), and MSG is message memory pool.

Notes:

If a delay time of 0 is used, POOLSCAN degrades system performance because it changes its priority level and does system context switch inhibits while scanning memory pool.

POOLSCAN runs in the privileged mode when scanning memory pool. Otherwise, POOLSCAN runs in the unprivileged mode.

Press <break> to terminate POOLSCAN.

POOLSCAN cannot be run in the batch mode or on a hard copy device.

It is not necessary to set the page size to 0 before running POOLSCAN. A combination of overprint and other carriage controls avoid line counting.

Aborts

FREE	indicates that a bad linkage was detected in the memory pool free linked list (only forward pointer checked)
IOQ	indicates that IOQ memory pool was requested but it is not configured in the system
MSG	indicates that MSG memory pool was requested but it is not configured in the system
USED	indicates that a bad linkage was detected in the memory pool used linked list (only forward pointer checked)

2.31 PORTPROT – Dial-up Port Protection

PORTPROT adds an extra level of system protection by requiring passwords for selected terminal ports. Documentation for customizing and protecting an individual system is located in the source file.

2.32 RNVOL – Rename Volume

RNVOL changes the name of a formatted disk volume. This utility functions on an unmounted volume only.

Syntax

RNVOL *devmnc currname newname*

devmnc specifies a 6-character device specification (refer to Appendix A) of the formatted volume to be renamed

currname specifies the current volume name

newname specifies the desired volume name

2.33 RT_DEBUG – Realtime Debugger

RT_DEBUG allows the user to dynamically observe and modify memory that is in a global common area. Global common variables can be addressed symbolically, complete with array subscripts, and by absolute address. Wildcard and subrange subscripts are provided for up to three-dimensional arrays. Display attributes can be saved on disk and recalled at a later time. RT_DEBUG allows for the display of up to 30 items on the screen at any time.

The user must provide a common definition file (COMDEF) for all variables to be viewed. COMDEF contains the declarations (type and dimensionality) for all the variables that reside in the shared partition, including the COMMON statement, and must conform to standard FORTRAN syntax. COMDEF must be included with the application source as it is compiled, via the INCLUDE statement. RT_DEBUG prompts the user for the partition definition and the COMDEF definition from which it allows the user to dynamically display and change variables that reside within the application's shared partition.

RT_DEBUG supports the TeleVideo 910, 921, 922, and 9220 terminals. Since program source is provided and the terminal drivers are isolated to three subroutines, the user can customize the drivers for use with any terminal that allows direct cursor addressing.

2.33.1 Preparing to Run RT_DEBUG

The following steps prepare RT_DEBUG for execution:

1. Refer to the MPX-32 source directory file US.RTDBG for the JCL to compile and catalog the program. This file creates the RT_DEBUG load module.
2. Recompile RT_DEBUG if changes have been made to the RT_DEBUG source.

2.33.2 Running RT_DEBUG

To run RT_DEBUG enter:

RT_DEBUG *pathname*

pathname specifies the pathname of the target task to be debugged

If *pathname* is not specified, the following prompt displays on the screen:

```
PLEASE ENTER PATHNAME OF THE TARGET LOAD MODULE
```

If *pathname* is specified, the following prompts display on the screen:

```
PLEASE ENTER PATHNAME OF THE SHARED PARTITION.
ARE THERE ANY MORE SHARED PARTITIONS TO INCLUDE (Y/N)
PLEASE ENTER PATHNAME OF THE COMMON DEFINITION FILE
PLEASE ENTER TERMINAL TYPE
```

The screen clears and the following heading displays across the top of the screen:

```
VARIABLE PAG ROW COL VALUE VARIABLE PAG ROW COL VALUE OFF
```

The cursor displays at the bottom of the screen after the `Command:` prompt.

Note: If the user enters a <break> at the `Command:` prompt, I/O terminates and RT_DEBUG aborts.

