MPX-32 ™

Resident Modules and Handlers

Revision 3.5

Technical Manual Volume II

April 1990

322-001552-500

ENCORE

Limited Rights

This manual is supplied without representation or warranty of any kind. Encore Computer Corporation therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Proprietary Information

The information contained herein is proprietary to Encore Computer Corporation and/or its vendors, and its use, disclosure, or duplication is subject to the restrictions stated in the standard Encore Computer Corporation License terms and conditions or the appropriate third-party sublicense agreement.

Restricted Rights

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at 252.227.7013.

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

MPX-32 ™

Resident Modules

Revision 3.5

Technical Manual Volume II(A)

April 1990

H.ADA
H.ALOC
H.BKDM
H.EXEC
H.EXSUB
H.FISE
H.IOCS
H.IP?? and H.SVC?
H.MDT
H.MEMM
H.MEMM2
H.MONS
H.MVMT
H.PTRAC
H.REMM
H.REXS
H.SURE
H.TAMM
H.TSM
H.VOMM
System Macros Cross-Reference

322-001552-500

History

The MPX-32 Release 3.2 Technical Manual, Publication Order Number **322-001550-000**, was printed September, 1983.

Publication Order Number **322-001550-100**, (Revision 1, Release 3.2B) was printed March, 1985.

Publication Order Number **322-001550-101**, (Change 1 to Revision 1, Release 3.2C) was printed December, 1985.

The MPX-32 Release 3.3 Technical Manual Volume I, Publication Order Number **322-001552-200**, was printed December, 1986.

Publication Order Number **322-001552-300**, (Revision 3, Release 3.4) was printed January, 1988.

Publication Order Number **322-001552-400**, (Revision 3.4U03) was printed October, 1989.

Publication Order Number **322-001552-500**, (Revision 3.5) was printed April, 1990

This manual contains the following pages:

Title page
Copyright page
Title page (A)
v through xi/xii

Overview

1-1 through 1-2

H.ADA

Title page
iii/iv
1-1/1-2
2-1 through 2-3/2-4
3-1/3-2

H.ALOC

Title page
iii/iv
1-1/1-2
2-1 through 2-9/2-10
3-1/3-2

H.BKDM

Title page
iii/iv
1-1/1-2
2-1/2-2
3-1 through 3-5/3-6

H.EXEC

Title page
iii through vi
1-1 through 1-3/1-4
2-1 through 2-25/2-26
3-1 through 3-31/3-32

H.EXSUB

Title page
iii and iv
1-1/1-2
2-1 through 2-26

H.FISE

Title page
iii and iv
1-1 and 1-2
2-1 through 2-11/2-12

H.IOCS

Title page
iii through v/vi
1-1 through 1-3/1-4
2-1 through 2-15/2-16
3-1 through 3-27/3-28

H.IP?? and H.SVC?

Title page
iii/iv
1-1/1-2
2-1 through 2-3/2-4
3-1 through 3-16

H.MDT

Title page
iii/iv
1-1/1-2
2-1 through 2-5/2-6
3-1 through 3-12

H.MEMM

Title page
iii and iv
1-1 and 1-2
2-1 through 2-4
3-1 through 3-14

H.MEMM2

Title page
iii/iv
1-1/1-2
2-1 through 2-6

H.MONS

Title page
iii through v/vi
1-1 through 1-3/1-4
2-1 through 2-26

H.MVMT

Title page
iii/iv
1-1/1-2
2-1 and 2-2

H.PTRAC

Title page
iii and iv
1-1 and 1-2

H.REMM

Title page
iii and iv
1-1 through 1-3/1-4
2-1 through 2-6
3-1 through 3-23/3-24

H.REXS

Title page
iii through vi
1-1 through 1-4
2-1 through 2-24
3-1 through 3-8

H.SURE

Title page
iii/iv
1-1/1-2
2-1/2-2

H.TAMM

Title page
iii/iv
1-1/1-2
2-1 through 2-6
3-1 through 3-8

H.TSM

Title page
iii/iv
1-1/1-2
2-1 through 2-8
3-1 through 3-3/3-4

H.VOMM

Title page
iii and iv
1-1 through 1-3/1-4
2-1 through 2-8
3-1 through 3-29/3-30

System Macros Cross-Reference

A-1 through A-11

Title page (B)

H.DCSCI

Title page
iii through v/vi
1-1/1-2
2-1 through 2-12
3-11/3-12

H.DCXIO

Title page
iii through vii/viii
1-1 through 1-3/1-4
2-1 through 2-12
3-1 through 3-15/3-16

H.DPXIO

Title page
iii through vii/viii
1-1 through 1-3/1-4
2-1 through 2-10
3-1 through 3-23/3-24

H.F8XIO

Title page
iii/iv
1-1/1-2
2-1 through 2-7/2-8

H.GPMCS

Title page
iii/iv
1-1 and 1-4
2-1 through 2-13/2-14

H.HSDG

Title page
iii/iv
1-1/1-2
2-1 through 2-10
3-1 through 3-6

H.IBLG

Title page
iii/iv
1-1/1-2
2-1 through 2-8
3-1 through 3-7/3-8

H.MDXIO

Title
iii and iv
1-1 through 1-2
2-1 through 2-7/2-8
3-1/3-2

H.MTSCI

Title page
iii throuh v/vi
1-1/1-2
2-1 through 2-14
3-1 through 3-12

H.XIOS

Title page
iii through vi
1-1/1-2
2-1 through 2-11/2-12
3-1 through 3-7/3-8
4-1 through 4-3/4-4

Documentation Conventions

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

Messages and Examples

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

```
VOLUME MOUNT SUCCESSFUL
```

or

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

Lowercase Italic Letters

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

$NOTE *message*

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

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

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

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

means a break occurred on the specified task.

Uppercase Letters

In directive syntax, uppercase letters specify the input required to execute that directive. Uppercase bold letters indicate the minimum that must be entered. For example,

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

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

```
$AS OUT TO OUTFILE
```

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

```
TERMDEF HAS NOT BEEN INSTALLED
```

Brackets [ ]

An element inside brackets is optional. For example,

$CALL *pathname* [*arg*]

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

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

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

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

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

BACKSPACE **F**ILE [[**FILES=**] *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 Technical Manual Volume II Overview

1.1 Using the Manual

The information in this manual is divided into two parts:

Module Descriptions

Handler Descriptions

Each module has a self-contained description that is prefaced by a tab. The following list includes the modules which are described in this manual:

H.ADA
H.ALOC
H.BKDM
H.EXEC
H.EXSUB
H.FISE
H.IOCS
H.IP?? and H.SVC?
H.MDT
H.MEMM
H.MEMM2
H.MONS
H.MVMT
H.PTRAC
H.REMM
H.REXS
H.SURE
H.TAMM
H.TSM
H.VOMM

Each module description has the following format as applicable:

Overview
Entry Points
Subroutines

Each handler has a self-contained description that is prefaced by a tab. The following list includes the handlers which are described in this manual.

H.DCSCI
H.DCXIO
H.DPXIO
H.F8XIO
H.GPMCS
H.HSDG
H.IBLG
H.MDXIO
H.MTSCI
H.XIOS

Each handler description has the following format, as applicable:

Overview
Usage
Entry Points
Subroutines

To customize this manual to a particular system, remove the descriptions of modules or handlers that are not installed on the system.

Ada Programming Language
Support Module (H.ADA)
MPX-32 Technical Manual
Volume II

Contents

1 H.ADA Overview

1.1 General Information

The Ada Programming Language Support Module (H.ADA) provides support for the Ada environment on MPX-32. This support includes arithmetic exception processing and abort processing. All system services are channeled through this module to provide the Ada/MPX-32 interface.

1.2 Ada Callable SVC Summary

SVC Number	Description
2,X'A4'	allocate signal stack space
2,X'A5'	exit from signal/exception state
2,X'A6'	call any SVC service
2,X'A7'	return to current working volume and directory

1.3 Subroutine Summary

Subroutine	Description
S.ADA1	dispatch task
S.ADA2	arithmetic exception handling
S.ADA3	dispatch control to abort receiver
S.ADA4	end-action processing

2 H.ADA - Ada Callable SVCs

2.1 SVC 2,X´A4´ - Allocate Signal Stack Space

This routine allocates space to an Ada task for arithmetic exception processing.

Entry Conditions

Calling Sequence

SVC 2,X'A4'

Registers

R3 logical address of stack
R4 number of bytes to use for stack

Exit Conditions

Return Sequence

M.RTRN

Registers

All registers unchanged.

Status

If CC1 is set, the address is invalid.

2.2 SVC 2,X'A5' - Exit From Signal/Exception State

This routine restores the task registers and PSD for proper return to the Ada task.

Entry Conditions

Calling Sequence

SVC 2,X'A5'

Registers

None

Exit Conditions

Return Sequence

M.RTRN

2.3 SVC 2,X'A6' - Call Any SVC Service

This routine intercepts all MPX-32 SVCs called by Ada tasks. This routine checks for recursive SVC calls and verifies the destination address. The SVC is then executed from this routine with the return address inside the entry point. After the SVC is executed, the routine builds the condition codes and registers into the current TSA stack frame and returns via M.RTRN.

Entry Conditions

Calling Sequence

SVC 2,X'A6'

Registers

R1	address of an 8-word register block
R2	address of a 9-word out-register block with condition codes left justified in the ninth word
R3	right 16 bits of the SVC instruction type and number

Exit Conditions

Return Sequence

M.RTRN

Status

CC1 if zero, no registers are affected

CC1 if one, R3 contains one of the following error codes:

Code	Meaning
1	invalid in-register buffer address
2	invalid out-register buffer address
3	recursive call to SVC

2.4 SVC 2,X'A7' - Return to Current Working Volume

This routine restores the Ada task environment before returning to the task.

Entry Conditions

Calling Sequence

```
SVC     2,X'A7';
```

Registers

R1	address of buffer to receive volume name
R2	address of buffer to receive directory name

Exit Conditions

Return Sequence

M.RTRN

Status

Normal return	if the volume and directory names are stored in the buffers
Error return	if either buffer contains an invalid address, then the volume and directory names are not stored in the buffer. No error is reported.

3 H.ADA Subroutines

3.1 Subroutine S.ADA1 - Reserved

3.2 Subroutine S.ADA2 - Arithmetic Exception Handling

This subroutine uses the argument list in H.IP0F.

Entry Conditions

Calling Sequence

BL	S.ADA2

Registers

R1	address of H.IP0F's argument list
R3	TSA address
R4	Ada exception handler routines
R6	PSD is being built

Exit Conditions

Return Sequence

No return.

LPSD	address of Ada task exception handler

3.3 Subroutine S.ADA3 - Reserved

3.4 Subroutine S.ADA4 - Reserved

Resource Allocator (H.ALOC)

MPX-32 Technical Manual

Volume II

Contents

1 H.ALOC Overview

1.1 General Information

The Resource Allocator Module (H.ALOC) performs compatible mode services associated with allocating and deallocating system resources.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.ALOC,1	N/A	construct TSA and DQE
H.ALOC,2	N/A	task activation processing
H.ALOC,3	N/A	task exit processing
H.ALOC,5	N/A	deallocate memory
H.ALOC,6	N/A	allocate file/device
H.ALOC,7	N/A	deallocate file/device
H.ALOC,8	69	get dynamic extended data space
H.ALOC,9	6A	free dynamic extended indexed data space
H.ALOC,10	67	get dynamic task execution space
H.ALOC,11	68	free dynamic task execution space
H.ALOC,12	71	share memory with another task
H.ALOC,13	72	get shared memory (INCLUDE)
H.ALOC,14	79	free shared memory (EXCLUDE)
H.ALOC,15	N/A	reserved
H.ALOC,16	N/A	reserved
H.ALOC,17	N/A	allocate disk file by space definition
H.ALOC,18	N/A	reserved
H.ALOC,19	1F	unlock and dequeue shared memory
H.ALOC,20	N/A	deallocate memory due to swapping
H.ALOC,21	N/A	locate allocated FPT/FAT
H.ALOC,99	N/A	SYSGEN initialization

N/A implies reserved for internal use by MPX-32

1.3 Subroutine Summary

Subroutine	Description
S.ALOC91	locate shared memory table entry

2 H.ALOC Entry Points

2.1 Entry Point 1 - Construct TSA and DQE

See H.TAMM,2 for a detailed description of this entry point.

2.2 Entry Point 2 - Task Activation Processing

See H.TAMM,3 for a detailed description of this entry point.

2.3 Entry Point 3 - Task Exit Processing

See H.TAMM,4 for a detailed description of this entry point.

2.4 Entry Point 4 - Allocate Memory

See H.MEMM,1 for a detailed description of this entry point.

2.5 Entry Point 5 - Deallocate Memory

See H.MEMM,2 for a detailed description of this entry point.

2.6 Entry Point 6 - Allocate File/Device

This entry point converts a three word RRS entry into a multiword RRS format. Transfer is then passed to H.REMM,6.

Entry Conditions

Calling Sequence

M.CALL H.ALOC,6

Registers

R1 address of 3-word RRS entry

Exit Conditions

Return Sequence

M.RTRN R1

(or)

M.RTRN R1,R6,R7

Registers

R1 zero if allocation was unsuccessful. Otherwise, R1 is unchanged.

CC1 set if allocation denied:

R6 contains the scan mask
R7 contains the device requirements mask

CC2 set if allocation error:

R6 contains an error code
R7 contains zero

Error Condition

Registers

R6 contains the following:

Value	Definition
1	permanent file nonexistent
2	illegal file password specified
3	no FAT/FPT space available
4	no blocking buffer space available
5	shared memory table entry not found
6	invalid shared memory table password specified
7	dynamic common specified in ASSIGN1
8	unrecoverable I/O error to directory
9	SGO assignment specified by terminal task
10	no UT file code exists for terminal task
11	invalid RRS entry
12	LFC in ASSIGN4 nonexistent
13	assigned device not on system
14	device in use by requesting task
15	SGO or SYC assignment by real-time task
16	common memory conflicts with allocated task
17	duplicate LFC allocation attempted

External Reference

System Macro

M.CALL
M.RTRN

System Services

H.REMM,6
H.REXS,20
H.REXS,76

System Subroutine

S.REXS8

2.7 Entry Point 7 - Deallocate File/Device

This entry point creates the calling sequence needed by H.REMM,7. Transfer is then passed to H.REMM,7.

Entry Conditions

Calling Sequence

M.CALL H.ALOC,7

Registers

R5 1- to 3-character right-justified ASCII logical file code

Exit Conditions

Return Sequence

M.RTRN

Registers

None

Error Condition

CC1 set if unrecoverable I/O error to directory

External Reference

System Macro

M.CALL
M.RTRN

System Service

H.REMM,7

System Subroutine

S.REXS8

2.8 Entry Point 8 - Get Dynamic Extended Data Space

See M.GD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.9 Entry Point 9 - Free Dynamic Extended Indexed Data Space

See M.FD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.10 Entry Point 10 - Get Dynamic Task Execution Space

See M.GE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.11 Entry Point 11 - Free Dynamic Task Execution Space

See M.FE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.12 Entry Point 12 - Share Memory With Another Task

See M.SHARE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Service

H.REMM,12

System Subroutines

S.REXS8
S.REXS9
S.ALOC91

2.13 Entry Point 13 - Get Shared Memory (INCLUDE)

See M.INCL in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Service

H.REMM,12

System Subroutines

S.REXS8
S.REXS9

2.14 Entry Point 14 - Free Shared Memory (EXCLUDE)

See M.EXCL in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Service

H.REMM,14

System Subroutine

S.REXS8

2.15 Entry Point 15 - Reserved

2.16 Entry Point 16 - Reserved

2.17 Entry Point 17 - Allocate Disk File By Space Definition

This entry point creates a multi-word RRS entry from the parameters provided by the caller. Transfer is then passed to H.REXS,21.

Entry Conditions

Calling Sequence

```
M.CALL   H.ALOC,17
```

Registers

R4	LFC (bit 0 set for system FAT/FPT)
R5	UDT index (bit 0 set for blocking buffer)
R6	sector address
R7	number of sectors

Exit Conditions

Return Sequence

M.RTRN R1,R2,R3,R5

Registers

R1	UDT address
R2	FPT address
R3	FAT address
R5	blocking buffer address if required
CC1	set if no FAT/FPT space
CC2	set if no blocking buffer space

External Reference

System Macro

M.CALL
M.RTRN

System Service

H.REXS,21

System Subroutine

S.REXS8

2.18 Entry Point 18 - Reserved

2.19 Entry Point 19 - Unlock and Dequeue Shared Memory

See M.SMULK in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN
M.SHUT
M.OPEN

System Service

H.REMM,24

System Subroutine

S.ALOC91

2.20 Entry Point 20 - Deallocate Memory Due to Swapping

See H.MEMM,11 for a detailed description of this entry point.

2.21 Entry Point 21 - Locate Allocated FPT/FAT

This subroutine locates the FPT/FAT pair associated with a given logical file code (LFC).

Entry Conditions

Calling Sequence

M.CALL H.ALOC,21

Registers

R5	left-justified, blank-filled (bytes 1-3), 3-ASCII character LFC (bit 0 set indicates system FPT/FAT)

Exit Conditions

Return Sequence

M.RTRN and CC1 set if LFC not found

Registers

CC1	LFC not found
R2	FPT address
R3	FAT address
R5	LFC with byte 0 clear

External Reference

System Macro

M.RTRN

System Subroutines

S.REMM12

S.REXS8

2.22 Entry Point 99 - SYSGEN Initialization

This entry point is for internal use only and is called during SYSGEN. H.ALOC sets up its entry point table, then returns to SYSGEN.

3 H.ALOC Subroutines

3.1 Subroutine S.ALOC91 - Locate Shared Memory Table Entry

This subroutine is used to find the first shared memory table (SMT) entry which contains the partition name and owner name (or task number) specified by the caller.

Entry Conditions

Calling Sequence

```
BL      S.ALOC91
```

Registers

R4,R5	owner name
(or)	
R4	zero
R5	task number
R6,R7	partition name

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1	address of matching SMT or zero if not matched
R3	destroyed
R4-R7	unchanged

Blocked Data Management Module (H.BKDM)

MPX-32 Technical Manual

Volume II

Contents

1 H.BKDM Overview

1.1 General Information

The Blocked Data Management Module (H.BKDM) performs all data management operations pertaining to blocked I/O requests.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.BKOP	N/A	predevice access processing
H.BKPX	N/A	postdevice access processing

N/A implies called only by IOCS

1.3 Subroutine Summary

Subroutine	Description
S.BKDM1	initialize blocking buffer
S.BKDM2	read logical blocked record
S.BKDM3	verify blocking buffer
S.BKDM4	perform blocked data positioning
S.BKDM5	save FCB parameters in SPAD
S.BKDM6	write logical blocked record
S.BKDM7	advance logical blocked record
S.BKDM8	restore FCB parameters from SPAD

2 H.BKDM Entry Points

2.1 Entry Point H.BKOP - Predevice Access Processing

This entry point performs predevice access processing on behalf of blocked data requests.

Entry Conditions

Calling Sequence

```
BU      H.BKOP
```

Registers

R1 FCB address

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.2 Entry Point H.BKPX - Postdevice Access Processing

This entry point performs postdevice access processing related to blocked I/O requests.