RT_DEBUG is ready to display data on the screen. The data is displayed in two columns with 15 values per column. The variable name (or its address) displays under the VARIABLE heading. If the variable is an array, the page, row and column subscripts display under the PAG ROW COL headings. The decimal value of the variable displays under VALUE. For example, scalar variable `SCA`, vector variable `MYVEC(3)`, matrix `M(2,4)` and location `X'7FEC0'` display as follows:

```
VARIABLE     PAG ROW COL VALUE VARIABLE PAG ROW COL VALUE OFF
SCA          ... ... ...    23
MYVEC                  3   742
M                  2   4     0
#7FEC0       ... ... ...   242
```

In this example, `SCA`, `MYVEC(3)`, `M(2,4)`, and location `7FEC0` have values of `23`, `742`, `0` and `242` (decimal), respectively.

RT_DEBUG displays and modifies INTEGER*1, INTEGER*2, INTEGER*4, INTEGER*8, REAL*4, REAL*8, LOGICAL*1 and LOGICAL*4 values. Absolute addressed items are treated as words; i.e., INTEGER*4.

The word OFF on the far right-hand side of the heading means the Bell (B) status is OFF (see the BELL command.)

RT_DEBUG updates the screen if any of the displayed items change value. For example, when the target task modifies these locations.

2.33.3 Display Mode

In display mode, RT_DEBUG constantly scans memory and compares the data with the contents of memory when the display item was first posted on the screen. If they are not equal, RT_DEBUG updates the screen and the saved value. (The screen is updated only when a change occurs; this gives a smaller time resolution in detecting memory value changes.)

2.33.4 Command Mode

To enter command mode, press the <break> key on the terminal. (Use <ctrl><break> for a TVI921 terminal.)

RT_DEBUG displays the following prompt:

```
COMMAND:
```

Enter one of the commands. For example, `@Z`<ret> executes the ZERO command. Commands (and all other user input to the terminal) can be in upper or lower case.

2.33.5 QUIT Command

The QUIT command exits RT_DEBUG.

Syntax

@Q

Examples

Exit RT_DEBUG with the following:

<break>
`@Q`

RT_DEBUG displays the following prompt:

```
Exiting are you sure? (Y/N)
```

Enter Y (yes) to exit RT_DEBUG, or N (no) to continue RT_DEBUG.

2.33.6 BELL Command

The BELL command toggles the terminal bell. When the bell is on and an item changes in the display list, the bell sounds. (When the bell is on, the update time on some terminal screens may slow down.)

Syntax

@B

2.33.7 CHANGE MEMORY Command

The CHANGE MEMORY (=) command changes the contents of a memory location. A location can be addressed symbolically or by absolute address. This command can change any variable listed in the COMDEF file whether it is currently displayed or not.

Syntax

address={ [**H**] *number* | *boolean* }

address specifies the memory location to be changed. *address* can be a symbolic or absolute address.

{ [**H**] *number* | *boolean* }
number is a decimal value, unless H (hexadecimal) is specified. Enter *number* without regard to decimal points or FORTRAN format statements. *boolean* is T (true) or F (false).

Examples

To change `M(2,4)` to `77`, use the following command sequence:

```
<break>
M(2,4)=77
```

To change more than one array item at a time, specify wildcard and subrange subscripts within parentheses. For example, to change the values of VECTOR(3), VECTOR(4) and VECTOR(5) to `142.6`, enter the following:

```
VECTOR(3:5)=142.6
```

To change all values of `VECTOR` to `0.123`, enter the following:

```
VECTOR(*)=0.123
```

Up to three-dimensional arrays can be addressed. Subranges, wildcards and constants can be mixed. For example, to change pages `2` through `5`, all rows and column `6` of matrix `A` to `19`, enter the following:

```
A(2:5,*,6)=19
```

To change all `XMAT` values to `0.005`, enter the following:

```
XMAT(*,*)=0.005
```

Values can be entered in hexadecimal by preceding the value with the letter H. For example to change all values of `IVEC` to `255` decimal, enter the following:

```
IVEC(*)=HFF
```

2.33.8 DISPLAY Command

The DISPLAY command allows the user to add a new item to the display list. Items are added by specifying their symbolic name or address. The user is prompted for any additional information that may be required.

Syntax

{*symbolic name* | *address*}

symbolic name
is the symbolic name of the item to add to the display list

address is the address of the item to add to the display list

Examples

Add `MAT(1,9)` to the display list with the following:

```
<break>
MAT(1,9)
```

Add location `7FEC0` to the display list with the following:

```
<break>
#7FEC0
```

Note: The wildcard and subrange addressing of arrays as described in the CHANGE MEMORY command applies to this command also. For example, to display all of the elements of `IVEC`, respond with `IVEC(*)`. To display the first two columns and all rows of matrix `XYZMAT`, enter `XYZMAT(*,1:2)`.

2.33.9 READ Command

The READ command allows the user to read a screen from disk. This screen must have been previously saved with the WRITE command. The user is prompted for a filename and the disk is examined to make sure the file exits. When reading in a screen, its contents are appended to the existing display screen. (To start with a clear screen, use the ZERO command prior to the READ command.) No more than 30 items can be displayed on any screen.

Syntax

@R

Examples

To read in a screen stored in file `MYSCREEN` and add this to the existing display screen, use the following command sequence:

```
<break>
@R
```

RT_DEBUG displays the following prompt:

```
ENTER FILENAME
```

In response to the prompt, enter the following:

```
MYSCREEN
```

RT_DEBUG appends `MYSCREEN` to the display list and returns to display mode.

2.33.10 WRITE Command

The WRITE command allows the user to save the contents of the current screen to disk. RT_DEBUG creates a new file for each screen and prompts for a filename.

Syntax

@W

Examples

The following example saves the current screen in a file called `SCREEN1`:

```
<break>
@W
```

RT_DEBUG displays the following prompt:

```
ENTER FILENAME
```

In response to the prompt, enter the following:

```
SCREEN1
```

RT_DEBUG writes the current screen to disk and returns to display mode.

2.33.11 ZERO Command

The ZERO command deletes the current display list from the screen.

Syntax

@Z

2.33.12 Reading and Writing Screens

There are two types of items that can be displayed on the screen: the contents of symbolic variables and the contents of addressed memory locations. Whenever a variable name is added to the display list, the variable must be mapped into a logical memory location. (Addressed memory is not mapped.) The address is then rounded down to the nearest word location for display.

When RT_DEBUG writes a screen to disk, variables are written with their symbolic names and subscripts. When a screen is read from disk, variable names and subscripts are read and remapped. This means that the user can write a screen to disk, modify and recompile the target program, and read the screen. All variables are remapped to new locations.

With addressed memory, the actual address is written to disk and reread from disk into the display list. If the target program has been modified and recompiled, the address displayed on a previous screen may now be incorrect, since the address constant is not remapped with input to disk.

2.33.13 RT_DEBUG Restrictions

The following is a list of the RT_DEBUG restrictions:

- Extended partitions are not supported.
- If a partition inclusion procedure fails with `IOSTAT=16` reported, the partition conflicts with RT_DEBUG address space. This means that the logical address of the partition overlaps RT_DEBUG address space.
- Only non-base register FORTRAN tasks are supported. This is because the dynamic data declaration file, COMDEF, must be in the form of a FORTRAN include file. (This file must be syntactically correct.) RT_DEBUG uses this file to determine if a symbol is valid, its dimensions, and if it is in the common block.
- The target debug task must be compiled with FORTRAN option 19 set. The task is then cataloged with option 19 set. This allows RT_DEBUG to access the symbol table information for the target task.
- The following data types are not supported: character, complex *8, and complex *16.

2.34 SDUTIL – Small Computer System Interface (SCSI) Disk Utility

SDUTIL is a menu-driven utility that provides additional media management functions for SCSI disks. It is assumed that the user is familiar with the SCSI common command set (CCS). For more information about the SCSI CCS, refer to ANSI SCSI Committee Working Document X3T9.2/85-52. (SDUTIL is based on Revision 4B of this document.) Following is a list of the functions that SDUTIL provides in the first menu:

- defect data management – allows you to move data contained on flawed media to good media.
- mode sense/select – displays drive operation parameters. This option can be used to override default parameters set by J.VFMT. If the drive rejects a request to change a given parameter, SDUTIL displays the message `ILLEGAL REQUEST` to the terminal.
- format – used with the mode sense/select option to format SCSI disk drives. SCSI CCS uses the term format to refer to media initialization. This differs from a J.VFMT FORMAT directive which writes disk volume structures to properly initialized disks.
- display defect data – displays primary and grown media flaws in either block number or byte offset from index format.
- diagnostic read/write data – performs a diagnostic check that reads and optionally rewrites data blocks.
- change logical block size – changes the logical block size. For an MPX-32 environment, this size can be 256 bytes or 768 bytes. A block in SCSI CCS terminology is equivalent to an MPX-32 sector.
- inquiry – displays device information such as media type, version information, vendor identification, product identification and revision level.
- read capacity – displays the total number of logical blocks and the size of each block in bytes.
- read MFP – displays the MFP model number, the firmware model number, and the firmware revision level.