Entry Conditions

Calling Sequence

```
BL      H.BKPX
```

Registers

R1 FCB address

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1 FCB address

3 H.BKDM Subroutines

3.1 Subroutine S.BKDM1 - Initialize Blocking Buffer

This routine is used to initialize a blocking buffer.

Entry Conditions

Calling Sequence

BL	S.BKDM1

Registers

R1	FCB address

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2	FAT address
R3,R4	destroyed

3.2 Subroutine S.BKDM2 - Read Logical Blocked Record

This routine performs a read of a logical blocked record. For example, it transfers a logical blocked record from a blocking buffer to a user's data area.

Entry Conditions

Calling Sequence

BL	S.BKDM2

Registers

R1	FCB address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1 FCB address
R2-R7 destroyed

3.3 Subroutine S.BKDM3 - Verify Blocking Buffer

This routine is used to verify that the blocking buffer contains valid control information.

Entry Conditions

Calling Sequence

BL S.BKDM3

Registers

R3 address of the buffer

Exit Conditions

Return Sequence

TRSW R0

Registers

R2,R4-R6 destroyed
R3 address of the buffer

3.4 Subroutine S.BKDM4 - Perform Blocked Data Positioning

This routine performs blocked data positioning for the I/O.

Entry Conditions

Calling Sequence

BL S.BKDM4

Registers

R1 FCB address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1	FCB address
R2-R7	destroyed

3.5 Subroutine S.BKDM5 - Save FCB Parameters in SPAD

This routine saves original FCB parameters and inserts new FCB parameters prior to physical I/O operations performed on behalf of a user who requested blocked I/O operations.

Entry Conditions

Calling Sequence

BL S.BKDM5

Registers

R1	FCB address
R3	blocking buffer address
R7	special status byte

Spad Cells Used:

1, 2, 3

Exit Conditions

Return Sequence

TRSW R0

Registers

R2	address of saved parameters
R4-R6	destroyed

3.6 Subroutine S.BKDM6 - Write Logical Blocked Record

This routine performs a write of a logical blocked record. For example, it transfers a logical blocked record from the user's data area to a blocking buffer.

Entry Conditions

Calling Sequence

BL	S.BKDM6

Registers

R1	FCB address

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1	FCB address
R2-R7	destroyed

3.7 Subroutine S.BKDM7 - Advance Logical Blocked Record

This routine performs an advance logical blocked record; no transfer is required, only next read/write address is updated.

Entry Conditions

Calling Sequence

BL	S.BKDM7

Registers

R1	FCB address
R2	current logical record
R3	blocking buffer address

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1	FCB address
R6	destroyed

3.8 Subroutine S.BKDM8 - Restore FCB Parameters

This routine restores original FCB parameters from the scratchpad subsequent to physical operations performed on behalf of a user who requested blocked I/O operations.

Entry Conditions

Calling Sequence

```
BL      S.BKDM8
```

Registers

R1	FCB address

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R2	address of saved parameters
R4,R6	destroyed

Executive Module (H.EXEC)

MPX-32 Technical Manual

Volume II

Contents

Page

3 H.EXEC Subroutines

Page

List of Figures

1 H.EXEC Overview

1.1 General Information

The Executive Module (H.EXEC) performs as a CPU scheduler, by allocating the CPU and IPU to tasks. The information listed in H.EXEC for entry points and subroutines applies to EXEC, as well as EXEC2 and EXEC3.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.EXEC,1	N/A	interactive input starting
H.EXEC,2	N/A	terminal output starting
H.EXEC,3	N/A	wait I/O starting
H.EXEC,4	N/A	no-wait I/O starting
H.EXEC,5	N/A	wait for any no-wait operation complete
H.EXEC,6	N/A	wait for memory pool
H.EXEC,7	N/A	memory request processing complete
H.EXEC,8	N/A	wait for memory scheduler event
H.EXEC,9	N/A	report memory scheduler event
H.EXEC,10	N/A	report memory pool available
H.EXEC,11	N/A	completion of unswappable I/O request
H.EXEC,12	N/A	no-wait I/O postprocessing complete
H.EXEC,13	N/A	wait for peripheral resource
H.EXEC,14	N/A	wait for disk file space
H.EXEC,15	N/A	report peripheral resource available
H.EXEC,16	N/A	report disk file space available
H.EXEC,17	N/A	reserved
H.EXEC,18	N/A	reserved
H.EXEC,19	N/A	resume execution of specified task
H.EXEC,20	N/A	suspend execution of current task
H.EXEC,21	N/A	suspend execution of specified task
H.EXEC,22	N/A	go to specified task context (AIDDB)
H.EXEC,23	N/A	run user break receiver (AIDDB)
H.EXEC,24	N/A	reserved (AIDDB)
H.EXEC,25	N/A	wait for any no-wait operation complete, message interrupt or break interrupt
H.EXEC,26	N/A	continue specified task
H.EXEC,27	N/A	general enqueue
H.EXEC,28	N/A	report run request postprocessing complete
H.EXEC,29	N/A	report wait mode run request starting
H.EXEC,30	N/A	enable AIDDB mode break
H.EXEC,31	N/A	hold current task
H.EXEC,32	N/A	hold specified task
H.EXEC,33	N/A	disable AIDDB mode break
H.EXEC,34	N/A	report no-wait message postprocessing complete
H.EXEC,35	N/A	report wait mode message starting
H.EXEC,36	N/A	general dequeue

Entry Point	SVC number	Description
H.EXEC,37	N/A	wait for memory available
H.EXEC,38	N/A	inhibit asynchronous abort/delete
H.EXEC,39	N/A	allow asynchronous abort/delete
H.EXEC,40	1D***	end action wait
H.EXEC,41	70***	get user context
H.EXEC,42	71***	put user context
H.EXEC,43	N/A	reserved for Symbolic Debugger/X32

*** This service is SVC 2,X'*nn*' callable.

N/A implies reserved for internal use by MPX-32.

1.3 Subroutine Summary

Subroutine	Description
S.EXEC1	interactive input complete
S.EXEC2	terminal output complete
S.EXEC3	wait I/O complete
S.EXEC4	no-wait I/O complete
S.EXEC4A	no-wait I/O complete (no postprocessing)
S.EXEC5	exit from interrupt
S.EXEC5A	exit from trap handler with abort
S.EXEC6	no-wait I/O postprocessing complete
S.EXEC7	report memory pool available
S.EXEC8	link entry to queue by priority
S.EXEC9	unlink entry from queue
S.EXEC10	link entry to bottom of queue
S.EXEC11	link entry to top of queue
S.EXEC12	report memory scheduler event
S.EXEC13	break specified task
S.EXEC14	resume specified task
S.EXEC20	CPU scheduler
S.EXEC21	process task interrupt
S.EXEC23	terminate messages in receiver queue
S.EXEC24	reserved
S.EXEC25	terminate next run request in receiver queue
S.EXEC27	transfer control to abort receiver
S.EXEC30	reserved
S.EXEC31	report no-wait run request postprocessing complete
S.EXEC34	reserved
S.EXEC35	report no-wait mode message postprocessing complete
S.EXEC40	reserved
S.EXEC41	exit run receiver
S.EXEC42	exit message receiver
S.EXEC44	change priority level of current task

Subroutine	Description
S.EXEC46	reserved
S.EXEC47	reserved
S.EXEC55	unlink task from designated list and link to ready list
S.EXEC56	resume memory scheduler
S.EXEC57	link task to ready list by priority
S.EXEC59	reserved
S.EXEC61	transfer parameters from MRRQ to receiver buffer
S.EXEC62	validate RXB
S.EXEC68	construct and vector context to end action PSD
S.EXEC69	common no-wait postprocessing merge point
S.EXEC72	report wait I/O starting
S.EXEC75	situational priority increment
S.EXEC77	reserved
S.EXEC79	push current context onto stack for deferred EA pull
S.EXEC80	start IPU and verify
S.EXEC81	enter base mode debugger entry point four
S.EXEC82	push calling task context onto the stack

2 H.EXEC Entry Points

2.1 Entry Point 1 - Interactive Input Starting

This entry point is called to report the beginning of processing for an interactive input request made by the currently executing task. The task is removed from the associated ready-to-run list, and placed in the wait for interactive input list. A return to the calling routine is made when the input request completes.

Entry Conditions

Calling Sequence

```
M.SHUT
UEI
M.CALL      H.EXEC,1
```

Registers

R0,Bit 0 one indicates task is swappable during input processing

Exit Conditions

Return Sequence

CPU scheduler (when I/O complete, with M.OPEN status)

Registers

None

2.2 Entry Point 2 - Terminal Output Starting

This entry point is called to report the beginning of processing for a terminal output request made by the currently executing task. The task is removed from the associated ready-to-run list, and placed in the wait for terminal output list. A return to the calling routine is made when the output request completes.

Entry Conditions

Calling Sequence

```
M.SHUT
UEI
M.CALL      H.EXEC,2
```

Registers

R0,Bit 0 one indicates task is swappable during output processing

Exit Conditions

Return Sequence

CPU scheduler (when I/O complete, with M.OPEN status)

Registers

None

2.3 Entry Point 3 - Wait I/O Starting

This entry point is called to report the beginning of processing for a wait I/O request made by the currently executing task. The task is removed from the associated ready-to-run list, and placed in the wait for I/O list. A return to the calling routine is made when the I/O request completes.

Entry Conditions

Calling Sequence

```
M.SHUT
UEI
M.CALL      H.EXEC,3
```

Registers

R0,Bit 0 one indicates task is swappable during I/O processing

Exit Conditions

Return Sequence

CPU scheduler (when I/O complete, with M.OPEN status)

Registers

None

2.4 Entry Point 4 - No-Wait I/O Starting

This entry point is called to report the beginning of processing for a no-wait I/O request made by the currently executing task. A return to the calling routine is made after recording the no-wait I/O start event.

Entry Conditions

Calling Sequence

```
M.SHUT
UEI
M.CALL      H.EXEC,4
```

Registers

R0,Bit 0 one indicates task is swappable during I/O processing

Exit Conditions

Return Sequence

```
M.OPEN
M.RTRN
```

Registers

None

2.5 Entry Point 5 - Wait for Any No-Wait Operation Complete

This entry point is functionally identical to H.EXEC,25 except that it does not check for outstanding message or break interrupt requests before placing a task on the ANYW queue. All queued end action requests are processed before a return is made to the calling routine. This entry point is used by IOCS when waiting for a particular no-wait I/O request to complete.

Entry Conditions

Calling Sequence

```
M.CALL      H.EXEC,5
```

Registers

R6 zero if indefinite wait; otherwise, this register contains the negative number of timer units for timed wait

Exit Conditions

Return Sequence

```
M.RTRN
```

Registers

None

2.6 Entry Point 6 - Wait for Memory Pool

This entry point is called when the required memory pool space is not available. The currently executing task is removed from the associated ready-to-run list, and placed in the wait for memory pool list. A return to the calling routine is made when any memory pool space is deallocated. The calling routine can then make another attempt to allocate the required memory pool space.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,6

Registers

None

Exit Conditions

Return Sequence

CPU scheduler

Registers

None

2.7 Entry Point 7 - Memory Request Processing Complete

This entry point is called by the memory scheduler when processing for a memory request is complete. The DQE associated with the memory request will have been unlinked from the memory request queue by the memory scheduler. The completed memory request is processed by H.EXEC,7 according to request type. (The DQE contains the request type information.) The task is then linked into the appropriate ready-to-run list. A return to the memory scheduler is made by issuing a M.RTRN.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,7

Registers

R2 DQE address

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.8 Entry Point 8 - Wait for Memory Scheduler Event

This entry point is called by the memory scheduler when either no additional processing of outstanding memory requests is possible, or the memory request list is empty. C.RRUN is examined. If C.RRUN is not equal to zero, and the memory request queue is not empty, the memory scheduler will be reexecuted. Otherwise, the memory scheduler will be removed from the ready-to-run list and placed in the wait for memory event list. A return to the memory scheduler occurs when:

- a new memory request is queued, or
- the memory request queue is not empty and the status of allocated memory changes such that it either is deallocated or becomes more eligible for swapping.

Entry Conditions

Calling Sequence

```
M.CALL      H.EXEC,8
```

Registers

None

Exit Conditions

Return Sequence

CPU scheduler

Registers

None

2.9 Entry Point 9 - Report Memory Scheduler Event

This entry point is called when the status of allocated memory changes (it is either deallocated, or becomes more eligible for swapping). This routine insures the appropriate execution of the memory scheduler task. If the memory-request list is empty, no additional processing is required and a return is made to the user. If the memory-request list is not empty, C.RRUN is incremented, and the memory scheduler state is checked. If the memory scheduler is in the wait for memory event list, it is removed from that list, and placed in the ready-to-run list at the priority of the highest priority entry in the memory-request list. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,9

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.10 Entry Point 10 - Report Memory Pool Available

This entry point is called when memory pool space is deallocated. This routine resumes the execution of all tasks in the wait for memory pool list. If the wait for memory pool list is empty, no additional processing is required and a return is made to the calling routine. Otherwise, each entry in the list is removed and placed in its associated ready-to-run list. It is expected that when these tasks resume execution, they will reissue the request for the required memory pool space. When all entries have been flushed from the wait for memory pool list, a return is made to the calling routine.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,10

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.11 Entry Point 11 - Completion of Unswappable I/O Request

This entry point is called by the IOCS post-transfer processing logic, executing on behalf of the current task. The count of unswappable I/O transfers in the DQE is decremented. If no other swap inhibit reasons exist, a call is made to H.EXEC,9 to report the memory scheduler event. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

```
M.CALL      H.EXEC,11
```

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.12 Entry Point 12 - No-Wait I/O Postprocessing Complete

This entry point is called by the IOCS no-wait I/O postprocessing logic to exit from the task interrupt state. The entry point clears the task interrupt processing lock, and returns to the point of task interrupt. It discards the most recent level of pushdown in the TSA stack, then issues an M.RTRN to return to the point of task interrupt.

Entry Conditions

Calling Sequence

```
M.CALL      H.EXEC,12
```

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.13 Entry Point 13 - Wait for Peripheral Resource

This entry point is called when the required peripheral resource is not available. The currently executing task is removed from the associated ready-to-run list and placed in the wait for peripheral resource list. A return to the calling routine is made when the specified peripheral is deallocated by its current user. The calling routine may then make another attempt to allocate the device.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,13

Registers

R6 peripheral requirements specification:

Bits	Definition
0-7	reserved
8-15	device type code
16-23	channel address
24-31	subchannel address

R7 requirements mask:

Value	Definition
X'00FF0000'	any device of this device type code
X'00FFFF00'	any device of the specified type code, on the specified channel
X'00FFFFFF'	the specific device described by this type code, channel address, and subchannel address

Exit Conditions

Return Sequence

CPU scheduler

Registers

None

2.14 Entry Point 14 - Wait for Disk File Space

This entry point is called when the required disc file space is not available. The currently executing task is removed from the associated ready-to-run list, and placed in the wait for disk list. A return to the calling routine is made when any disk file space is deallocated. The calling routine may then make another attempt to allocate the required disk file space.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,14

Registers

R6 disk device requirements specification:

Bits	Definition
0-7	reserved
8-15	device type code
16-23	channel address
24-31	subchannel address

R7 disk device requirements mask:

Value	Definition
X'00000000'	any disk
X'00FF0000'	any disk of the specified type code
X'00FFFF00'	any disk of the specified type code on the specified channel
X'00FFFFFF'	the specific disk device described by this type code, channel address, and sub-channel address

Exit Conditions

Return Sequence

CPU scheduler

Registers

None

2.15 Entry Point 15 - Report Peripheral Resource Available

This entry point is called when a peripheral device is deallocated. This routine resumes the execution of the tasks in the wait for peripheral resource list, which have specified requirements that will be satisfied by the deallocated device. If no such tasks exist, no additional processing is required and a return is made to the calling routine. Otherwise, each such entry in the list is removed and placed in its associated ready-to-run list. When these tasks resume execution, they are expected to reissue the request for the required device. When all appropriate entries have been flushed from the wait for peripheral resource list, a return is made to the calling routine.

Entry Conditions

Calling Sequence

```
M.CALL    H.EXEC,15
```

Registers

R6 peripheral resource:

Bits	Definition
0-7	reserved
8-15	device type code
16-23	channel address
24-31	subchannel address

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.16 Entry Point 16 - Report Disk File Space Available

This entry point is called when disk space is deallocated. This routine resumes the execution of the tasks in the wait for disc list which have specified requirements that may be satisfied by the deallocated disk file space. If no such tasks exist, no additional processing is required and a return is made to the calling routine. Otherwise, each such entry in the list is removed and placed in its associated ready-to-run list. When these tasks resume execution, they are expected to reissue the request for the required space. When all appropriate entries have been flushed from the list, a return is made to the calling routine.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,16

Registers

R6 disk device resource:

Bits	Definition
0-7	reserved
8-15	device type code
16-23	channel address
24-31	subchannel address

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.17 Entry Point 17 - Reserved

2.18 Entry Point 18 - Reserved

2.19 Entry Point 19 - Resume Execution of Specified Task

This entry point is called to resume execution of the specified task. This routine calls S.EXEC14 to accomplish the resume function. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,19

Registers

R2 DQE address of task to be resumed

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.20 Entry Point 20 - Suspend Execution of Current Task

This entry point is called to suspend execution of the current task, either for an indefinite period, or for the specified number of time units. The specified time (if any) is stored as a one-shot timer in the DQE along with a resume-program timer function code. S.EXEC15 is then called to suspend execution of the current task. A return is not made until the timer expires or until the task is resumed.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,20

Registers

R6 zero if indefinite suspend; otherwise, this register contains the negative number of timer units for timed suspend

Exit Conditions

Return Sequence

M.RTRN (on time-out or resume)

Registers

None

2.21 Entry Point 21 - Suspend Execution of Specified Task

This entry point is called to suspend execution of the specified task, either for an indefinite period or for the specified number of time units. The specified time (if any) is stored as a one-shot timer in the DQE of the specified task, along with a resume-program timer function code. S.EXEC16 is then called to suspend execution of the specified task. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

```
M.CALL    H.EXEC,21
```

Registers

R2	DQE of task to be suspended
R6	zero if indefinite suspend; otherwise, this register contains the negative number of timer units for timed suspend

Exit Conditions

Return Sequence

```
M.RTRN
```

Registers

None

2.22 Entry Point 22 - Go to Specified Task Context (AIDDB)

This entry point is called by AIDDB to either begin or continue processing of the task being debugged. The execution context (registers and PSD) are contained in a parameter block associated with the call. The AIDDB mode is reset and control is passed to the specified user context, by pushing the context onto the TSA stack and invoking the CPU scheduler.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,22

Registers

R1 address of context block where:

Word	Contents
0-7	R0-7
8-9	PSD

The context block must be word bounded.

Exit Conditions

Return Sequence

Control will be passed to the specified context. AIDDB will not be re-entered until a trap, break, or abort is encountered.

2.23 Entry Point 23 - Run User Break Receiver (AIDDB)

This entry point is called by AIDDB to initiate execution of the user break receiver. The contents of T.CONTXT are pushed onto the TSA stack. The AIDDB mode is reset. The user break request flag is set, and control is passed to the CPU scheduler.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,23

Registers

None

Exit Conditions

Return Sequence

Control will be passed to the user break receiver by the CPU scheduler. AIDDB will not be re-entered until a trap, break, break exit, or abort is encountered.

2.24 Entry Point 24 - Reserved

2.25 Entry Point 25 - Wait for Any No-Wait Operation Complete

This entry point is called to place the current task in a wait state, waiting for the completion of any no-wait mode I/O request, no-wait mode message request, no-wait mode run request, or the receipt of a message or break interrupt. The wait state can be either indefinite in length or can have an associated time-out value. A return is not made until one of the wait conditions is satisfied, or until expiration of the time-out value.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,25

Registers

R6 zero if indefinite suspend; otherwise, this register contains the negative number of timer units for timed suspend

Exit Conditions

Return Sequence

M.RTRN (on time-out or satisfaction of wait condition)

Registers

None

2.26 Entry Point 26 - Continue Specified Task

This entry point is called to continue a task that is in the hold state. The DQE of the specified task is unlinked from the hold-state queue and linked to the ready-to-run queue.

Note: If the task is not in the hold state, the hold request flag in the DQE is reset. A return to the calling routine is then made.

Entry Conditions

Calling Sequence

M.RTRN

Registers

R1 DQE address of task to be continued

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.27 Entry Point 27 - General Enqueue

This entry point is called to place the current task in the general wait queue (C.SWGQ). The task remains on the general wait queue until either the optional timer expires, or a corresponding general dequeue call (to H.EXEC,36) is made.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,27

Registers

R4 zero if indefinite wait, otherwise contains negative number of timer units for timed wait.

R5

Bits	Definition
0	zero if normal (priority independent) swapping (an outswapped task may be a higher priority than an inswapped task); one if the task is to be swapped only by a higher priority task
1-23	unused
24-31	function code (0-255). See DQE.GQFN.

R6,R7 enqueue ID

Exit Conditions

Return Sequence

M.RTRN (on timer expiration or dequeue call with corresponding function code and ID - with M.OPEN in effect)

Note: Swap on priority restriction removed before M.RTRN.

Registers

R3 zero if wait state terminated by corresponding dequeue call, one if wait state time-out

2.28 Entry Point 28 - Report Run Request Postprocessing Complete

This entry point is called by the run-request postprocessing logic to exit from the end action interrupt state. Its purpose is to clear the task interrupt processing lock, and to return to the point of task interrupt. It discards one level of pushdown in the TSA stack. A M.RTRN will then be issued to return to the point of task interrupt or the point following the M.ANYW call.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,28

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.29 Entry Point 29 - Report Wait Mode Run Request Starting

This entry point is called to report the beginning of processing for a wait mode run request issued by the currently executing task. The task is removed from the associated ready-to-run list, and placed in the wait for run complete list. A return to the calling routine is made upon completion of the run request by the destination task.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,29

Registers

None

Exit Conditions

Return Sequence

CPU scheduler (when run request complete)

Registers

None

2.30 Entry Point 30 - Enable AIDDB Mode Break

This entry point is called by the AIDDB program to allow a break while the task is in AIDDB mode. It is used in conjunction with H.EXEC,33 (Disable AIDDB Mode Break).

Entry Conditions

Calling Sequence

M.CALL H.EXEC,30

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.31 Entry Point 31 - Hold Current Task

This entry point is called to remove the current task from execution and place it in a hold state. Task execution does not continue until a continue request is issued to H.EXEC,26.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,31

Registers

None

Exit Conditions

Return Sequence

M.RTRN (after continue request)

Registers

None

2.32 Entry Point 32 - Hold Specified Task

This entry point is called to place the specified task in a hold state. The hold request system action interrupt flag is set in the DQE of the specified task. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,32

Registers

R1 DQE address of task to be placed in hold state

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.33 Entry Point 33 - Disable AIDDB Mode Break

This entry point is called by the AIDDB program to disable a break while the task is in AIDDB mode. This routine is provided for use in conjunction with H.EXEC,30 (Enable AIDDB Mode Break). Normally, AIDDB mode break is not enabled.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,33

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.34 Entry Point 34 - Report No-Wait Message Postprocessing Complete

This entry point is called by the message request postprocessing logic to exit from the end action interrupt state. Its purpose is to clear the task interrupt processing lock, and to return to the point of task interrupt. It discards one level of pushdown in the TSA stack. An M.RTRN is then issued to return to the point of task interrupt (or to the point following the M.ANYW call).

Entry Conditions

Calling Sequence

```
M.CALL      H.EXEC,34
```

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.35 Entry Point 35 - Report Wait Mode Message Starting

This entry point is called to report the beginning of processing for a wait mode message request issued by the currently executing task. The task is removed from the associated ready to run list, and placed in the wait for message complete list. A return to the calling routine is made when message processing by the destination task completes.

Entry Conditions

Calling Sequence

```
M.CALL      H.EXEC,35
```

Registers

None

Exit Conditions

Return Sequence

CPU scheduler (when message request complete)

Registers

None

2.36 Entry Point 36 - General Dequeue

This entry point is called to release the highest priority task queued for the specified function code and Enqueue ID. If none exist, the request is ignored.

Entry Conditions

Calling Sequence

```
M.CALL      H.EXEC,36
```

Registers

R5	function code (0-255)
R6,R7	enqueue ID

Exit Conditions

Return Sequence

M.RTRN (with M.SHUT in effect)

Registers

R2	program number of dequeued task, or zero if none dequeued

2.37 Entry Point 37 - Wait for Memory Available

This entry point is called when the required memory space is not available. The currently executing task is removed from the associated ready-to-run list, and placed in the memory request list. A return to the calling routine is made when the memory request has been satisfied.

Entry Conditions

Calling Sequence

```
M.CALL      H.EXEC,37
```

Registers

R5 bytes 0 and 1 specify the type of memory required:

Value	Memory Class
1	E
2	H
3	S
4	H1(CPU shadow)
5	H2(IPU shadow)
6	H3(CPU/IPU shadow)

Bytes 2 and 3 specify the number of memory blocks required.

R7 memory request definition word.
Byte 0 specifies the memory request type:

Value	Definition
0	inswap task
1	pre-activation request
2	activation request
3	memory expansion request
4	IOCS buffer request
6	system buffer request
7	release swap file space (see H.MEMM,8)

Bytes 2 and 3 specify the map register to be used; subtract the contents of C.MSD from the map register number (0-31).

or

R7 shared memory request definition word.
Byte 0 specifies the memory request type:

Value	Definition
5	shared memory request

Bytes 1 through 3 specify the address of the appropriate SMT entry.

Exit Conditions

Return Sequence

CPU scheduler (when memory is allocated)

Registers

None

2.38 Entry Point 38 - Inhibit Asynchronous Abort/Delete

This entry point is called to inhibit an asynchronously requested task abort or task delete. This entry point is used for gating purposes and is called when a program sequence is started that must be completed in order to maintain system integrity. Any asynchronous abort or delete requests received while abort/delete is inhibited is deferred until the system critical sequence is complete, and a call is made to H.EXEC,39 to remove the inhibit status.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,38

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.39 Entry Point 39 - Allow Asynchronous Abort/Delete

This entry point is called at the conclusion of a system critical program sequence, to remove the asynchronous abort/delete inhibit state previously invoked by a call to H.EXEC,38. Any deferred abort or delete requests are processed.

Entry Conditions

Calling Sequence

M.CALL H.EXEC,39

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.40 Entry Point 40 - End Action Wait

See M.EAWAIT or M_AWAITACTION in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.41 Entry Point 41 - Get User Context

See M_GETCTX in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.42 Entry Point 42 - Put User Context

See M_PUTCTX in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.43 Entry Point 43 - Reserved for Symbolic Debugger/X32

3 H.EXEC Subroutines

3.1 Subroutine S.EXEC1 - Interactive Input Complete

This routine is called by the appropriate I/O handler from the interrupt service routine. Its purpose is to report the completion of processing for an interactive input request. The associated task is removed from the wait for interactive input list and linked to the ready-to-run list (or to the memory-request list if an inswap is required).

Entry Conditions

Calling Sequence

BL S.EXEC1

Registers

R1 DQE entry number

Exit Conditions

Return Sequence

TRSW R0

Registers

None returned.
None saved.

3.2 Subroutine S.EXEC2 - Terminal Output Complete

This routine is called by the appropriate I/O handler from the interrupt service routine. Its purpose is to report the completion of processing for a terminal output request. The associated task is removed from the wait for terminal output list and linked to the ready-to-run list (or to the memory-request list if an inswap is required).

Entry Conditions

Calling Sequence

BL S.EXEC2

Registers

R1 DQE entry number

Exit Conditions

Return Sequence

TRSW R0

Registers

None returned.
None saved.

3.3 Subroutine S.EXEC3 - Wait I/O Complete

This routine is called by the appropriate I/O handler from the interrupt service routine. Its purpose is to report the completion of processing for a wait I/O request. The associated task is removed from the wait I/O list and linked to the ready-to-run list (or to the memory-request list if an inswap is required).

Entry Conditions

Calling Sequence

BL S.EXEC3

Registers

R1 DQE entry number

Exit Conditions

Return Sequence

TRSW R0

Registers

R6 unchanged

3.4 Subroutine S.EXEC4 - No-Wait I/O Complete

This routine is called by the appropriate I/O handler from the interrupt service routine. Its purpose is to report the completion of processing for a no-wait I/O request. The associated task may be in the wait for any I/O list. If so, it is removed from that list and linked to the ready-to-run list (or to the memory request list if an inswap is required).

The I/O queue entry is linked to the DQE task interrupt list and contains the no-wait I/O postprocessing service address. When the scheduler dispatches CPU control to this task, the specified routine is entered as a preemptive system service. Preemptive system services take precedence over execution of the task, but do not take precedence over system services being executed on behalf of the task.

Entry Conditions

Calling Sequence

```
BL      S.EXEC4
```

Registers

R1	DQE entry number
R6	I/O queue entry address (the first eight words of the I/O queue entry must be in the preemptive system service list entry header format)

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R6	unchanged

3.5 Subroutine S.EXEC4A - No Wait I/O Complete (No Postprocessing)

This routine is called by the appropriate handler from the interrupt service routine. Its purpose is to report the completion of processing for a no-wait I/O request. The associated task may be in the wait for any I/O list. If so, it is removed from that list and linked to the ready-to-run list (or to the memory request list if an inswap is required).

Entry Conditions

Calling Sequence

```
BL      S.EXEC4A
```

Registers

R1	DQE entry number
R6	I/O queue entry address

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R6	unchanged

3.6 Subroutine S.EXEC5 - Exit from Interrupt

This routine is called as an exit service by all interrupt service routines. Its purpose is to allow CPU scheduling based on events which may have occurred at an interrupt level. If lower level interrupts are active, processing continues at the last (highest) interrupted level. If no interrupts are active, the CPU scheduler is entered.

Entry Conditions

Calling Sequence

```
BEI
DAI/DACI (for associated level)
BL          S.EXEC5
```

Registers

R2	address of register save block containing registers from interrupted environment
R6,R7	PSD from interrupted environment

Exit Conditions

Return Sequence

LPSD (or) CPU scheduler

Registers

None

3.7 Subroutine S.EXEC5A - Exit from Trap Handler with Abort

This routine is called as an exit service from the system error trap handlers: nonpresent memory, undefined instruction, privilege error, address exception, and map fault. Its purpose is to request that the current task be aborted and transfer execution back to the CPU scheduler, S.EXEC20. If lower levels of interrupt are active, a system kill is executed.

Entry Conditions

Calling Sequence

```
BEI
BL          S.EXEC5A
```

Registers

R2	address of register save area
R5	abort code
R6,R7	PSD from interrupt environment

Exit Conditions

Return Sequence

CPU scheduler or M.KILL

Registers

None

3.8 Subroutine S.EXEC6 - No-Wait I/O Postprocessing Complete

This routine is called to report the completion of no-wait I/O postprocessing. Its purpose is to clear the task interrupt processing lock, and to return to the point of task interrupt. It discards one level (the most recent) of pushdown in the TSA stack. An M.RTRN is then issued to return to the point of task interrupt.

Entry Conditions

Calling Sequence

```
BL          S.EXEC6
```

Registers

None

Exit Conditions

Return Sequence

M.RTRN (to previous context)

Registers

None

3.9 Subroutine S.EXEC7 - Report Memory Pool Available

This routine is called when memory pool space is deallocated. The purpose of this subroutine is to resume the execution of all tasks in the wait for memory pool list. If the wait for memory pool list is empty, no additional processing is required and a return is made to the calling routine. Otherwise, each entry in the list is removed and placed in its associated ready-to-run list. It is expected that when these tasks resume execution, they will re-issue the request for the required memory pool space. When all entries have been flushed from the wait for memory pool list, a return is made to the calling routine.

Entry Conditions

Calling Sequence

```
BL      S.EXEC7
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R5,R6	saved
R1-R4,R7	destroyed

3.10 Subroutine S.EXEC8 - Link Entry to Queue by Priority

This routine is a register-reentrant subroutine, callable from either a software or interrupt priority level. Its purpose is to link an entry into the list associated with the designated head cell, by priority. This routine assumes that a standard head cell and entry header format are used. After the specified linkage is performed, a return is made to the calling program.

Entry Conditions

Calling Sequence

(Gating as appropriate)

```
BL      S.EXEC8
```

Registers

R1	head cell address
R2	address of entry to be linked

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R2,R4,R6,R7	saved
R1,R3,R5	destroyed

3.11 Subroutine S.EXEC9 - Unlink Entry from Queue

This routine is a register-reentrant subroutine, callable from either a software or interrupt priority level. Its purpose is to unlink the specified entry from the list associated with the designated head cell. This routine assumes that a standard head cell and entry header format are used. After the entry is unlinked, a return is made to the calling program.

Entry Conditions

Calling Sequence

(Gating as appropriate)

BL	S.EXEC9

Registers

R1	head cell address
R2	address of entry to be unlinked

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2,R4-R7	saved
R1,R3	destroyed

3.12 Subroutine S.EXEC10 - Link Entry to Bottom of Queue

This routine is a register-reentrant subroutine, callable from either a software or interrupt priority level. Its purpose is to link an entry to the bottom of the list associated with the specified head cell. This routine assumes that a standard head cell and entry header format are used. After the specified linkage is performed, a return is made to the calling program.

Entry Conditions

Calling Sequence

(Gating as appropriate)

BL	S.EXEC10

Registers

R1	head cell address
R2	address of entry to be linked

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1,R2,R4-R7	saved
R3	destroyed

3.13 Subroutine S.EXEC11 - Link Entry to Top of Queue

This routine is a register-reentrant subroutine, callable from either a software or interrupt priority level. Its purpose is to link an entry to the top of the list associated with the specified head cell. This routine assumes that a standard head cell and entry header format are used. After the specified linkage is performed, a return is made to the calling program as shown in the following text.

Entry Conditions

Calling Sequence

(Gating as appropriate)

BL	S.EXEC11

Registers

R1	head cell address
R2	address of entry to be linked

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1,R4-R7	saved
R2	address of linked entry
R3	destroyed

	0 7	8 15	16 23	24 31
Word 0	String forward address. See Note 1.			
1	String backward address. See Note 2.			
2	Priority. See Note 3.	Count. See Note 4.	Reserved	

Notes:

1. The string forward address is a one word field which points to the first entry in the top-to-bottom chain. When the list is empty, the field contains the address of the head cell.
2. The string backward address is a one word field which points to the first entry in the bottom-to-top chain. When the list is empty, the field contains the address of the head cell.
3. The head cell priority is a one byte field which contains a dummy head cell priority which is always zero.
4. The count value is a one byte field which contains the number of entries in the list. This value is incremented/decremented as required by subroutines S.EXEC8 through S.EXEC11.

	0 7	8 15	16 23	24 31
Word 0	String forward address. See Note 1.			
1	String backward address. See Note 2.			
2	Priority. See Note 3.	Available for use as defined by entry format.		

Notes:

1. The string forward address is a one word field which points to the next entry in the top-to-bottom chain. If this is the last entry in the top-to-bottom chain, the string forward address will be the address of the head cell.
2. The string backward address is a one word field which points to the next entry in the bottom-to-top chain. If this is the last entry in the chain, the string backward address will be the address of the head cell.
3. The priority field is a one byte field containing the priority of this entry. The acceptable range of this value is 1-255.

   Priority zero is reserved for use as a dummy priority by the head cell.

The last entry in the top-to-bottom chain is the first entry in the bottom-to-top chain. The last entry in the bottom-to-top chain is the first entry in the top-to-bottom chain.

3.14 Subroutine S.EXEC12 - Report Memory Scheduler Event

This routine is called when the status of allocated memory changes (it is either deallocated or becomes more eligible for swapping). The purpose of this subroutine is to insure the appropriate execution of the memory scheduler task. If the memory-request list is empty, no additional processing is required and a return is made to the user. If the memory-request list is not empty, C.RRUN is incremented, and the memory scheduler state is checked. If the memory scheduler is in the wait for memory event list, it is removed from that list and placed in the ready-to-run list at the priority of the highest priority entry in the memory-request list. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

```
BL      S.EXEC12
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R3,R6	saved
R1,R2,R4,R5,R7	destroyed

3.15 Subroutine S.EXEC13 - Break Specified Task

This routine is called by the appropriate I/O handler from the interrupt service routine, or by the M.INT monitor service. Its purpose is to set the AIDDB-break requested flag in the DQE if AIDDB is associated with the task and if the AIDDB mode active task interrupt flag is not already set. If AIDDB is not associated with the task, but a user break receiver has been established, the user-break requested flag is set in the DQE. If AIDDB is not associated with the task, and no user break receiver has been established, the break is ignored. A return to the calling routine is made upon completion of processing.

Entry Conditions

Calling Sequence

BL	S.EXEC13

Registers

R2	DQE address of task to receive break
R3	address of the 22 word scratchpad area (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3-R3-4W	scratchpad address
R1,R2,R4	saved
R5,R6,R7	destroyed

3.16 Subroutine S.EXEC14 - Resume Specified Task

This routine can be called either from an interrupt service routine or from a system service operating on behalf of a task. Its purpose is to resume the execution of a suspended task. If the specified task is not in a suspended state, no action is taken. If the specified task is suspended and outswapped, it is unlinked from the suspended list, and linked to the memory request list. Otherwise, it is unlinked from the suspended list, and linked to the ready-to-run list at its current priority.

Entry Conditions

Calling Sequence

BL	S.EXEC14

Registers

R2	DQE address of task to be resumed

Exit Conditions

Return Sequence

TRSW R0

Registers

None returned.

3.17 Subroutine S.EXEC20 - CPU Scheduler

This routine is an internal H.EXEC subroutine. It is called by S.EXEC5 when all outstanding interrupts or traps have been exited, or by M.RTRN/M.RTNA with the return context on the TSA stack. Its purpose is to check for a ready-to-run task that is higher in priority than the currently executing task. This check is quickly made because the linkage of any task to the ready-to-run queue will cause a priority comparison between that task and the currently executing task. If the newly ready-to-run task is of higher priority, an indicator is set for S.EXEC20. S.EXEC20 either returns to the context of the current task, processes a task interrupt on behalf of the current task, or selects a higher priority task for execution.