Syntax

```
TSM>SDUTIL
SDU>SCSI DEV=DMccss
```

cc is the channel address

ss is the device address

Note: If SDUTIL encounters an error in the directive line, processing ends and returns to TSM.

2.35 SPRINT – Serial Printer Formatter/Spooler

SPRINT formats data for a serial printer that has been connected to an 8512 (-1 or -2) eight-line asynchronous controller. While SPRINT does not support all the functions of the J.SOUT spooler, it provides a clean interface to a serial printer. SPRINT runs with NEC 7715 and NEC 8815 printers and the H.F8XIO handler. It uses WXON and REMOTE as additional J.TINIT parameters.

Note: This program is written in FORTRAN 77+.

The form control characters that can be processed by SPRINT in the formatted input mode are:

1	form feed
0	double space
+	overprint
-	title. The first dash (-) causes a form feed. Other dashes have no effect until a line without a title character is found. Title lines are not retained and printed at each top of form as with J.SOUT.

All other form control characters are treated as a single space.

Syntax

SPRINT

Prompts

`PATHNAME TO PRINT ...........:`
Enter any uncompressed file where the user has read access. The records can contain up to 132 bytes; any excess is truncated.

`FORMATTED ? (Y/N) .........:`
Enter Y to indicate that column one should be interpreted as a carriage control character. Enter N to indicate that the first byte is data to be printed. The default is N.

`EMBEDDED FORMS CONTROL ...:`
Enter Y to indicate that the data in the file contains embedded form control characters (files output from ROFF contain embedded form control characters). Internal line counting is inhibited. Y is normally entered only with the FORMATTED=N mode. Enter N for embedded form control characters to have no effect. The default is N.

`DEVICE ADDRESS .......:`
Enter the device address where the printer is configured or the default that is defined at compile time.

`COPIES (1 TO 20) .........:`
Enter the number of copies to print of the specified file. When making large numbers of copies, the terminal is tied up for an extended period of time.

`LINE LENGTH (1 TO 133) ..:`
Enter the maximum line length to output to the serial printer. This value must not exceed the line size defined for the device.

Aborts

`RT10` The value entered at the line length prompt exceeds the defined line size.

2.36 SRCH – Interactive String Search

SRCH is an interactive program that searches for a string in an uncompressed source file.

Syntax

SRCH [-A]

[-A] specifies ANSI terminal

Default Assignment

```
$AS DO TO LFC=UT
```

2.37 SW.CHART – Swapper Percentage Active Monitor

SW.CHART (JCL source file US.CHART) is used with J.SWAPR version 3.3.16 or later to display the percentage of time that SWAPPER is active per second. The information is output as a bar graph.

Note: Use of this program with a version of J.SWAPR prior to 3.3.16 can result in unpredictable system behavior, including system crashes.

Syntax

SW.CHART

or

$AS OUT TO DEV=TY*nnnn*
$ACTI SW.CHART

nnnn specifies the hexadecimal number assigned to TY at SYSGEN

2.38 SW.MODI – Interactive Swap Parameter Modifier Program

SW.MODI allows the user to interactively modify the swap parameters defined in SJ.SWAPR2. This enables users to test swap parameters without modifying J.SWAP2, and without rebuilding and restarting MPX-32.

Syntax

SW.MODI

Notes:

It is recommended that SW.MODI be used when there is minimal swap activity.

2.39 SW.MON – Swap Monitor Program Version 2

SW.MON is used with J.SWAPR version 3.3.16 or later to display outswaps, inswaps, and shared memory includes. SW.MON does not require any special preparations to run it, unlike SWAPMON which uses SYSTEM TRACE. Refer to the SWAPMON - Swap Monitor Program Version 1 section.

Notes:

Use of this program with a version of J.SWAPR prior to 3.3.16 can result in unpredictable system behavior, including system crashes.