Entry Conditions

Calling Sequence

BU S.EXEC20

Registers

None

Exit Conditions

Return Sequence

Dispatch of CPU control

There are five paths through the scheduler (S.EXEC20):

1. task scheduled for execution (see Figure 3-1)
2. context switch inhibited, current task resumed (see Figure 3-2)
3. time quantum two has expired (see Figure 3-3)
4. higher priority task requested processor (see Figure 3-4)
5. no other tasks requesting CPU, current task resumed (see Figure 3-4)

Paths two and five resume the task previously in control of the CPU; no context switch takes place.

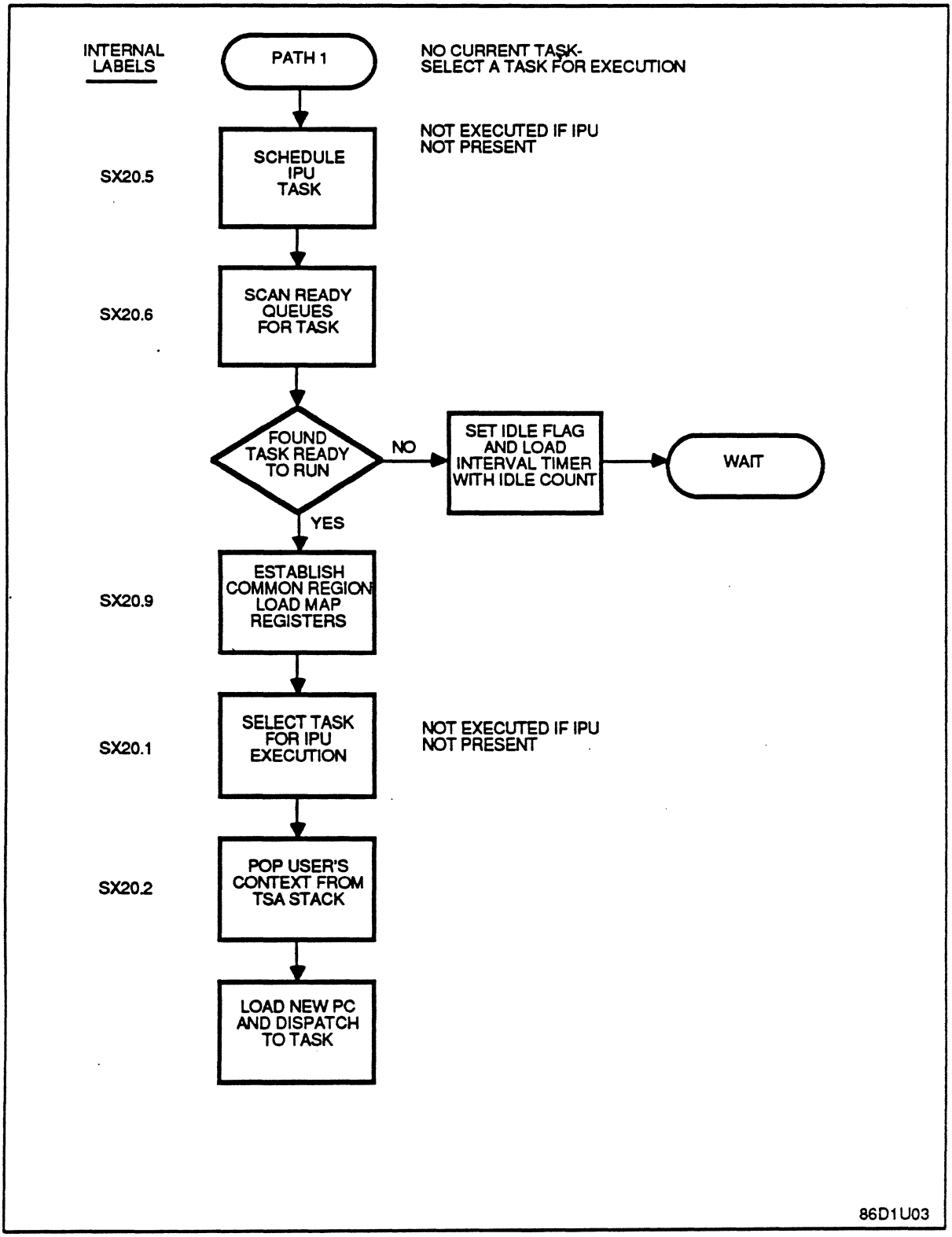


Figure 3-1
S.EXEC20 Path One

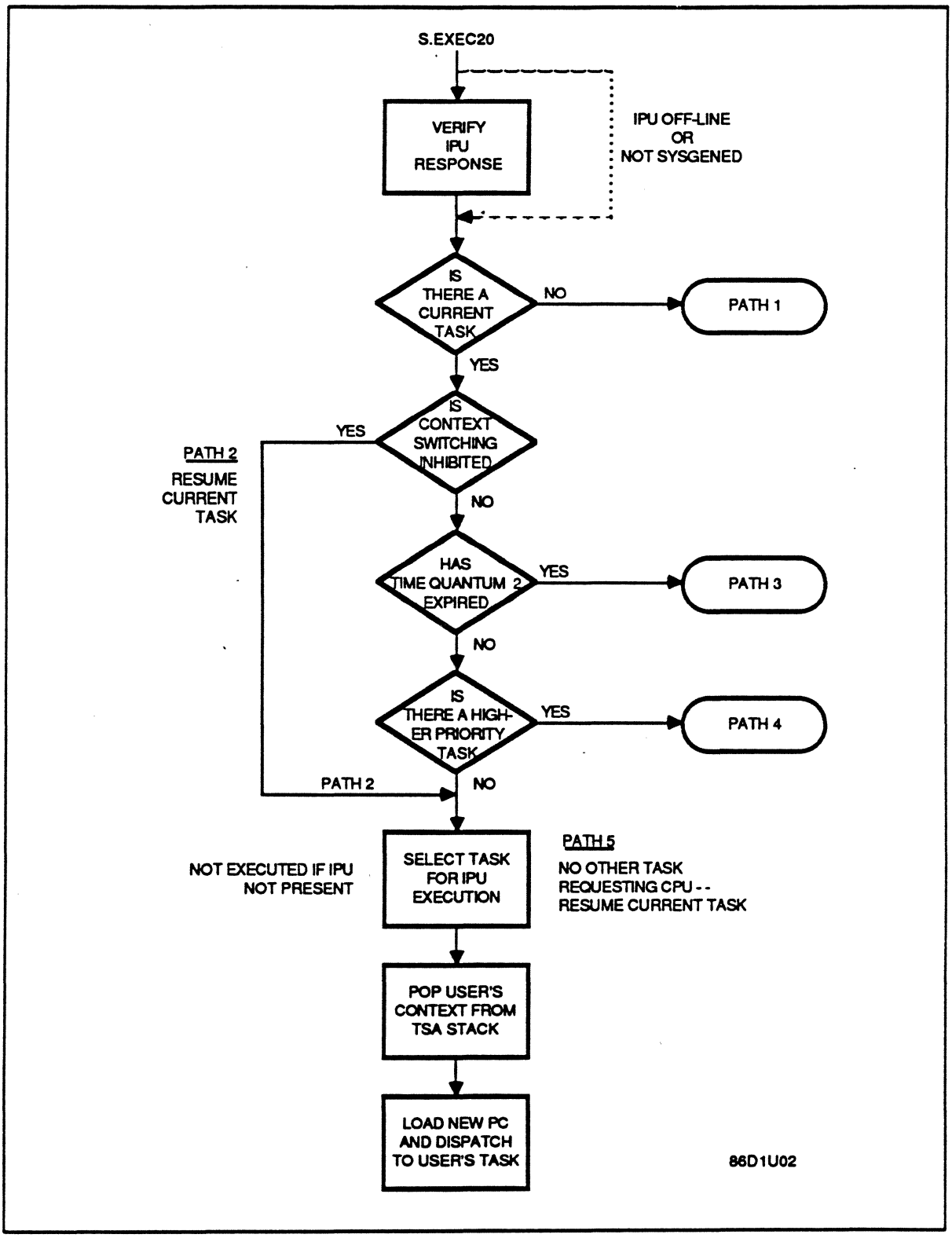


Figure 3-2
S.EXEC20 Path Two and Five

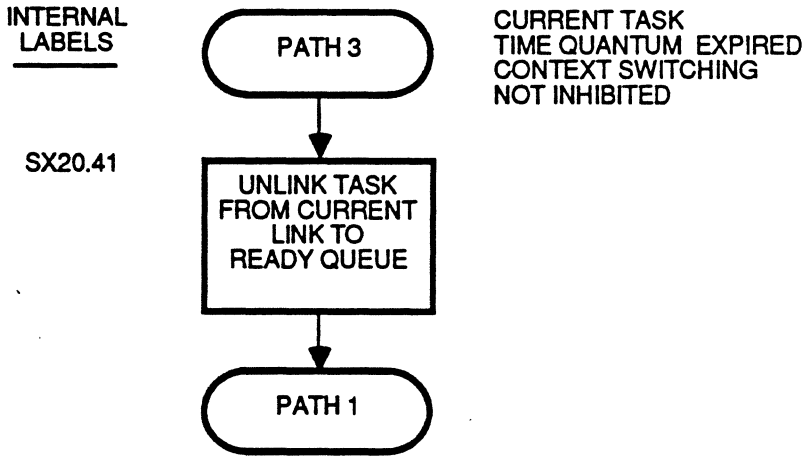


Figure 3-3
S.EXEC20 Path Three

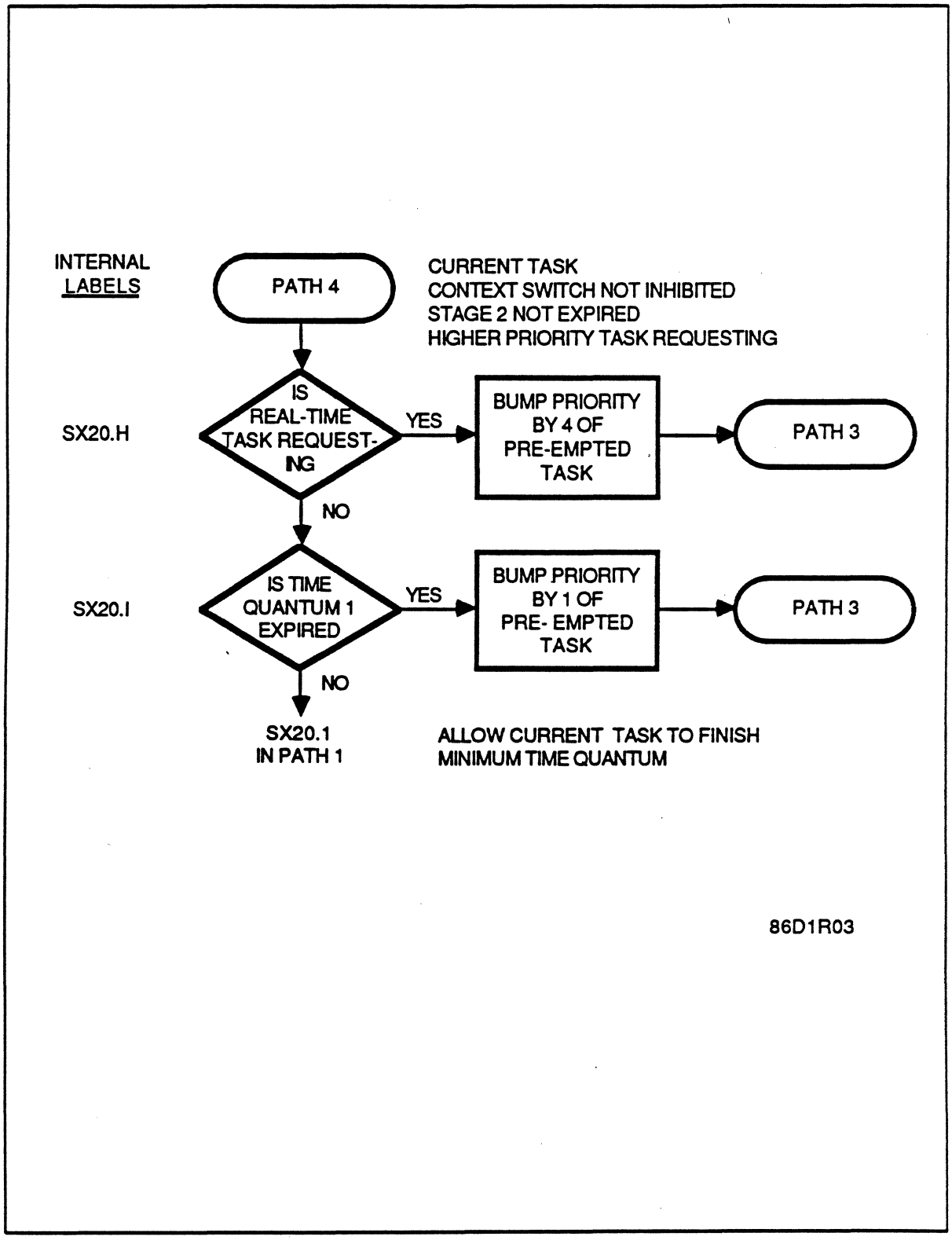


Figure 3-4
S.EXEC20 Path Four

3.18 Subroutine S.EXEC21 - Process Task Interrupt

This routine is called by S.EXEC20 to process any task interrupt requests, after the CPU scheduler has determined that the return address in the task context is not in the operating system area. It processes both system action interrupt requests in DQE.SAIR, and requested software task interrupts in DQE.RTI.

Entry Conditions

Calling Sequence

BU S.EXEC21

Registers

R2 DQE address

Exit Conditions

Return Sequence

Transfer control as appropriate for task interrupt, or return to interrupted context.

Task Interrupt Request Processing (S.EXEC21)

System Action Task Interrupts (DQE.SAIR):

Priority (bit)	Description	Processing routine
0	delete task request (DQE.DELR)	S.EXEC28
1	reserved	
2	hold task request (DQE.HLDR)	S.EXEC51
3	abort task request (DQE.ABTR)	S.EXEC19
4	exit task request (DQE.EXTR)	S.EXEC29
5	suspend task request (DQE.SUSR)	S.EXEC21
6	run request (DQE.RRRQ)	S.EXEC21
7	reserved	

Requested Software Task Interrupts (DQE.RTI):

Priority (bit)	Description
0	reserved
1	end action request (Priority 1) (DQE.EA1R)
2	debug break request (DQE.DBBR)
3	user break request (DQE.UBKR)
4	end action request (Priority 2) (DQE.EA2R)
5	message interrupt request (DQE.MSIR)
6-7	reserved

3.19 Subroutine S.EXEC23 - Unlink Messages in Receiver Queue

This routine is called from one of the task termination processing control subroutines (S.EXEC19, S.EXEC28, or S.EXEC29). Its purpose is to unlink all messages from the receiver queue and to terminate these messages, relinking any waiting tasks to their respective ready-to-run queues. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

```
BL      S.EXEC23
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

All registers destroyed

3.20 Subroutine S.EXEC24 - Reserved

3.21 Subroutine S.EXEC25 - Terminate Next Run in Queue

This routine is called from one of the task termination processing control subroutines (S.EXEC19, S.EXEC28, or S.EXEC29). Its purpose is to unlink the next run request from the receiver queue, and to terminate the requests with abnormal status. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

```
BL      S.EXEC25
```

Registers

```
R3      scratchpad address (doubleword bounded)
```

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3=R3-4W	scratchpad address
R1,R2,R4-R7	destroyed

3.22 Subroutine S.EXEC27 - Transfer Control to Abort Receiver

This routine is called from the abort task processing control subroutine (S.EXEC19). Its purpose is to transfer control to the user task abort receiver if one exists. Otherwise, a return is made to the calling routine.

Entry Conditions

Calling Sequence

BL	S.EXEC27

Registers

R2	current task DQE address

Exit Conditions

Return Sequence

TRSW	R0 (or LPSD to abort receiver)

Registers

All registers preserved

3.23 Subroutine S.EXEC30 - Reserved

3.24 Subroutine S.EXEC31 - No-Wait Run Request

This routine is called to report the completion of no-wait run request postprocessing. Its purpose is to clear the task interrupt processing lock, and to return to the point of task interrupt. It discards the most recent level of push down in the TSA. A M.RTRN is then issued to return to the point of task interrupt.

Entry Conditions

Calling Sequence

```
BL      S.EXEC31
```

Registers

None

Exit Conditions

Return Sequence

M.RTRN (to previous context)

Registers

None

3.25 Subroutine S.EXEC34 - Reserved

3.26 Subroutine S.EXEC35 - Report No-Wait Postprocessing

This routine is called to report the completion of no-wait mode message postprocessing. Its purpose is to clear the task interrupt processing lock, and to return to the point of task interrupt. It discards the most recent level of pushdown in the TSA stack. An M.RTRN is then used to return to the point of task interrupt.

Entry Conditions

Calling Sequence

```
BL      S.EXEC35
```

Registers

None

Exit Conditions

Return Sequence

CPU scheduler (to previous context)

Registers

None

3.27 Subroutine S.EXEC40 - Reserved

3.28 Subroutine S.EXEC41 - Exit Run Receiver

This routine is called to exit a run receiver when the M.XRUNR exit type is invoked. Its purpose is to process the exit according to the specifications contained in the receiver exit block (RXB). The run receiver queue will be examined, and if not empty, the task is executed again at the point following the M.XRUNR call on behalf of the next request. If the queue is empty, the exit options are examined. If option bit 0 is set, the task is placed in a wait-state, waiting for the next run request to be received. If option bit 0 is reset, the task exits the system.

Entry Conditions

Calling Sequence

```
BL      S.EXEC41
```

Registers

R1	current task DQE address
R2	receiver exit block (RXB) address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

No return (rerun task or exit)

3.29 Subroutine S.EXEC42 - Exit Message Receiver

This routine is called to exit a message receiver when a M.XMSGR service has been called. If the message interrupt is not active, the task is aborted. Its purpose is to reset the task interrupt lock and to process the exit according to the specifications in the receiver exit block (RXB). The message receiver queue is examined, and if not empty, the message interrupt is invoked again on behalf of the next request. If the queue is empty, a return is made to the point of interrupt or following M.SUSP/M.ANYW at the base execution level.

Entry Conditions

Calling Sequence

```
BL      S.EXEC42
```

Registers

R1	current task DQE address
R2	receiver exit block (RXB) address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

No return to caller (rerun message receiver or return to user base level context)

3.30 Subroutine S.EXEC44 - Change Priority of Current Task

This routine is called to change the priority level of the current task. The specified priority level is stored in DQE.CUP and DQE.BUP and as the priority level of the currently executing task. No relink of this task is required since it is linked to the special state chain for the currently executing task. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

BL	S.EXEC44

Registers

R6	priority level

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2	current task DQE address
R1,R3-R7	saved

3.31 Subroutine S.EXEC46 - Reserved

3.32 Subroutine S.EXEC47 - Reserved

3.33 Subroutine S.EXEC55 - Link Task to Ready to Run List

This routine is called to link a task to the ready-to-run queue. It unlinks the task from the designated list, then links the task to the ready list associated with its current priority.

If the optional CPU/IPU scheduler has been selected by specifying the SYSGEN DELTA directive, the task is linked at its base priority minus the delta value if the following criteria is true:

1. The task is IPU biased.
2. The task is real time.
3. The task has none of the following characteristics:
   - IPU inhibited,
   - outstanding system or pseudosystem action requests,
   - PSD is in the operating system.

If the task does not meet any of these criteria, the task is linked to the ready-to-run list as described above. If the task is outswapped and cannot run, it is linked to the memory request queue and the memory scheduler is resumed.

Entry Conditions

Calling Sequence

```
BEI
BL          S.EXEC55
```

Registers

R1	designated list headcell address
R2	DQE address

Exit Conditions

Return Sequence

```
TRSW        R0
```

Registers

R2,R4,R7	saved
R1,R3,R5,R6	destroyed

3.34 Subroutine S.EXEC56 - Resume Memory Scheduler

This routine is called as a result of a memory scheduler event. An immediate return is made if no memory requests are queued. Otherwise, the memory scheduler is made ready to run at the priority of the highest priority memory request queued.

Entry Conditions

Calling Sequence

```
BEI
BL      S.EXEC56
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1-R3,R5	destroyed
R4,R6,R7	saved

3.35 Subroutine S.EXEC57 - Link Task to Ready List by Priority

This routine is called to link the designated task to the ready-to-run list associated with its current priority.