This program was written in FORTRAN 77+.

Syntax

SW.MON

or

$AS 6 to DEV=TY*nnnn*
$ACTI SW.MON

nnnn specifies the hexadecimal number assigned to TY at SYSGEN

2.40 SWAPMON – Swap Monitor Program Version 1

SWAPMON monitors J.SWAPR activity and displays outswaps, inswaps, and shared memory includes. If the memory partition was not cleared, it also reports on previous conditions, such as the conditions prior to a system crash.

SWAPMON includes a trace partition that must be configured for system trace as described in the MPX-32 Technical Manual, Volume I, Chapter 6. J.SWAPR1 and SH.IP06 must be reassembled with C.TRACF set true in the PRE file to activate the logic needed for the data collection.

Syntax

SWAPMON [C] [N]

or

$AS 6 TO DEV=TY*nnnn*
$ACTI SWAPMON

[C]	specifies clear the trace buffer
[N]	specifies exit when the trace buffer is empty
nnnn	specifies the hexadecimal number assigned to TY at SYSGEN

2.41 SYSINFO – Display Communications Region

SYSINFO is an interactive task which provides a static display of some of the information from the communications region on the user's terminal.

Syntax

SYSINFO

2.42 TABS – Change Tabs

TABS is used to change or display terminal tabs. There are pre-defined settings as well as user-defined settings.

Syntax

TABS [FORT|SHOW|STAN| *tab,tab...* **|TENS]**

FORT specifies the following FORTRAN tabs: 6, 7, 10, 13, 16, 19, 22, 25

SHOW displays the current tabs. Default.

STAN specifies the following standard ASSEMBLE tabs: 10, 20, 36, 57, 65, 68, 72, 80

tab specifies a user-defined tab. Up to 8 tabs can be specified.

TENS specifies tabs at the following intervals: 10, 20, 30, 40, 50, 60, 70, 80

2.43 TERMLOAD – Terminal Initializer/Loader

TERMLOAD sends predefined control data to a terminal to set up a software controllable operation mode and/or load soft function keys. It can be run in the interactive or independent modes. In the interactive mode, the user specifies a command file and loading options by the command line. The result of command file processing is sent to the terminal where activation was performed. If activated independently by OPCOM or SYSGEN, the system file M.CTLOAD is sent to the system console. If SYSGEN activation is desired, use the SEQUENCE directive for best results. If the file does not exist, nothing is done.

Note: This program is written in FORTRAN 77+.

Syntax

TERMLOAD [-] *file*

[-] inhibits a hardcopy audit trail from being spooled to SLO by LFC LO. If errors are encountered, a spool file is still created.

file is the pathname of an uncompressed file that contains commands for the TERMLOAD processor

Command File Syntax

cmd arg[,*arg*] [,'*str*'] [*com*]

cmd is a 3-character command followed by a blank or a comment character (!,*, or C) in the first byte. See Command Summary.

arg is a 1- or 2-character argument terminated by a comma or a blank (or byte 72) to indicate the end of the argument list. 2-character arguments are processed as ASCII/hexadecimal, and 1-character arguments are plain ASCII characters.

[,'*str*'] *str*']'u>(550u+1n) .br is ASCII text started by a single quote and terminated by another quote or EOL (byte 72). This text can be X'20' to X'7E', excluding the single quote character X'27'.

[*com*] is a comment following the argument list termination blank

Command Summary

ABS *arg* | *str* [,*arg* | *str*]...[*com*]
indicates absolute load. Sends the data in the argument list (without any leading or trailing data) to the unit connected to LFC OT.

DEL *arg* [*com*]
delays command processing for the number of seconds specified by *arg* (hexadecimal). *arg* can be from X'1' to X'FF' seconds.

DEV *arg* | *str* [*com*]
indicates device redefinition. Changes the target device to another TY device, such as TY7EA0.

END [*com*] terminates the command file

HED *arg* | *str* [,*arg* | *str*]...[*com*]
indicates header definition. Retains the argument list data (unless overwritten) and uses this data as a prefix for the data supplied by the MID directive.

MID *arg* | *str* [,*arg* | *str*]...[*com*]
indicates middle definition. Prefixes the argument list data with the current HED data and suffixes the argument list data with the current TER data. The result is retained and sent to the unit connected to LFC OT. If HED and TER directives have not been performed previously or if the sum of the HED, TER, and MID argument lists exceeds 80 bytes, TERMLOAD issues a warning error and ignores the directive.

MSG [*com*] echoes any text following the MSG directive (through byte 72) to the unit connected to LFC OT.

REM [*com*] treats any text following the REM directive as a comment (!, *, and C are also valid)

REP *arg* [*com*]
repeats the last valid ABS or MID directive the number of times defined by *arg* (hexadecimal). *arg* can be from X'1' to X'FF'.

TER *arg* | *str* [,*arg* | *str*]...[*com*]
indicates terminator definition. Retains the argument list data (unless overwritten) and uses this data as a suffix for data supplied by the MID directive.

Resource Summary

Unit SI — is a dynamically allocated file that contains uncompressed 80 byte records, numbered or unnumbered. There is no default (specified by directive line) unless it is OPCOM or SYSGEN activated, so SI will be assigned to the system file M.CTLOAD.

Unit LO — is an audit trail and error message unit. Any message, except for an error message, can be suppressed. The default is to SLO (static, reassignable).

Unit OT — is the destination for directive output. The default is to UT (user's terminal, dynamic, or device CT if SYSGEN or OPCOM activated).

Notes:

All of the directives can have online comments, as long as they are preceded by the argument list termination blank.

Errors always produce output to unit LO.

All directives start in the first byte position and end in byte position 72.

Directive File Example

`*`	
`** Termload example`	
`*`	
`ABS 2B,1A,1B,1C,08,20,0D`	Clear the screen
`MSG` The first three function keys of the terminal are about to be loaded	
`DEL 02`	Wait a few seconds
`HED 1B,1B,|`	Define lead in for the terminal
`TER 19`	and terminator
`MID 1,1,!,L,I,S,T,0D`	OPCOM LIST in key one
`MID 2,1,E,X,I,T,0D`	EXIT directive in key two
`MID 3,1,'SIGNAL',0D`	SIGNAL directive in key three
`ABS 2B,1A,1B,1C,08,20,0D`	Clear the screen
`ABS 2B,07,0D`	Load a beep
`REP 10`	Send it 16 times
`DEV 'TY7EC3'`	Change devices
`REP 10`	Send it 16 beeps
`MSG LOAD COMPLETE`	
`END`	End

2.44 TSCAN – TSM Scanner Demo

TSCAN demonstrates the TSM scanner (M.TSCAN). TSCAN accepts input and then displays the M.TSCAN output with additional information about length and delimiters. For more information, refer to the System Services chapter in Volume I of the MPX-32 Reference Manual.

Syntax

TSCAN [*string*]

[*string*] is a 1- to 16-character string

2.45 UDTS – Display UDT

UDTS displays information about all or a specified UDT entry.

Syntax

UDTS [*devaddr*]

[*devaddr*] specifies device address without device type specified

Default Assignment

```
$AS OUT TO LFC=UT
```

2.46 US.ERR – Unsupported Software Error Codes

US.ERR is an error file for the unsupported software abort codes and is searched for US*nn* aborts.

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* [**BLO**CKED={**Y** | **N**}]
[**DEN**SITY={**N** | **P** | **G** | **800** | **1600** | **6250**}] [**ID=***id*] [**MUL**TIVOL=*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)

[**BLO**CKED={**Y** | **N**}]
if Y is specified, medium is blocked. If N is specified, medium is not blocked. If not specified the default is blocked.

[**DEN**SITY={**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).

[**MUL**TIVOL=*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 TEMP**[=(*volname*)] [**ACC**ESS=([**R**EAD] [**WR**ITE] [**M**ODIFY] [**U**PDATE] [**A**PPEND])]
[**BLO**CKED={ **Y** | **N** }] [**PRI**NT | **PUN**CH] [**SIZ**E=*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] [**WR**ITE] [**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