Entry Conditions

Calling Sequence

```
BEI
BL      S.EXEC57
```

Registers

R2	DQE address of specified task

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2,R4,R6,R7	saved
R1,R3,R5	destroyed

3.36 Subroutine S.EXEC59 - Reserved

3.37 Subroutine S.EXEC61 - Transfer Parameters from MRRQ

This routine is called for the destination task after a request for message or run request parameters has been made. The subroutine transfers the sent parameters from the MRRQ entry to the receiver buffer specified in the PRB.

Entry Conditions

Calling Sequence

BL	S.EXEC61

Registers

R1	MRRQ address
R2	PRB address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1,R2,R4,R5	saved
R3=R3-4W	scratchpad address
R6	status code: 0=OK, 4=receiver buffer length exceeded
R7	destroyed

3.38 Subroutine S.EXEC62 - Validate RXB

This routine is called to validate the receiver exit block (RXB) after the destination task has issued an exit from message or run request processing.

Entry Conditions

Calling Sequence

BL	S.EXEC62

Registers

R2	RXB address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0 (CC1 set indicates validation error)

Registers

R1,R4,R5,R7	destroyed
R2	saved
R3=R3-4W	scratchpad address
R6	error code if CC1 set; otherwise, R6 is destroyed

3.39 Subroutine S.EXEC68 - Construct and Vector Context

This routine is called by the send request postprocessing logic to vector to a user-specified end action routine. Control is transferred with mapped, unblocked status.

Entry Conditions

Calling Sequence

BL	S.EXEC68

Registers

R1	PSB address
R6	user end action routine address

Exit Conditions

Return Sequence

No return. LPSD to user end action routine.

Registers

R2 PSB address

3.40 Subroutine S.EXEC69 - Postprocessing Merge Point

This routine is called to remove the task interrupt processing lock and mark the task interrupt request if additional end action entries are queued. The most recent level of context in the TSA stack is discarded, and an M.RTRN is issued to pop to the previous context level.

Entry Conditions

Calling Sequence

BL S.EXEC69

Registers

R1 current task DQE address

Exit Conditions

Return Sequence

No return to caller. M.RTRN to previous context level.

Registers

None

3.41 Subroutine S.EXEC72 - Report Wait I/O Starting

This routine is called to process a report of wait mode I/O starting. A check is made to see if the associated I/O has already completed. If so, an immediate return is made. Otherwise, the swap inhibit flag is set, unless R0 bit 1 is set, and the task is linked to the designated wait list.

Entry Conditions

Calling Sequence

BU	S.EXEC72

Registers

R0	bit 0 set indicates task is swappable during I/O
R1	address of wait list headcell

Exit Conditions

Return Sequence

No return to caller. CPU scheduler return to context on TSA stack when I/O complete.

Registers

None

3.42 Subroutine S.EXEC75 - Situational Priority Increment

This routine is called to increment the priority of the specified task. Priority adjustment is bypassed if the specified task is a real time task.

Entry Conditions

Calling Sequence

BL	S.EXEC75

Registers

R2	DQE address of target task
R4	situational priority increment

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1-R4,R6,R7	saved
R5	destroyed

3.43 Subroutine S.EXEC77 - Reserved

3.44 Subroutine S.EXEC79 - Push Current Context onto Stack

This routine is called to format a PSD with the privileged mode set, the condition codes clear, the extended addressing mode clear, the right halfword instruction clear, the arithmetic exception trap clear, and the PC pointing into H.EXEC25 in word 1. Word 2 has the map mode set and CPIX of the specified task. The subroutine then places the PSD and the registers with the contents at the end of the subroutine onto the stack. After placing the information onto the stack, the subroutine returns to the calling routine.

Entry Conditions

Calling Sequence

BL	S.EXEC79

Registers

R2	DQE address of specified task

Exit Conditions

Return Sequence

R1,R4,R5	destroyed
R2,R3,R6,R7	unchanged

3.45 Subroutine S.EXEC80 - Start IPU and Verify

When this routine is called and IPU inhibit context switch is not set, it resets the IPU run flag, starts the IPU and then initializes and starts the IPU verification timer. The subroutine then returns control to the calling routine.

Entry Conditions

Calling Sequence

BL	S.EXEC80

Registers

None

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1-R3,R5-R7	unchanged
R4	destroyed

3.46 Subroutine S.EXEC81 - Enter Debugger Entry Point Four

This routine enters the base mode debugger's entry point 4.

Entry Conditions

Calling Sequence

BL	S.EXEC81

Registers

R2	base register stack level. Used to load the task's base registers as needed by the dispatcher and the debugger.

Exit Conditions

Return Sequence

M.RTRN
or

M.RTRN	CC1 set, if error

Registers

R7	status, if error

3.47 Subroutine S.EXEC82 - Push Calling Task onto the Stack

This subroutine pushes the calling task context onto the task's TSA stack. A return PSW is supplied to specify the instructions to be executed when that TSA stack level is popped.

This subroutine cannot be used by the IPU.

Entry Conditions

Calling Sequence

BL	S.EXEC82

Registers

R4	return PSW

Exit Conditions

Return Sequence

TRSW	R0

Executive Subroutine Module (H.EXSUB)

MPX-32 Technical Manual

Volume II

Contents

Contents

1 H.EXSUB Overview

1.1 General Information

The Executive Subroutine Module (H.EXSUB) lists subroutines called by the Executive Module (H.EXEC). These subroutines involve scheduling tasks for allocation by the CPU and IPU.

The subroutines of H.EXSUB can reside in extended memory.

1.2 Subroutine Summary

Subroutine	Description
S.EXEC15	suspend execution of current task
S.EXEC16	suspend execution of specified task
S.EXEC17	abort current task
S.EXEC18	abort specified task
S.EXEC19	abort task processing control
S.EXEC22	wait for completion of all no-wait operations
S.EXEC26	remove task gating
S.EXEC28	delete task processing control
S.EXEC29	exit task processing control
S.EXEC32	report wait mode run request complete
S.EXEC33	report no-wait mode run request complete
S.EXEC36	report wait mode message complete
S.EXEC37	report no-wait mode message complete
S.EXEC38	inhibit swap of current task
S.EXEC39	enable swap of current task
S.EXEC43	reactivate run receiver task
S.EXEC45	change priority level of specified task
S.EXEC48	convert task number to DQE address
S.EXEC49	construct MRRQ
S.EXEC50	link MRRQ to run receiver of destination task
S.EXEC51	link current task to designated wait state
S.EXEC52	message or run request postprocessing
S.EXEC53	validate PSB
S.EXEC54	move byte string
S.EXEC58	link MRRQ to message receiver of destination task
S.EXEC60	validate PRB
S.EXEC63	transfer return parameters from destination task to MRRQ
S.EXEC64	no-wait mode message postprocessing
S.EXEC65	no-wait mode run request postprocessing
S.EXEC66	deallocate MRRQ
S.EXEC67	link entry to end action queue
S.EXEC70	terminate all run requests in receiver queue of current task
S.EXEC71	insure start-up of destination run receiver task
S.EXEC73	replace context on TSA stack
S.EXEC74	reset stack to user level
S.EXEC76	update task execution accounting value
S.EXEC78	move context from stack to T.CONTXT

2 H.EXSUB Subroutines

2.1 Subroutine S.EXEC15 - Suspend Execution of Current Task

This routine is called to suspend execution of the current task. The DQE for the current task is unlinked from the ready-to-run list, and linked to the suspended list. Control is then transferred to the CPU scheduler to select the next task for execution.

Entry Conditions

Calling Sequence

BL S.EXEC15

Registers

R6 zero if indefinite suspend; otherwise, contains negative timer units

Exit Conditions

Return Sequence

Branch to CPU scheduler; when resumed, the task continues operation at the most recent context in the pushdown stack.

2.2 Subroutine S.EXEC16 - Suspend Execution of Specified Task

This routine is called to suspend execution of the specified task. The DQE is marked to indicate that an asynchronous suspend has been requested. The suspend request is processed on behalf of the task being suspended, when the CPU scheduler selects that task for execution.

Entry Conditions

Calling Sequence

BL S.EXEC16

Registers

R2 DQE address of task to be suspended

R6 zero if indefinite suspend; otherwise, contains negative timer units

Exit Conditions

Return Sequence

TRSW R0

Registers

None returned.

2.3 Subroutine S.EXEC17 - Abort Current Task

This routine is called either from a system service, or from a system trap level. Its purpose is to store the abort code and set the abort requested bit in the DQE. It then resets the TSA stack to a level routine.

If the task has an abort receiver established, the DQE.ABRA flag is set and a return is made to the calling routine. If no abort receiver is established, the DQE.ABRT flag is set and the task is marked as leaving the system (by setting the DQE.TLVS flag). A return is then made to the calling routine.

Entry Conditions

Calling Sequence

BL S.EXEC17

Registers

R5	abort code characters 1-4
R6,R7	abort code characters 5-12

Exit Conditions

Return Sequence

TRSW R0

Registers

R2	DQE address of current task
R4,R7	saved
R1,R3,R5,R6	destroyed

2.4 Subroutine S.EXEC18 - Abort Specified Task

This routine is called from a system service. Its purpose is to store the abort code and set the abort requested asynchronous bit in the DQE. It is then returned to the calling routine. The abort requested bit is not examined until the scheduler selects this task for execution.

If the specified task is in the SUSP, ANYW, HOLD or RUNW list, it is unlinked from the wait list and linked to the ready-to-run list. If the task does not have an abort receiver established, DQE.TLVS and DQE.ABRT are set. If an abort receiver has been established, only DQE.ABRA is set.

Entry Conditions

Calling Sequence

```
BL      S.EXEC18
```

Registers

R1	DQE address of task to be aborted
R5	abort code characters 1-4
R6,R7	abort code characters 5-12

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R2	DQE address of task to be aborted
R4,R7	saved
R1,R3,R5,R6	destroyed

2.5 Subroutine S.EXEC19 - Abort Task Processing Control

This routine is an internal H.EXEC subroutine which is called only by S.EXEC21 to process an abort request for the currently executing task. The purpose of S.EXEC19 is to control the sequencing of abort processing by calling the following abort processing modules associated with the pertinent subsystems.

Module	Description
S.EXEC26	remove context switch gating, if any
S.EXEC23	terminate all messages in receiver queue
S.EXEC22	defer continued processing until all outstanding no-wait operations are complete
S.EXEC25	terminate active run request, if any
S.EXEC43	reactivate task if additional run requests queued
S.EXEC27	transfer control to abort receiver, if any
S.REXS2	remove task associated timers
H.REXS38	disconnect interrupt (if interrupt connected)
S.EXEC76	update execution accounting value

Module	Description
[H.TSM,4]	TSM task abort/delete processing (called only if TSM task)
H.REMM,3	close and deallocate system critical files
	close and deallocate user I/O files/devices with blocking buffer purge and automatic EOF as appropriate
	deallocate all swap files and memory (excluding TSA)
	deallocate TSA memory, DQE space, clear C.PRNO, transfer control to the CPU scheduler

Entry Conditions

Calling Sequence

BU	S.EXEC19

Registers

None

Exit Conditions

Return Sequence

Clear	C.PRNO
BU	S.EXEC20

Registers

None

2.6 Subroutine S.EXEC22 - Completion of All No-Wait Operations

This routine is called from either the task abort or task exit processing control subroutines (S.EXEC19 or S.EXEC29). Its purpose is to delay until all outstanding no-wait processing is complete. It accomplishes this by calling H.EXEC,25 until the number of outstanding no-wait requests is equal to zero. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

BL	S.EXEC22

Registers

R2	current task DQE address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1-R5,R7 saved
R6 destroyed

2.7 Subroutine S.EXEC26 - Remove Task Gating

This routine is called from one of the task termination processing subroutines (S.EXEC19, S.EXEC28, or S.EXEC29). Its purpose is to remove any outstanding gating mechanisms, like a context switch or resource mark lock, associated with the terminating task. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

BL S.EXEC26

Registers

R2 current task DQE address

Exit Conditions

Return Sequence

TRSW R0

Registers

All registers are preserved.

2.8 Subroutine S.EXEC28 - Delete Task Processing Control

This routine is an internal H.EXEC subroutine which is called only by S.EXEC21 to process a task delete request on behalf of the currently executing task. The purpose of S.EXEC28 is to control the sequencing of the following delete task processing modules associated with the pertinent subsystems.

Module	Description
S.REXS2	remove task associated timers
H.REXS,38	disconnect interrupt (if connected)
S.EXEC26	remove context switch gating if any
H.IOCS,38	terminate all outstanding I/O requests

Module	Description
S.EXEC23	terminate all messages in receiver queue
S.EXEC25	terminate active run request (if any)
H.REMM,3	close and deallocate system critical files
	close and deallocate user I/O files/devices with minimal processing required (no attempt to preserve data integrity)
	deallocate all swap files and memory (excluding TSA)
	deallocate TSA memory, DQE space, clear C.PRNO, transfer control to CPU scheduler
S.EXEC43	reactivate task if additional run requests queued
S.EXEC76	update task execution accounting value
[H.TSM,4]	TSM task abort/delete processing (called only if TSM task)
S.REMM2	deallocate TSA and DQE

Entry Conditions

Calling Sequence

BU	S.EXEC28

Registers

None

Exit Conditions

Return Sequence

Clear	C.PRNO
BU	S.EXEC20

Registers

None

2.9 Subroutine S.EXEC29 - Exit Task Processing Control

This routine is an internal H.EXEC subroutine which is called only by S.EXEC21 to process an exit request on behalf of the currently executing task. The purpose of S.EXEC29 is to control the sequencing of exit processing by the following exit processing modules associated with the pertinent subsystems.

Module	Description
S.REXS2	remove task associated timers
H.REXS,38	disconnect interrupt (if connected)
S.EXEC26	remove context switch or FISE gating, if any

Module	Description
S.EXEC22	defer continued processing until all outstanding no-wait operations are complete
S.EXEC25	terminate active run request, if any
H.REMM,3	close and deallocate system critical files
	close and deallocate user I/O files/devices with blocking buffer purge and automatic EOF as appropriate
	deallocate all swap files and memory (excluding TSA)
	deallocate TSA memory, DQE space, clear CPRNO, transfer control to CPU scheduler
S.EXEC43	reactivate task if additional run requests queued
S.EXEC76	update task execution accounting value
[H.TSM,3]	TSM task exit processing (called only if TSM task)
S.REMM2	deallocate TSA and DQE

Entry Conditions

Calling Sequence

BU	S.EXEC29

Registers

None

Exit Conditions

Return Sequence

Clear	C.PRNO
BU	S.EXEC20

Registers

None

2.10 Subroutine S.EXEC32 - Completion of Wait Run Request

This routine is called by the appropriate run request exit processor on behalf of the requested task. Its purpose is to report completion of the wait mode run request to the requesting (waiting) task. The waiting task is removed from the wait list and placed in the ready-to-run list (or in the memory request list if an inswap is required).

Entry Conditions

Calling Sequence

BL	S.EXEC32

Registers

R1	DQE entry address of sending task
R2	MRRQ address
R3	address of 22 word scratchpad area (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3=R3-4W	scratchpad address
R1,R2,R4-R7	destroyed

2.11 Subroutine S.EXEC33 - Report No-Wait Run Request Complete

This routine reports the completion of run request processing. The call is made on behalf of the task which processed the run request. The requesting task may be in the wait for any run request completion state. If so, it is removed from that list and linked to the ready-to-run list (or to the memory request list if an inswap is required).

The run request queue entry is linked to the DQE task interrupt list and contains the no-wait mode run request postprocessing service address. When the scheduler dispatches control to the task, the specified routine is entered as a preemptive system service.

Entry Conditions

Calling Sequence

BL	S.EXEC33

Registers

R1	DQE address of sending task
R2	MRRQ address
R3	address of 22 word scratchpad area (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3=R3-4W	scratchpad address
R1,R2,R4-R7	destroyed

2.12 Subroutine S.EXEC36 - Report Wait Mode Message Complete

This routine is called by the appropriate message exit processor on behalf of the task that processed the message. Its purpose is to report completion of wait mode message processing to the waiting task. The waiting task is removed from the wait list and placed in the ready-to-run list or in the memory request list if an inswap is required.

Entry Conditions

Calling Sequence

BL	S.EXEC36

Registers

R1	DQE entry address of sending task
R2	MRRQ address
R3	address of 22 word scratchpad area (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3=R3-4W	scratchpad address
R1,R2,R4-R7	destroyed

2.13 Subroutine S.EXEC37 - No-Wait Message Mode Complete

This routine is called to report the completion of message processing. The call is made on behalf of the task which processed the message. The task which sent the message may be in the wait for any message completion queue. If so, it is removed from that list and linked to the ready-to-run list (or to the memory request list if an inswap is required).

The message queue entry is linked to the DQE task interrupt list and contains the no-wait mode message postprocessing service address. When the scheduler dispatches control to the task, the specified routine is entered as a preemptive system service.

Entry Conditions

Calling Sequence

```
BL      S.EXEC37
```

Registers

R1	DQE address of requesting task
R2	MRRQ address
R3	address of 22 word scratchpad area (doubleword bounded)

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R3=R3-4W	scratchpad address
R1,R2,R4-R7	destroyed

2.14 Subroutine S.EXEC38 - Inhibit Swap of Current Task

This routine is called to set the inhibit swap flag (DQE.LKIM) in the DQE of the current task. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

```
BL      S.EXEC38
```

Registers

None

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2	destroyed
R1,R3-R7	saved

2.15 Subroutine S.EXEC39 - Enable Swap of Current Task

This routine is called to reset the inhibit swap flag (DQE.LKIM) in the DQE of the current task. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

BL	S.EXEC39

Registers

None

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2	destroyed
R1,R3-R7	saved

2.16 Subroutine S.EXEC43 - Reactivate Run Receiver Task

This routine is called to examine the run receiver queue of a run receiver task that has used a standard M.EXIT call. S.EXEC43 is called by S.EXEC28, S.EXEC29, or S.EXEC19. If queued run requests exist, a call to H.REMM,1 is made to reactivate the task. If H.REMM,1 makes a denial return, all outstanding requests are terminated with abnormal status. If H.REMM,1 successfully starts the activation, S.EXEC43 links any remaining queued requests to the DQE of the task being activated. A return is then made to the calling routine.