[**SIZ**E=*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 Advance File	1,X'33' 1,X'34'	H.IOCS,7 H.IOCS,8	6.2 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 Execution	M.SUME M_SUME	1,X'53' 1,X'53'	H.REXS,16 H.REXS,16	6.2 7.2
Rewind File	M.RWND M_REWIND	1,X'37' 1,X'37'	H.IOCS,2 H.IOCS,2	6.2 7.2
Rewrite Descriptor	M.REWRIT M_REWRIT	2,X'2B' 2,X'2B'	H.VOMM,12 H.VOMM,12	6.2 7.2
Rewrite Descriptor User Area	M.REWRTU M_REWRTU	2,X'32' 2,X'32'	H.VOMM,27 H.VOMM,27	6.2 7.2
Scan Terminal Input Buffer	M.TSCAN M_TSCAN	1,X'5B' 1,X'5B'	H.TSM,2 H.TSM,2	6.2 7.2
Send Message to Specified Task	M.SMSGR M_SMSGR	1,X'6C' 1,X'6C'	H.REXS,44 H.REXS,44	6.2 7.2
Send Run Request to Specified Task	M.SRUNR M_SRUNR	1,X'6D' 1,X'6D'	H.REXS,45 H.REXS,45	6.2 7.2
Set Asynchronous Task Interrupt	M.ASYNCH M_ASYNCH	1,X'1C' 1,X'1C'	H.REXS,68 H.REXS,68	6.2 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 Resource Lock	M.LOCK M_LOCK	2,X'44' 2,X'44'	H.REMM,23 H.REMM,23	6.2 7.2
Set IPU Bias	M.IPUBS M_IPUBS	2,X'5B' 2,X'5B'	H.REXS,82 H.REXS,82	6.2 7.2
Set Option Lower	M.SOPL M_SOPL	2,X'77' 2,X'77'	H.TSM,13 H.TSM,13	6.2 7.2
Set Synchronization File Lock	M.FSLS	1,X'23'	H.FISE,24	6.4
Set Synchronous Resource Lock	M.SETSYNC M_SETSYNC	2,X'46' 2,X'46'	H.REMM,25 H.REMM,25	6.2 7.2
Set Synchronous Task Interrupt	M.SYNCH M_SYNCH	1,X'1B' 1,X'1B'	H.REXS,67 H.REXS,67	6.2 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	M.VADDR	2,X'59'	H.REXS,33	6.2
Range	M_VADDR	2,X'59'	H.REXS,33	7.2
Wait for Any	M.ANYW	1,X'7C'	H.REXS,37	6.2
No-wait Operation Complete, Message Interrupt, or Break Interrupt	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 Reserve FHD Port - Base Mode Reserve Dual-ported Disc/Set Single-channel ACM Mode	H.IOCS,24	N/A N/A M.RESP M_RESP	6.3 7.3 6.2 7.2
27	Release FHD Port Release FHD Port - Base Mode Release Dual-ported Disc/Set Dual-channel ACM Mode	H.IOCS,27	N/A N/A M.RELP M_RELP	6.3 7.3 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 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 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

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

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

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	0	0	0	0	0	0	0	0	0
5	1	0	0	0	0	1	1	1	1
6	2	0	0	1	1	0	0	1	1
7	3	0	1	0	1	0	1	0	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* =*xxxxxxxxxxxx*	Load CRAM with *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* <ret>	Read physical memory address location *xxxxxx* Increment and read memory address
MAV=*xxxxxx* <ret>	Read virtual memory address location *xxxxxx* Increment and read memory address
MD=*xxxxxxxx* =*xxxxxxxx* <ret>	Write memory data *xxxxxxxx* into last location addressed Increment and write memory data *xxxxxxxx* 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 <ret>	Instruction step 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.

Word	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

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)

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;

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)

Byte (Hex)	Symbol	Description
	DQE.IPUF	IPU flag byte; Field length = 1B;

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)

Byte (Hex)	Symbol	Description
	DQE.NWIO	Number of no-wait I/O requests; Field length = 1B.
	DQE.SOPO	Priority bias only swapping control flags; Field length = 1B;

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

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;
	DQE.TIFC	Timer function code; Field length = 1B;
	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.

DQE.SHF bits:

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).

DQE.TIFC values:

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
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

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.
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)
	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.

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
	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
	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
	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.
	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 2=0)	(Bit 2=1)	Bit 8	Bit 9		Bit 10	Bit 11	Bit 12
ALIM	Read:	**Bit 2**	**Bit 8**	**Bit 9**	**Read**	On Read:		
(Asynch-	receive							
ronous	data	0	1	0	=Blind mode reset	1= Inhibit		
Line	(bytes)	0	0	1	=Echo on read	conversion		
Interface	defined	1	N/A	N/A	=Receive data	of lower		
Module)	for	0	0	0	=Receive data	case		
Terminals	transfer					characters		
(TY)	count					to upper		
					Write	case		
	Write:	0	N/A	0	=Formatted write	0= Convert		
	formatted	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 2=0)	(Bit 2=1)	Bit 8	Bit 9		Bit 10	Bit 11	Bit 12
ALIM	Read:	**Bit 2**	**Bit 8**	**Bit 9**	**Read**	On Read:		
(Asynch-	receive							
ronous	data	0	1	0	=Blind mode reset	1= Inhibit		
Line	(bytes)	0	0	1	=Echo on read	conversion		
Interface	defined	1	N/A	N/A	=Receive data	of lower		
Module)	for	0	0	0	=Receive data	case		
Terminals	transfer					characters		
(TY)	count					to upper		
					Write	case		
	Write:	0	N/A	0	=Formatted write	0= Convert		
	formatted	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	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

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)
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
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	Contents
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	Contents
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	Contents
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	Contents
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.

Word	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.

Word	0–7	8–15	16–23	24–31
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)

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

Cataloged Priority of 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	Receive task	Priority in PTASK block	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)

Word	0 7 8 15 16 23 24 31
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)

Word	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

Word	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

Word	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

Word	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)			

Type 6 - Assign by Resource ID

<table>
<tr><th></th><th>0 — 7</th><th>8 — 15</th><th>16 — 23</th><th>24 — 31</th></tr>
<tr><td>Word 0</td><td>Zero</td><td colspan="3">Logical file code (RR.LFC)</td></tr>
<tr><td>1</td><td>Type (RR.TYPE). See Note 1.</td><td>Size (RR.SIZE)</td><td>Zero</td><td>Reserved</td></tr>
<tr><td>2</td><td colspan="4">Access (RR.ACCS). See Note 3.</td></tr>
<tr><td>3</td><td colspan="4">Options (RR.OPTS). See Note 4.</td></tr>
<tr><td>4-7</td><td colspan="4">Volume name (16 characters; left-justified, blank-filled) (RR.NAME1)</td></tr>
<tr><td>8</td><td colspan="4">Binary creation date (RR.DATE)</td></tr>
<tr><td>9</td><td colspan="4">Binary creation time (RR.TIME)</td></tr>
<tr><td>10</td><td colspan="4">Resource descriptor block address (RR.DOFF)</td></tr>
<tr><td>11</td><td colspan="2">Reserved</td><td colspan="2">Resource type (RR.RTYPE)</td></tr>
</table>

Type 7 - Reserved for Future Use

Type 8 - Reserved for Future Use

Type 9 - Mount by Device Mnemonic

<table>
<tr><th></th><th>0 — 7</th><th>8 — 15</th><th>16 — 23</th><th>24 — 31</th></tr>
<tr><td>Word 0</td><td>Zero</td><td colspan="3">System ID (RR.SYSID). See Note 11.</td></tr>
<tr><td>1</td><td>Type (RR.TYPE). See Note 1.</td><td>Size (RR.SIZE)</td><td colspan="2">Zero</td></tr>
<tr><td>2</td><td colspan="4">Access (RR.ACCS). See Note 3.</td></tr>
<tr><td>3</td><td colspan="4">Options (RR.OPTS). See Note 4.</td></tr>
<tr><td>4-7</td><td colspan="4">Volume name (16 characters; left-justified, blank-filled) (RR.NAME1)</td></tr>
<tr><td>8</td><td>Device type (RR.DT9). See Note 8.</td><td>Reserved</td><td>Channel number (RR.CHN9). See Note 9.</td><td>Subchannel number (RR.SCHN9)</td></tr>
<tr><td>9</td><td colspan="4">Zero</td></tr>
</table>

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) 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

- B -

- D -

- E -

- F -

- G -

- H -

- P -

- R -

- U -

- W -

MPX-32™ Unsupported Software
Reference Manual
No. 323-001554-600

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

PLEASE TAPE DO NOT STAPLE

CUT ALONG LINE