Entry Conditions

Calling Sequence

```
BL      S.EXEC43
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

All registers are destroyed.

2.17 Subroutine S.EXEC45 - Change Priority of Specified Task

This routine is called to change the priority level of the specified, not current, task. The specified task may either be in a ready-to-run state, or in a wait state. If the task is in a ready-to-run state, the specified priority is stored in DQE.CUP and in DQE.BUP. The task is then unlinked from its current ready-to-run list and relinked to the ready to run list associated with the new priority. If the task was in a wait state, it is relinked according to its new priority into the same wait list. A return is then made to the calling program.

Entry Conditions

Calling Sequence

```
BL      S.EXEC45
```

Registers

R1	DQE address of specified task
R6	priority level

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

All registers are destroyed.

2.18 Subroutine S.EXEC48 - Convert Number to DQE Address

This routine is called to calculate the DQE address of the specified task.

Entry Conditions

Calling Sequence

BL	S.EXEC48

Registers

R7	task activation sequence number of specified task

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2	DQE address of specified task
R1,R3-R5,R7	saved
R6	destroyed

2.19 Subroutine S.EXEC49 - Construct MRRQ

This routine is called to construct an MRRQ entry for either a message or run request. Space for the MRRQ is allocated from memory pool. The MRRQ is constructed according to the contents of the parameter send block (PSB) specified as a calling parameter.

Entry Conditions

Calling Sequence

BL	S.EXEC49

Registers

R2	parameter send block (PSB) address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0 (CC1 set indicates memory pool unavailable)

Registers

R1	MRRQ address
R3=R3-4W	scratchpad address
R2,R4	saved
R5-R7	destroyed

2.20 Subroutine S.EXEC50 - Link MRRQ to Receiver Queue

This routine is called to link the designated MRRQ entry to the run receiver queue of the specified task. If the target task is in a RUNW wait state, it is unlinked from the wait list and linked to the ready-to-run list.

Entry Conditions

Calling Sequence

BL	S.EXEC50

Registers

R1	target task DQE address
R2	MRRQ address
R3	scratchpad address (doubleword bounded)
R5	zero indicates unlink if in PREA list
R7	task activation sequence number of target task

Exit Conditions

Return Sequence

TRSW	R0 (CC1 set indicates invalid target task)

Registers

R3=R3-4W	scratchpad address
R1,R2	saved
R4-R7	destroyed

2.21 Subroutine S.EXEC51 - Link Current Task to Wait State

This routine is called to place the currently executing task in the designated wait state. If the task is a time-distribution task, the execution time accounting value is updated in the TSA. The current quantum value for next dispatch is updated in DQE.CQC for both real-time and time-distribution tasks. When linkage to the wait state is complete, the memory scheduler is resumed if entries are queued in the memory request list.

Entry Conditions

Calling Sequence

```
BEI
BL      S.EXEC51
```

Registers

R1	address of wait state headcell
R2	current task DQE address

Exit Conditions

Return Sequence

No return to calling routine. M.RTRN to TSA stack context when task is ready to run.

Registers

None

2.22 Subroutine S.EXEC52 - Message/Run Request Processing

This routine is called for the sending task when a message or run request has been processed by the destination task. It transfers the return parameters to the return buffer designated in the PSB, updates PSB status, and deallocates the MRRQ.

Entry Conditions

Calling Sequence

```
BL      S.EXEC52
```

Registers

R2	MRRQ address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3=R3-4W	scratchpad address
R5	saved
R1,R2,R4,R6,R7	destroyed

2.23 Subroutine S.EXEC53 - Validate PSB

This routine is called to validate the parameters contained in the parameter send block (PSB) associated with a message or run request. An immediate return is made if the most recent context on the TSA stack reflects a privileged caller. Otherwise, S.REMM20 is called to verify the PSB address arguments, and general parameter validation is performed.

Entry Conditions

Calling Sequence

BL	S.EXEC53

Registers

R2	PSB address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0 (CC1 set indicates validation error)

Registers

R3=R3-4W	scratchpad address
R6	contains error code if CC1 set, otherwise destroyed
R2	saved
R1,R4,R5,R7	destroyed

2.24 Subroutine S.EXEC54 - Move Byte String

This routine is a register reentrant routine which moves a byte string of the designated length from the origin address to the destination address.

Entry Conditions

Calling Sequence

BL	S.EXEC54	
BL	S.EXEC54A	move from the task to the operating system (mapped out mode)
BL	S.EXEC54B	move from the operating system to the task (mapped out mode)

Registers

R1	origin (from) address
R2	destination (to) address
R7	negative number of bytes to be transferred

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3-R5	saved
R1,R2,R6,R7	destroyed

2.25 Subroutine S.EXEC58 - Link MRRQ to Message Receiver

This routine is called for the sending task to link an MRRQ entry to the message receiver queue of the destination task.

Entry Conditions

Calling Sequence

BL	S.EXEC58

Registers

R1	destination task DQE address
R2	MRRQ address
R3	scratchpad address (doubleword bounded)
R7	task activation sequence number of destination task

Exit Conditions

Return Sequence

TRSW	R0 (CC1 set indicates invalid destination task)

Registers

R3=R3-4W	scratchpad address
R1,R2,R4	saved
R5,R6,R7	destroyed

2.26 Subroutine S.EXEC60 - Validate PRB

This routine is called to validate the parameter receive block (PRB) of the destination task, when the destination task has made a request for the message or run request parameters. Validation is bypassed if the most recent pushdown on the TSA stack reflects a privileged caller. Otherwise, general PRB validation is performed.

Entry Conditions

Calling Sequence

BL	S.EXEC60

Registers

R2	PRB address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0 (CC1 set indicates validation error)

Registers

R1,R4,R5,R7	destroyed
R2	saved
R3=R3-4W	scratchpad address
R6	error code if CC1 set; otherwise, R6 is destroyed

2.27 Subroutine S.EXEC63 - Transfer Return Parameters

This routine is called for the destination task to transfer return parameters to the MRRQ after the destination task has issued an exit from message or run request processing.

Entry Conditions

Calling Sequence

BL	S.EXEC63

Registers

R1	MRRQ address
R2	RXB address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1,R2,R4,R5	saved
R3=R3-4W	scratchpad address
R6,R7	destroyed

2.28 Subroutine S.EXEC64 - No-Wait Message Postprocessing

This routine is invoked as a preemptive system service (end action priority 2) for the sending task. It in turn calls S.EXEC52 to accomplish postprocessing of the MRRQ. It will optionally vector to a user-specified end action routine, or call H.EXEC,34 to report no-wait message postprocessing complete.

Entry Conditions

Calling Sequence

Preemptive system service

Registers

R2	MRRQ address

Exit Conditions

Return Sequence

No return is made. S.EXEC64 exits to the user end action routine or to H.EXEC,34.

Registers

R1 PSB address on entry to user end action routine

2.29 Subroutine S.EXEC65 - No-Wait Mode Run Request Postprocessing

This routine is invoked as a preemptive system service (end action priority 2) for the sending task. It in turn calls S.EXEC52 to accomplish postprocessing of the MRRQ. It will optionally vector to a user-specified end action routine, or call H.EXEC,28 to report no-wait run request postprocessing complete.

Entry Conditions

Calling Sequence

Preemptive system service

Registers

R2 MRRQ address

Exit Conditions

Return Sequence

No return is made. S.EXEC65 exits to the user end action routine or to H.EXEC,28.

Registers

R1 PSB address on entry to user end action routine

2.30 Subroutine S.EXEC66 - Deallocate MRRQ

This routine is called to deallocate an MRRQ when processing associated with the MRRQ is complete. S.REMM22 is called to return the MRRQ space to memory pool.

Entry Conditions

Calling Sequence

BL	S.EXEC66

Registers

R2	MRRQ address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1,R2,R4,R6,R7	destroyed
R3=R3-4W	scratchpad address
R5	saved

2.31 Subroutine S.EXEC67 - Link Entry to End Action Queue

This routine is called for the destination task when destination task processing of a no-wait mode message or run request is complete. Its purpose is to link the MRRQ entry to the end action queue of the sending task. This causes the appropriate postprocessing routine to be invoked as a preemptive system service on behalf of the sending task. For more information, refer to the MPX-32 Reference Manual Volume I, Chapter 2.

Entry Conditions

Calling Sequence

```
BEI
BL      S.EXEC67
```

Registers

R1	DQE address of sending task
R2	MRRQ address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1,R2,R4,R6	saved
R3=R3-4W	scratchpad address
R5,R7	destroyed

2.32 Subroutine S.EXEC70 - Terminate Run Requests in Queue of Task

This routine is called when an error is encountered in attempting to reactivate a terminating task with additional run requests queued. S.EXEC70 in turn calls S.EXEC25 until the receiver queue is empty.

Entry Conditions

Calling Sequence

```
BL      S.EXEC70
```

Registers

R2	current task DQE address
R3	scratchpad address (doubleword bounded)

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1,R4-R7	destroyed
R2	saved
R3=R3-4w	scratchpad address

2.33 Subroutine S.EXEC71 - Start-up of Run Receiver Task

This routine is called to unlink the destination task from the RUNW or PREA list, unless R5 is not zero, and to link the destination task to the ready list at the priority specified in R6.

Entry Conditions

Calling Sequence

```
BEI
BL      S.EXEC71
```

Registers

R2	DQE address of destination task
R5	zero if task is to be unlinked from PREA list
R6	priority

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1,R3,R5-R7	destroyed
R2,R4	saved

2.34 Subroutine S.EXEC73 - Replace Context on TSA Stack

This routine is called to replace the most recent context on the TSA stack with the designated context block.

Entry Conditions

Calling Sequence

BL	S.EXEC73

Registers

R1	address of 10 word context block

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1,R3	saved
R2,R4-R7	destroyed

2.35 Subroutine S.EXEC74 - Reset Stack to User Level

This routine is called on abnormal task termination to reset the stack to the point of last user call.

Entry Conditions

Calling Sequence

BL	S.EXEC74

Registers

None

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1,R3,R5	destroyed
R2,R4,R6,R7	saved

2.36 Subroutine S.EXEC76 - Update Execution Accounting Value

This routine is called during task termination processing. The interval timer is read and the elapsed time added to T.ITAC in the TSA.

Entry Conditions

Calling Sequence

BL	S.EXEC76

Registers

R2	current task DQE address

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1,R2,R4,R6,R7	saved
R3	current task TSA address
R5	destroyed

2.37 Subroutine S.EXEC78 - Move Context from Stack to T.CONTXT

This routine is called when the context of the PSD, R0-R7, and PC are copied from the TSA of the current task to the debug context area. After filling the T.CONTXT area, it returns to the calling routine.

Entry Conditions

Calling Sequence

```
BL      S.EXEC78
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1	current pushdown address from T.REGP
R2,R4,R5	unchanged
R3	TSA address of current task
R6,R7	destroyed

File System Executive (H.FISE)

MPX-32 Technical Manual

Volume II

Contents

Page

Contents

1 H.FISE Overview

1.1 General Information

The File System Executive Module (H.FISE) performs the compatible mode file system services.

1.2 Entry Points

Entry Point	SVC Number	Description
H.FISE,1		reserved
H.FISE,2		reserved
H.FISE,3		N/A allocate temporary disk space
H.FISE,4		N/A deallocate temporary disk space
H.FISE,5		reserved
H.FISE,6		reserved
H.FISE,7		reserved
H.FISE,8	N/A	ASCII compression
H.FISE,9		reserved
H.FISE,10		reserved
H.FISE,11		reserved
H.FISE,12	75	create permanent file
H.FISE,13	76	change temporary file to permanent
H.FISE,14	77	delete permanent file or non-SYSGEN memory partition
H.FISE,15	N/A	permanent file log
H.FISE,16		reserved
H.FISE,17		reserved
H.FISE,18		reserved
H.FISE,19		reserved
H.FISE,20	N/A	RTM CALM create permanent file
H.FISE,21	N/A	RTM CALM change temporary file to permanent
H.FISE,22	21	set exclusive file lock
H.FISE,23	22	release exclusive file lock
H.FISE,24	23	set synchronization file lock
H.FISE,25	24	release synchronization file lock
H.FISE,26		reserved
H.FISE,27		reserved
H.FISE,28		reserved
H.FISE,29		reserved
H.FISE,30		reserved
H.FISE,31		reserved
H.FISE,32	N/A	wait for FLT entry space

N/A implies reserved for internal use by MPX-32.

Entry Point	SVC Number	Description
H.FISE,33		reserved
H.FISE,34		reserved
H.FISE,35		reserved
H.FISE,36		reserved
H.FISE,99	N/A	SYSGEN initialization

N/A implies reserved for internal use by MPX-32.

2 H.FISE Entry Points

2.1 Entry Point 1 - Reserved

2.2 Entry Point 2 - Reserved

2.3 Entry Point 3 - Allocate Temporary Disk Space

This entry point is used to effect the allocation of temporary disk file space. The space definition is computed and return is made to the caller. If the specified device does not contain available space of sufficient length to satisfy the request, a denial return is made to the caller with the space definition zeroed.

Entry Conditions

Calling Sequence

M.CALL H.FISE,3

Registers

R4	denial return address in case of an unrecoverable I/O error to the allocation map
R5	unit definition table (UDT) index (entry number) of the entry which defines the disk for which the space allocation is requested
R7	number of 192 word blocks requested for allocation

Exit Conditions

Return Sequence

M.RTRN 5,6,7

Registers

R5	UDT index of the disk where the space was allocated
R6,R7	space definition of the allocated file space or zero if the requested space was not available

(or)

Return Sequence

M.RTNA 4 R4 is the denial return address in case of an unrecoverable I/O error to the allocation map

Registers

None

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.RTRN
M.RTNA

System Services

H.VOMM,19 M

2.4 Entry Point 4 - Deallocate Temporary Disk Space

This entry point is called to release temporary disk file space, making it available for allocation.

Entry Conditions

Calling Sequence

M.CALL H.FISE,4

Registers

R4	denial return address in case of an unrecoverable I/O error to the allocation map
R5	UDT index of the disk on which the file resides
R6	starting disk address
R7	number of 192 word blocks to release

Exit Conditions

Return Sequence

M.RTRN

Registers

None

(or)

Return Sequence

M.RTNA 4 R4 is the denial return address in case of an unrecoverable I/O error to the allocation map

Registers

None

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.RTRN
M.RTNA

System Services

H.VOMM,20

2.5 Entry Point 5 - Reserved

2.6 Entry Point 6 - Reserved

2.7 Entry Point 7 - Reserved

2.8 Entry Point 8 - ASCII Compression

This entry point performs compression on an ASCII character string to yield its halfword equivalent.

Entry Conditions

Calling Sequence

M.CALL H.FISE,8

Registers

R6,R7 contain the 1- to 8-character ASCII string, left-justified, and blank-filled

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 contains the equivalent of the ASCII string in the right halfword

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.RTRN

2.9 Entry Point 9 - Reserved

2.10 Entry Point 10 - Reserved

2.11 Entry Point 11 - Reserved

2.12 Entry Point 12 - Create Permanent File

See M.CREATE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Services

H.VOMM,1
H.REX,20

System Subroutines

S.REXS8
S.REXS9

2.13 Entry Point 13 - Change Temporary File To Permanent

See M.PERM in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Services

H.VOMM,9
H.VOMM,11
H.VOMM,12
H.REXS,20

System Subroutines

S.REXS8
S.REXS9

2.14 Entry Point 14 - Delete File or Memory Partition

See M.DELETE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

Systems Services

H.VOMM,5
H.REXS,20

System Subroutines

S.REXS8
S.REXS9

2.15 Entry Point 15 - Permanent File Log

This entry point provides a log of currently existing permanent files and memory partitions. Information on the requested file is returned in the format of an eight word SMD entry.

See M.LOG in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Services

H.VOMM,10

System Subroutines

S.REXS8
S.REXS9

2.16 Entry Point 16 - Reserved

2.17 Entry Point 17 - Reserved

2.18 Entry Point 18 - Reserved

2.19 Entry Point 19 - Reserved

2.20 Entry Point 20 - RTM CALM Create Permanent File

This entry point allocates disk space for the specified permanent file and writes a corresponding entry into the directory. Optionally, the allocated space is zeroed.

Entry Conditions

Calling Sequence

```
CALM      X'75'
(or)
M.CALL    H.FISE,20
```

Registers

R1 bytes 1 and 2 contain the restricted file code, byte 3 contains the file type code

R2 file size

R3

Bits	Meaning if Set
0	system file
1	prezero
10	not a save device
11	fast file
14	read only
15	password only
16-23	device type code
24-31	optional device address (if present, bit 16 is also set)

R4,R5 password or R4 is zero

R6,R7 file name

Exit Conditions

Return Sequence

M.RTRN

Registers

None

(or)

Return Sequence

M.RTRN 6,7

Registers

R6

Value	Definition
1	file already exists
2	fast file collision mapping occurred
3	restricted access but no password supplied
4	disk space unavailable
5	specified device not configured or available
6	specified device is off-line
7	directory is full
8	specified device type is not configured
9	file name or password contain invalid characters

R7 zero

Abort Cases

FS01	unrecoverable I/O error to directory
FS02	unrecoverable I/O error to disk allocation map

Output Messages

None

External Reference

System Macro

M.RTRN
M.CALL

2.21 Entry Point 21 - RTM CALM

This entry point changes the status of a temporary file to permanent. The file must be an SLO or SBO file, and must be open, temporary, and allocated to the calling task.

Entry Conditions

Calling Sequence

CALM X'76'

(or)

M.CALL H.FISE,21

Registers

R1 byte 3 contains the file type code

R2 bytes 0, 1, and 2 contain the logical file code

R3

Bit	Meaning if Set
0	system file
1	prezero
10	not a save device
11	fast file
14	read only
15	password only

R4,R5 password or R4 is zero

R6,R7 file name

Exit Conditions

Return Sequence

M.RTRN

Registers

None

(or)

Return Sequence

M.RTRN 6,7

Registers

R6

Value	Description
1	file already exists
2	fast file collision mapping occurred
3	restricted access but no password supplied
4	file not open, temporary, SLO or SBO file
7	directory is full
9	file name or password contain invalid characters

R7 zero

Abort Cases

FS01 unrecoverable I/O error to directory
FS02 unrecoverable I/O error to disk allocation map

Output Messages

None

External Reference

System Macro

M.RTRN
M.CALL

2.22 Entry Point 22 - Set Exclusive File Lock

See M.FXLS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.23 Entry Point 23 - Release Exclusive File Lock

See M.FXLR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.24 Entry Point 24 - Set Synchronization File Lock

See M.FSLS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.25 Entry Point 25 - Release Synchronization File Lock

See M.FSLR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.26 Entry Point 26 - Reserved

2.27 Entry Point 27 - Reserved

2.28 Entry Point 28 - Reserved

2.29 Entry Point 29 - Reserved

2.30 Entry Point 30 - Reserved

2.31 Entry Point 31 - Reserved

2.32 Entry Point 32 - Wait for FLT Entry Space

This entry point is provided for system static and dynamic allocation services. It returns immediately if FLT entry space is available, otherwise the task is placed in a wait state until FLT space is available.

Entry Conditions

Calling Sequence

```
M.CALL      H.FISE,32
```

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.33 Entry Point 33 - Reserved

2.34 Entry Point 34 - Reserved

2.35 Entry Point 35 - Reserved

2.36 Entry Point 36 - Reserved

2.37 Entry Point 99 - SYSGEN Initialization

This entry point is for internal use only and is called during SYSGEN. H.FISE sets up its entry point table then returns to SYSGEN.

Input/Output Control System (H.IOCS)

MPX-32 Technical Manual

Volume II

Contents

Page

3 H.IOCS Subroutines

Page

1 H.IOCS Overview

1.1 General Information

The Input/Output Control System Module (H.IOCS) performs the device-independent portion of the I/O request management. This includes preprocessing and postprocessing of the I/O requests, as well as IOQ manipulation and management.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.IOCS,1	30	open file
H.IOCS,2	37	rewind file
H.IOCS,3	31	read record
H.IOCS,4	32	write record
H.IOCS,5	38	write end-of-file
H.IOCS,6	N/A	reserved
H.IOCS,7	33	advance record
H.IOCS,8	34	advance file
H.IOCS,9	35	backspace record
H.IOCS,10	25	execute channel program
H.IOCS,11	N/A	reserved
H.IOCS,12	3A	reserve channel
H.IOCS,13	3B	release channel
H.IOCS,14	N/A	reserved
H.IOCS,15	N/A	suspend user until I/O complete
H.IOCS,16	N/A	reserved
H.IOCS,17	N/A	get memory pool buffer
H.IOCS,18	N/A	reserved
H.IOCS,19	36	backspace file
H.IOCS,20	10	upspace
H.IOCS,21	3E	erase or punch trailer
H.IOCS,22	0D	eject/purge routine
H.IOCS,23	39	close file
H.IOCS,24	26	reserve dual-ported disk/set single-channel 8-line mode
H.IOCS,25	3C	wait I/O
H.IOCS,26	3D	system console wait
H.IOCS,27	27	release dual-ported disk/set dual-channel 8-line mode
H.IOCS,28	N/A	reserved
H.IOCS,29	N/A	handler opcode processing and I/O queue start interface
H.IOCS,30	N/A	adjust TCW format to bytes
H.IOCS,31	N/A	adjust TCW format to halfwords
H.IOCS,32	N/A	adjust TCW format to words
H.IOCS,33	N/A	request end-action task interrupt with no I/O request

N/A implies reserved for internal use by MPX-32.

Entry Point	SVC Number	Description
H.IOCS,34	2C	no-wait I/O end-action return
H.IOCS,35	N/A	reserved
H.IOCS,36	N/A	restart I/O
H.IOCS,37	N/A	virtual address validate
H.IOCS,38	N/A	kill all outstanding I/O
H.IOCS,39	N/A	discontiguous E-memory data address check
H.IOCS,40	N/A	discontiguous data address check for 24 bit address
H.IOCS,41	N/A	reserved
H.IOCS,42	N/A	reserved
H.IOCS,43	N/A	reserved
H.IOCS,44	N/A	SYSGEN initialization

N/A implies reserved for internal use by MPX-32.

1.3 Subroutine Summary

Subroutine	Description
S.IOCS0	no-wait I/O end-action entry
S.IOCS1	post I/O processing
S.IOCS2	reserved
S.IOCS3	unlink I/O queue
S.IOCS4	buffer to buffer move routine (halfword)
S.IOCS5	peripheral time-out
S.IOCS6	buffer to buffer move routine (byte)
S.IOCS7	buffer to buffer move routine (word)
S.IOCS8	buffer to buffer move routine (doubleword)
S.IOCS9	reserved
S.IOCS10	delete I/O queue and OS buffer
S.IOCS11	allocate memory pool
S.IOCS12	store IOCDs for extended I/O
S.IOCS13	allocate I/O queue and buffer space
S.IOCS14	abort code formatting
S.IOCS15	reserved
S.IOCS16	error handling
S.IOCS17	no-wait I/O exit
S.IOCS18	delete I/O queue and OS buffer
S.IOS19A, B, C, and D	move memory
S.IOCS20	get data address and transfer count
S.IOCS21	reserved
S.IOCS22	reserved
S.IOCS23	IOCS exit
S.IOCS24	reserved
S.IOCS25	reserved
S.IOCS26	close FPT and FAT if close request

Subroutine	Description
S.IOCS27	validate FCB and perform implicit open
S.IOCS28	initialize IOQ entry
S.IOCS29	report I/O complete
S.IOCS30	initialize FCB
S.IOCS31	mark units off-line
S.IOCS32	predevice access request validation
S.IOCS33	disk FAT processor
S.IOCS34	allocate variable IOQ entry
S.IOCS35	reserved
S.IOCS36	file segment processor
S.IOCS37	update FAT and ART for disk I/O
S.IOCS38	reserved
S.IOCS39	reserved
S.IOCS40	build IOCDs for XIO transfers

2 H.IOCS Entry Points

2.1 Entry Point 1 - Open File

See M.FILE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.2 Entry Point 2 - Rewind File

See M.CLSE or M_CLSE, and M.RWND or M_REWIND in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.3 Entry Point 3 - Read Record

See M.READ or M_READ in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.4 Entry Point 4 - Write Record

See M.WRIT or M_WRITE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.5 Entry Point 5 - Write End-of-File

See M.CLSE or M_CLSE, and M.WEOF or M_WRITEEOF in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.6 Entry Point 6 - Reserved

2.7 Entry Point 7 - Advance Record

See M.FWRD or M_ADVANCE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.8 Entry Point 8 - Advance File

See M.FWRD or M_ADVANCE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.9 Entry Point 9 - Backspace Record

See M.BACK or M_BACKSPACE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.10 Entry Point 10 - Execute Channel Program

See the Execute Channel Program system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.11 Entry Point 11 - Reserved

2.12 Entry Point 12 - Reserve Channel

See M.RSRV or M_RSRV in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.13 Entry Point 13 - Release Channel

See M.RRES or M_RRES in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.14 Entry Point 14 - Reserved

2.15 Entry Point 15 - Suspend User Until I/O Complete

This entry point checks the operation in progress bit in the FCB. If the bit is set, suspend calls the executive to wait until I/O completes. If the bit is reset, suspend performs a M.RTRN immediately.

Entry Conditions

Calling Sequence

```
M.CALL  H.IOCS,15
```

Registers

R1	FCB address (this address must be the address of the same FCB used to initiate the transfer on which the I/O suspend is being made)

Exit Conditions

Return Sequence

M.RTRN

Registers

None

Abort Cases

None

Output Messages

None

2.16 Entry Point 16 - Reserved

2.17 Entry Point 17 - Get Memory Pool Buffer

This entry point is used to obtain blocks of memory from the system memory pool. The maximum amount of memory to allocate is 192 words. All memory can be zeroed before returning to the calling task.

If memory is not available, the calling task is suspended by H.EXEC,6 until available. All memory returned has the attribute that its virtual address is the same as its absolute address.

Entry Conditions

Calling Sequence

M.CALL H.IOCS,17

Registers

R0	bit 0 is set if zeroing of memory is desired or bit 0 is reset if no zeroing is desired
R7	number of words to allocate

Exit Conditions

Return Sequence

M.RTRN R6,R7

Registers

R6	start virtual (same as absolute) address
R7	actual number of words in buffer (may be more than requested amount)

Abort Cases

None

Output Messages

None

2.18 Entry Point 18 - Reserved

2.19 Entry Point 19 - Backspace File

See M.BACK or M_BACKSPACE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.20 Entry Point 20 - Upspace

See M.UPSP or M_UPSP in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.21 Entry Point 21 - Erase or Punch Trailer

See the Erase or Punch Trailer system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.22 Entry Point 22 - Eject/Purge Routine

See the Eject/Purge Routine system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.23 Entry Point 23 - Close File

See M.CLSE or M _CLSE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.24 Entry Point 24 - Reserve Dual-Ported Disc/Set Single-Channel

See the M.RESP or M_RESP system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.25 Entry Point 25 - Wait I/O

See M.WAIT or M_WAIT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.26 Entry Point 26 - System Console Wait

See M.CWAT or M_CWAT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.27 Entry Point 27 - Release Dual-ported Disc/Set Dual-channel 8-line Mode

See the M.RELP or M_RELP system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.28 Entry Point 28 - Reserved

2.29 Entry Point 29 - Handler Opcode Processing and I/O Queue Interface

This entry point performs the following functions:

- calls opcode processing entry point of the device handler to perform predevice access processing
- places I/O request in a prioritized queue
- calls I/O queue start entry point of the device handler to start I/O if the queue is not currently driven
- branches to appropriate executive entry point to report type of I/O initiated. For wait I/O, branches to I/O postprocessing; for no-wait I/O, returns immediately to the user.

Entry Conditions

Calling Sequence

M.CALL	H.IOCS,29	used internally to H.IOCS by blocked I/O routines
(or)		
BU	H29.	used by internal H.IOCS routines to complete normal I/O processing

Registers

R1	FCB address

Exit Conditions

Return Sequence

See return sequence for S.IOCS1.

2.30 Entry Point 30 - Adjust TCW Format to Bytes

This entry point adjusts the transfer control word (TCW) format to bytes. The adjusted quantity is clamped so it does not exceed the maximum quantity specified.

Entry Conditions

Calling Sequence

M.CALL H.IOCS,30

Registers

R1	FCB address
R6	TCW
R7	maximum quantity

Exit Conditions

Return Sequence

M.RTRN 4,6

Registers

R4	adjusted quantity
R6	adjusted TCW

Abort Cases

IO24	boundary error

Output Messages

None

2.31 Entry Point 31 - Adjust TCW Format to Halfwords

This entry point adjusts the transfer control word (TCW) format to halfwords. The adjusted quantity is clamped so it does not exceed the maximum quantity specified.

Entry Conditions

Calling Sequence

M.CALL H.IOCS,31

Registers

R1	FCB address
R6	TCW
R7	maximum quantity

Exit Conditions

Return Sequence

M.RTRN 4,6

Registers

R4	adjusted quantity
R6	adjusted TCW

Abort Cases

IO24	boundary error

Output Messages

None

2.32 Entry Point 32 - Adjust TCW Format to Words

This entry point adjusts the transfer control word (TCW) format to words. The adjusted quantity is clamped so it does not exceed the maximum quantity specified.

Entry Conditions

Calling Sequence

M.CALL H.IOCS,32

Registers

R1	FCB address
R6	TCW
R7	maximum quantity

Exit Conditions

Return Sequence

M.RTRN 4,6

Registers

R4 adjusted quantity
R6 adjusted TCW

Abort Cases

IO24 boundary error

Output Messages

None

2.33 Entry Point 33 - Request End-action Task Interrupt with No I/O Request

This entry point is used by J.TSM to process open denials at its end-action routine by requesting an end-action task interrupt without performing an I/O request. A file control block (FCB) is not needed.

Entry Conditions

Calling Sequence

M.CALL H.IOCS,33

Registers

R7 end-action entry address

Exit Conditions

Return Sequence

M.RTRN

Registers

R1 passed unchanged to the end-action entry

Abort Cases

None

Output Messages

None

2.34 Entry Point 34 - No Wait I/O End-Action Return

See M.XIEA or M_XIEA in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.35 Entry Point 35 - Reserved

2.36 Entry Point 36 - Restart I/O

This entry point is used to restart I/O for devices where no-wait I/O incurred error or wait I/O retry aborted.

It is also used to restart I/O after an I/O channel is released back to the system.

Entry Conditions

Calling Sequence

M.CALL H.IOCS,36

Registers

R0 I/O queue address (from CDT.FIOQ)

Exit Conditions

Return Sequence

M.RTRN

Registers

None

Abort Cases

None

Output Messages

None

2.37 Entry Point 37 - Virtual Address Validate

This entry point verifies that a given virtual start address through an optional transfer length is within a user's legal limits of program execution.

Entry Conditions

Calling Sequence

M.CALL H.IOCS,37

Registers

R6 bits 0-7 are ignored; bits 8-31 are the virtual start address

R7 bits 0-15 are ignored; bits 16-31 are the transfer length in bytes

Exit Conditions

Return Sequence

M.RTRN 6

Registers

R6 virtual start address (same as on entry), or zero (transfer outside legal limits)

Abort Cases

None

Output Messages

None

2.38 Entry Point 38 - Kill All Outstanding I/O

This entry point is used to terminate all outstanding I/O for the current executing task.

Peripheral timeout is forced for pending I/O whereas queued I/O is removed from the CDT or UDT string. Appropriate status is set to indicate either device timeout or "I/O killed" respectively.

This entry point also kills outstanding I/O issued by J.TSM for a task issuing a command line recall and edit read. The current IOQ associated with task's DCA is checked for J.TSM's program number. If found, the I/O is halted and normal kill processing continues.

Entry Conditions

Calling Sequence

M.CALL H.IOCS,38

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

Abort Cases

None

Output Messages

None

2.39 Entry Point 39 - Discontiguous E-Memory Data Address Check

This entry point ensures that a given virtual data transfer is within E-memory and does not cross noncontiguous memory blocks based on an optional transfer length.

Entry Conditions

Calling Sequence

M.CALL H.IOCS,39

Registers

R6	bits 0-11 are ignored; bits 12-31 are the virtual start address
R7	bits 0-15 are ignored; bits 16-31 are the transfer length in bytes

Exit Conditions

Return Sequence

M.RTRN zero

Registers

R0	Value	Definition
	0	transfer address out of E-memory, or transfer crosses noncontiguous memory blocks
	not 0	transfer address is within E-memory:
		1 transfer address is in operating system portion of E-memory (virtual = absolute)
		2 transfer address is not in operating system portion of E-memory (virtual = absolute)

Abort Cases

None

Output Messages

None

2.40 Entry Point 40 - Discontiguous Data Address Check for 24 Bit Address

This entry point ensures that a given virtual data transfer is within E-memory and does not cross noncontiguous memory blocks based on an optional transfer length.

Entry Conditions

Calling Sequence

M.CALL H.IOCS,40

Exit Conditions

Return Sequence

M.RTRN zero

Registers

R0	Value	Definition
	0	transfer address out of E-memory, or transfer crosses noncontiguous memory blocks
	not 0	transfer address is within E-memory:
	1	transfer address is in operating system portion of E-memory (virtual = absolute)
	2	transfer address is not in operating system portion of E-memory (virtual = absolute)
R6	bits 0-7 are ignored; bits 8-31 are the virtual start address	
R7	bits 0-15 are ignored; bits 16-31 are the transfer length in bytes	

Abort Cases

None

Output Messages

None

2.41 Entry Point 41 - Reserved

2.42 Entry Point 42 - Reserved

2.43 Entry Point 43 - Reserved

2.44 Entry Point 44 - SYSGEN Initialization

Performs any required H.IOCS initialization at SYSGEN time.

Entry Conditions

Calling Sequence

M.EIR

(or)

Branch to H.IOCS.I

Registers

None

Exit Conditions

Return Sequence

M.XIR H.IOCS. (special SYSGEN initialization termination macro)

Registers

Same as on entry

Abort Cases

None

Output Messages

None

3 H.IOCS Subroutines

3.1 Subroutine S.IOCS0 - No-Wait I/O End-Action Entry

This routine re-establishes file control block linkages to the IOQ file assignment table and restores the user's call frame in the task service area to its context before device access.

Entry Conditions

Calling Sequence

LPSD IOQ.PSD from H.EXEC

Registers

R3 IOQ address

Exit Conditions

Return Sequence

BU S.IOCS1

Registers

R3 IOQ address

3.2 Subroutine S.IOCS1 - Post I/O Processing

This routine is entered by a branch and link directly after a wait I/O request completes or indirectly as a task interrupt service when a no-wait I/O request completes.

If wait I/O completed with errors, applicable error messages are output and I/O retry attempted unless error processing is inhibited. If error processing is not applicable, an abort message is output or the error return address is taken. Wait I/O would take the error return in FCB word 6.

For the no-wait I/O requests or wait I/O requests, appropriate data conversions and buffer transfers from system buffers to user buffers are performed. Any system buffer which had been allocated and the I/O queue itself are deallocated.

If no-wait I/O completed with errors, the no-wait error end-action address in word 14 of the FCB is honored if present. The user must return with H.IOCS,34 to exit the error end-action service. If no error end-action address is present, status information is posted in the FCB and a return is made to the user. If an abort of this task had been issued, the end-action routine is not called and a status bit is set to indicate this.

For wait I/O with errors for which retry is applicable yet the operator aborted, all system buffers and the I/O queue are deallocated and the I/O queue is unlinked.

For no-wait I/O that completes without error, the no-wait normal end-action address in word 13 of the FCB is honored if present. The user must return via H.IOCS,34 to exit the normal end action service. If an abort had been issued for this task, the end-action routine is not called and a status bit is set to indicate this.

Wait I/O that completes without error always returns to the next instruction following the original point of call.

Entry Conditions

Calling Sequence

BL	S.IOCS1 wait I/O
(or)	
LPSD	(I/O queue + 4W) no-wait I/O

Registers

R3	I/O queue address

Exit Conditions

Return Sequence

M.RTRN	normal wait I/O completion or error processing inhibited	
(or)		
M.RTNA R6	error return, wait I/O; address from FCB word 6	
(or)		
X1	FCB address	
BL	ERROR-END-ACTION	no-wait, complete with error address from FCB.NWER Note: User exits through H.IOCS,34.

(or)

X1	FCB address	
BL	NORMAL-END-ACTION	no-wait, complete without error address for FCB.NWOK **Note:** User exits through H.IOCS,34.
M.CALL	H.EXEC,12	normal no-wait I/O completion, or error processing inhibited, or no end-action addresses specified **Note:** Post I/O processing is re-initiated when I/O completes.

Wait I/O Retry

(or)

M.CALL	H.EXEC,1	wait I/O interactive input
	H.EXEC,2	wait I/O terminal output
	H.EXEC,3	wait I/O not interactive input or terminal output **Note:** Post I/O processing will be reinitiated when I/O completes.

Registers

None unless noted.

Abort Cases

IO06	INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS DESTROYED, (3) TRANSFER ERROR DURING FILE INPUT.
IO21	IOCS HAS ENCOUNTERED AN UNRECOVERABLE I/O ERROR IN ATTEMPTING TO PROCESS AN I/O REQUEST ON BEHALF OF A TASK.

Output Messages

**dtchsa* INOP: R,A?

dt	device type mnemonic (for example, LP)
ch	channel number
sa	subaddress

I/O ERR DEVICE: *dtchsa* STATUS (XXCCDDDD)=*zzzzzzzz* LFC *kkk date time*

dt	device type mnemonic
ch	channel number
sa	subaddress
zzzzzzzz	actual status returned
kkk	logical file code associated with I/O
date	date stamp in format *mm/dd/yy*
time	time stamp in format *hh:mm:ss*

For XIO devices, a second line is also displayed:

XIO SENSE STATUS = *senseword*

senseword sense status information returned in FCB.IST1 of an extended FCB. Refer to the appropriate hardware technical manual for a description.

3.3 Subroutine S.IOCS2 - Reserved

3.4 Subroutine S.IOCS3 - Unlink I/O Queue

This routine unlinks the just completed I/O queue entry from the CDT or UDT active I/O queue string. This routine must be externally gated.

Entry Conditions

Calling Sequence

BL S.IOCS3

Registers

R2 I/O queue address

R3 CDT address

Exit Conditions

Return Sequence

TRSW R7

Registers

R1	CDT address
R2,R3,R5	unchanged
R4,R6,R7	destroyed

Abort Cases

None

Output Messages

None

3.5 Subroutine S.IOCS4 - Buffer to Buffer Move Routine (Halfword)

This routine moves the contents of one buffer to another buffer, one halfword at a time. See S.IOS19A, B, C, or D which can perform the same function.

Entry Conditions

Calling Sequence

BL S.IOCS4

Registers

R1	from buffer halfword address
R2	to buffer halfword address
R4	negative number of bytes to convert

Exit Conditions

Return Sequence

TRSW R0

Registers

R1,R2,R4,R6 destroyed

Abort Cases

None

Output Messages

None

3.6 Subroutine S.IOCS5 - Peripheral Time-out

This routine performs peripheral time-out checking for all devices with I/O outstanding. This routine is entered every timer unit, and will branch to the device handler lost interrupt entry point for processing if the time limit is exceeded.

Entry Conditions

Calling Sequence

```
BL      S.IOCS5
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1-R7 destroyed

Abort Cases

None

Output Messages

None

3.7 Subroutine S.IOCS6 - Buffer to Buffer Move Routine (Byte)

This routine moves the contents of one buffer to another buffer one byte at a time. See S.IOS19A, B, C, or D which can perform the same function.

Entry Conditions

Calling Sequence

```
BL      S.IOCS6
```

Registers

R1	from buffer byte address
R2	to buffer byte address
R4	negative number of bytes to move

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1,R2,R4,R6 destroyed

Abort Cases

None

Output Messages

None

3.8 Subroutine S.IOCS7 - Buffer to Buffer Move Routine (Word)

This routine moves the contents of one buffer to another buffer one word at a time. See S.IOS19A, B, C, or D which can perform the same function.

Entry Conditions

Calling Sequence

```
BL      S.IOCS7
```

Registers

R1	from buffer word address
R2	to buffer word address
R4	negative number of words to move

Exit Conditions

Return Sequence

TRSW R0

Registers

R1,R2,R4,R6 destroyed

Abort Cases

None

Output Messages

None

3.9 Subroutine S.IOCS8 - Buffer to Buffer Move Routine (Doubleword)

This routine moves the contents of one buffer to another buffer, one doubleword at a time. See S.IOS19A, B, C or D which can perform the same function.

Entry Conditions

Calling Sequence

BL S.IOCS8

Registers

R1	from buffer doubleword address
R2	to buffer doubleword address
R4	negative number of doublewords to move

Exit Conditions

Return Sequence

TRSW R0

Registers

R1,R2,R4,R6,R7 destroyed

Abort Cases

None

Output Messages

None

3.10 Subroutine S.IOCS9 - Reserved

3.11 Subroutine S.IOCS10 - Delete I/O Queue and OS Buffer

This routine deallocates the I/O queue and any system memory pool areas used during the I/O operation.

Entry Conditions

Calling Sequence

```
BL      S.IOCS10
```

Registers

R3 I/O queue address

Exit Conditions

Return Sequence

```
TRSW    R6
```

Registers

R1 FCB address
R2-R7 destroyed

Abort Cases

None

Output Messages

None

3.12 Subroutine S.IOCS11 - Allocate Memory Pool

This entry point is used to obtain memory from the system memory pool. The maximum amount of memory which can be allocated is 192 words. An abort condition occurs if more than 192 words are requested. All memory can be zeroed before returning to the calling task.

If memory is not available, the calling task will be suspended by H.EXEC,6 until memory is available. All memory returned has the attribute that its virtual address is the same as its absolute address.

Entry Conditions

Calling Sequence

BL S.IOCS11

Registers

R7 number of words to allocate

Exit Conditions

Return Sequence

TRSW R4

Registers

R1,R3 reserved
R2,R4,R5 destroyed
R6 address of memory pool
R7 actual number of words allocated

Abort Cases

None

Output Messages

None

3.13 Subroutine S.IOCS12 - Store IOCDs for Extended I/O

This routine dynamically stores IOCDs into the I/O queue as required during extended I/O request processing. An abort condition occurs if there is not sufficient space for the IOCDs.

Entry Conditions

Calling Sequence

BL S.IOCS12

Registers

R3 I/O queue address
R6 IOCD most significant word
R7 IOCD least significant word

Exit Conditions

Return Sequence

TRSW R0

Registers

R3	I/O queue address
R5	last dynamic IOCD location
R6	IOCD most significant word
R7	IOCD least significant word

Abort Cases

None

Output Messages

None

3.14 Subroutine S.IOCS13 - Allocate I/O Queue and Buffer Space

This routine must be called by the opcode processing entry point of device handlers processing opcodes for which standard I/O queue entries are required, for example, operations that result in a device access.

Entry Conditions

Calling Sequence

BL S.IOCS13

Registers

R1 FCB address

Exit Conditions

Return Sequence

TRSW R0

Registers

R2 destroyed

Abort Cases

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.

Output Messages

None

3.15 Subroutine S.IOCS14 - Abort Code Formatting

This routine is used to determine which operation caused an abort and on which logical file code (LFC) the abort occurred. A valid abort code must be specified in register five upon entry.

Entry Conditions

Calling Sequence

BU S.IOCS14

Registers

R1 FCB address
R5 valid 4-character abort code

Exit Conditions

Return Sequence

BU S.IOCS16

Registers

R1	FCB address
R4	destroyed
R5	abort code
R6	4-character (ASCII) operation causing the abort
R7	1- to 3-character LFC where the abort occurred

Abort Cases

None

Output Messages

None

3.16 Subroutine S.IOCS15 - Reserved

3.17 Subroutine S.IOCS16 - Error Handling

This routine processes user error exit parameters in the FCB. For I/O requests which are not aborted, the 12-character abort message is saved in words 11, 12 and 13 of the current call frame at the time of the abort. A task will not abort if any one of the following conditions exist:

- task is a no-wait I/O request
- error processing inhibit bit is set in the FCB
- valid error exit address is specified

Entry Conditions

Calling Sequence

BU S.IOCS16

Registers

R1	FCB address
R2-R4	destroyed
R5	4-character abort code
R6,R7	extended abort code

Exit Conditions

Return Sequence

BU	S.IOCS17	no-wait I/O request
M.RTRN		error processing inhibit bit set
M.RTNA	R6	error exit address is valid
M.CALL	H.REXS,28	task is aborted

Registers

R1 FCB address (no-wait I/O requests only)

3.18 Subroutine S.IOCS17 - No-Wait I/O Exit

This routine processes no-wait I/O requests with an end-action address specified in the FCB. The request is queued to DQE.TISF.

Entry Conditions

Calling Sequence

BU S.IOCS17

Registers

R1 FCB address

Exit Conditions

Return Sequence

M.RTRN

Registers

None

3.19 Subroutine S.IOCS18 - Delete I/O Queue and OS Buffer

This routine deallocates the I/O queue and any system memory pool buffer areas used during I/O. It is called from the handlers as a result of an OPCOM KILL request. This routine is called with the CPU running unmapped.

Entry Conditions

Calling Sequence

BL S.IOCS18

Registers

R3 I/O queue address

Exit Conditions

Return Sequence

TRSW R6

Registers

R1	I/O queue address
R2-R7	destroyed

Abort Cases

None

Output Messages

None

3.20 Subroutines S.IOS19A, B, C, and D - Move Memory

These routines move memory from one memory location to a different memory location. The move is performed using the largest transfer unit possible (such as doublewords, words, halfwords, or bytes). The transfer quantity used depends on the alignment of the address locations that memory is moving from and to. The choice of which routine to use is dependent on whether the source and target address locations reside in the task or operating system address space outlined as follows:

Entry Conditions

Calling Sequence

			Source	Target
BL	S.IOS19A	move memory	from task	to task
BL	S.IOS19B	move memory	from task	to OS
BL	S.IOS19C	move memory	from OS	to task
BL	S.IOS19D	move memory	from OS	to OS

Registers

R1	address memory is being moved from
R2	address memory is being moved to
R5	number of bytes of memory to be moved

Exit Conditions

Return Sequence

TRSW R0

Registers

R1	contains the next available memory address after the from address, which is the sum of the contents of R1 and R5
R2	contains the next available memory address after the to address, which is the sum of the contents of R1 and R5
R3	unchanged
R4-R7	destroyed

Abort Cases

None

Output Messages

None

Scratchpad Usage

The last two words of the current stack level are used.

Note: For compatibility with nonextended and extended mode images, the calling sequence which was used before version 3.5 can still be used: BL S.IOCS19

3.21 Subroutine S.IOCS20 - Get Data Address and Transfer Count

This routine extracts the user's data address and transfer count from an 8 or 16 word FCB. The extracted transfer count is always in bytes and the extracted data address is always a pure address, with no F and C bits. The transfer count is clamped to the maximum value for the device or transfer type.

Entry Conditions

Calling Sequence

BL S.IOCS20

Registers

R1 FCB address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1	FCB address
R2,R5	unchanged
R3,R4	destroyed
R6	data address
R7	transfer count

3.22 Subroutine S.IOCS21 - Reserved

3.23 Subroutine S.IOCS22 - Reserved

3.24 Subroutine S.IOCS23 - IOCS Exit

This routine exits IOCS by performing a return to the caller for wait I/O requests. For no-wait I/O requests, IOCS is exited by calling S.IOCS17. It calls to S.IOCS26 to close the FPTs and FATs before returning to the caller by M.RTRN.

Entry Conditions

Calling Sequence

BU S.IOCS23

Registers

R1 FCB address

Exit Conditions

Return Sequence

M.RTRN	wait I/O requests
BU	no-wait I/O requests

Registers

R2-R7	destroyed

Abort Cases

IO41	BLOCKING ERROR DURING NON-DEVICE ACCESS
IO44	NON-DEVICE ACCESS I/O ERROR. THIS ERROR MAY BE THE RESULT OF CHANNEL/CONTROLLER INITIALIZATION FAILURE.

3.25 Subroutine S.IOCS24 - Reserved

3.26 Subroutine S.IOCS25 - Reserved

3.27 Subroutine S.IOCS26 - Close FPT and FAT if Close Request

This routine is called by IOCS to complete close processing of a task. It causes the FPT and FAT entries associated with the task's FCB to be closed.

Entry Conditions

Calling Sequence

BL	S.IOCS26

Registers

R1	FCB address

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1	FCB address
R2-R7	destroyed

3.28 Subroutine S.IOCS27 - Validate FCB and Perform Implicit Open

This routine checks to ensure there is a FCB.LFC and FPT.LFC match and that the FCB is linked to the FPT. If I/O is outstanding, it is suspended. If the FCB is not open, an implicit open is performed.

Entry Conditions

Calling Sequence

```
BL      S.IOCS27
```

Registers

R1 FCB address

Spad Cell Used

2

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1	FCB address
R2,R6,R7	destroyed
R3	FPT address
R4	24-bit mask
R5	unchanged

3.29 Subroutine S.IOCS28 - Initialize IOQ Entry

This routine initializes the IOQ parameters from the FCB, CDT, UDT, and FAT. It also stores the program number into the IOQ and sets FCB.IOQA.

Entry Conditions

Calling Sequence

```
BL      S.IOCS28
```

Registers

R1	FCB address (or TCPB address)
R2	FAT address (or zero)
R3	CDT address
R6	IOQ address
R7	number of extra words in this IOQ

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1-R3,R6	same as on entry (masked with X'FFFFF')
R4,R7	destroyed
R5	unchanged

3.30 Subroutine S.IOCS29 - Report I/O Complete

This routine is called by handlers to report I/O completion. This routine is called with the CPU running unmapped.

Note: IOQ.RTN can be used to save the return address before calling S.EXEC1, S.EXEC2, S.EXEC3, or S.EXEC4.

Entry Conditions

Calling Sequence

```
BL      S.IOCS29
```

Registers

R1	program number
R2	IOQ address

Exit Conditions

Return Sequence

TRSW R0

Registers

R2,R6 IOQ address
R1,R3-R5,R7 destroyed

3.31 Subroutine S.IOCS30 - Initialize FCB

This routine clears status in the user's FCB, establishes linkage from the FCB to the FPT and inserts the specified operation code.

Entry Conditions

Calling Sequence

BL S.IOCS30

Registers

R1 FCB address
R3 FPT address
R4 X'00FFFFFF' (24 bit mask value)
R5 hexadecimal opcode, X'0' to X'F'

Exit Conditions

Return Sequence

TRSW R0

Registers

R1 FCB address
R2 FAT address
R3 FPT address
R4,R7 unchanged
R5 opcode
R6 destroyed

3.32 Subroutine S.IOCS31 - Mark Units Off-line

This routine marks a controller, and all the units connected to the controller off-line.

Entry Conditions

Calling Sequence

BL S.IOCS31

Registers

R1 CDT address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1 CDT address
R3,R7 destroyed

3.33 Subroutine S.IOCS32 - Predevice Access Request Validation

This routine uses the file control block (FCB) opcode and an internal table of valid operations to check for the following violations. If any violation is found, the abort code and reason for the abort is issued:

```
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.
IO08    DEVICE ASSIGNMENT IS REQUIRED FOR AN UNPRIVILEGED
        TASK TO USE THE SERVICE.
IO09    ILLEGAL OPERATION ON THE SYC FILE.
IO28    ILLEGAL OPERATION ATTEMPTED ON AN OUTPUT ACTIVE
        FILE OR DEVICE.
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.
```

If the transfer count is zero, the transfer is considered complete and control is passed to S.IOCS23. If the opcode is read, end-of-file (EOF) status is returned.

Entry Conditions

Calling Sequence

```
BL      S.IOCS32
```

Registers

R1 FCB address

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1 FCB address
R2-R7 destroyed

3.34 Subroutine S.IOCS33 - Disc FAT Processor

This routine checks for EOF, EOM, and BOM on disk files. It is called by the disk handlers at opcode processing time to check for a valid block number. It will also extend the file for a write past EOM, if allowed. This routine enqueues a reader trying to read past EOF on an implicit shared file if there is a writer on the file.

Entry Conditions

Calling Sequence

```
BL      S.IOCS33
```

Registers

R1 FCB address
R2 FAT address
R7 blocks spanned in operation (+ if forward, - if backward)

Exit Conditions

Return Sequence

TRSW R0

Registers

R3,R4 destroyed

R5 starting relative disk position

R6 zero if operation within file bounds; -1 if operation caused an EOF, EOM, or BOM; +1 if operation is to be truncated

If R6 = +1

R7 actual number of blocks to transfer; otherwise this register is undefined

3.35 Subroutine S.IOCS34 - Allocate Variable IOQ Entry

This routine is called by the opcode processing entry point of device handlers processing opcodes for which I/O queue entries are required, for example, opcodes resulting in a device access. It allows the handler to specify the amount of additional space it wants added to the end of the IOQ entry for creation of the actual IOCL chain. The IOQ may be extended by an additional three words if blocked I/O is being done.

For F-class devices, this routine allocates and initializes an IOQ entry.

For D-class devices, this routine allocates and initializes an IOQ entry, allocates an operating system I/O buffer if necessary, and builds a transfer control word (TCW) if necessary.

Entry Conditions

Calling Sequence

BL S.IOCS34

Registers

R1 FCB address

R7 number of words to extend the IOQ

Note: Enters S.IOCS13 for completion of IOQ building.

Spad Cells Used

12, 13, 15-22

Exit Conditions

Return Sequence

TRSW R0

Registers

R1	FCB address
R2	destroyed
R3-R7	unchanged

3.36 Subroutine S.IOCS35 - Reserved

3.37 Subroutine S.IOCS36 - File Segment Processor

This routine searches the segment descriptor list to find the real binary block number that corresponds to the relative block number that is passed to this routine. This routine also returns the number of blocks to the end of the current segment that contains the starting relative block number.

Entry Conditions

Calling Sequence

BL S.IOCS36

Registers

R1	FCB address
R2	FAT address
R5	relative block number

Exit Conditions

Return Sequence

TRSW R0

Registers

R3	destroyed
R6	real binary disk address
R7	blocks to the end of this segment

3.38 Subroutine S.IOCS37 - Update FAT and ART for Disc I/O

This routine updates the FAT and ART for implicit shared files at postprocessing time for transfers to a disk file. This routine updates the current disk position to the next block to be transferred and updates the EOF block number for an output operation.

Entry Conditions

Calling Sequence

BL S.IOCS37

Registers

R1 FCB address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1 FCB address
R2,R4-R7 destroyed

3.39 Subroutine S.IOCS38 - Reserved

3.40 Subroutine S.IOCS39 - Reserved

3.41 Subroutine S.IOCS40 - Build IOCDs for XIO Transfers

This routine breaks down input/output control doublewords (IOCDs) into two or more transfers and sets the data chaining bit in IOCD if discontiguous. The IOCDs are stored in the IOCD buffer within the I/O queue and the IOCD buffer address is incremented as required.

Entry Conditions

Calling Sequence

```
BL      S.IOCS40
```

Registers

R3	IOQ address
R6	IOCD word 1 with command only (bits 8-31=0)
R7	IOCD word 2 with flags only (bits 8-31=0)

Input

IOQ.FCT2	logical data address
IOQ.FCT3	transfer count
IOQ.IST1	address to store the IOCD that is built

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R2,R4	destroyed

Output

IOQ.FCT2	address of end of data buffer
IOQ.FCT3	destroyed
IOQ.IST1	address to store next IOCD built

Interrupt and Trap Processors (H.IP?? and H.SVC?)

MPX-32 Technical Manual

Volume II

Contents

1 H.IP?? and H.SVC? Overview

1.1 General Description

The Interrupt and Trap Processing Module (H.IP?? and H.SVC?) defines the routines which activate interrupts and trap processing. These routines list the indication of failure or interrupt and the action taken.

1.2 Interrupt and Trap Processing (H.IP?? and H.SVC?)

Interrupt Processor	Description
H.IP13	console attention interrupt processor
H.IP0A	call monitor interrupt processor
H.IPCL	real-time clock interrupt processor

1.2.1 Trap Processors

Trap Processor	Description
H.IP00	power fail trap and base mode task dispatch
H.IPAS	autostart
H.IP02	memory parity
H.IP03	nonpresent memory
H.IP04	undefined instruction
H.IP05	privilege violation
H.IP06	SVC trap processor
H.SVC0, H.SVC6	M.CALL SVC
H.SVC1, H.SVC2	supervisor call
H.SVC3	M.OPEN SVC
H.SVC4	M.RTRN/M.RTNA SVC
H.SVCN	invalid SVC type
H.IP07	machine check
H.IP08	system check
H.IP09	map fault
H.IP0C	address specification
H.IPHT	CPU halt
H.IP0F	arithmetic exception
H.IP10	cache memory parity error

2 H.IP?? and H.SVC? Interrupt Processors

2.1 Console Attention Interrupt Processor (H.IP13)

The console interrupt processor is directly connected to the console interrupt and is activated when a console interrupt occurs. This processor recognizes the console interrupt, and issues a break request for the associated task.

Entry Conditions

Calling Sequence

The processor is entered directly when a console interrupt occurs (priority level 13). All registers are saved by this processor.

Exit Conditions

Return Sequence

All registers are restored to their content at the time of the interrupt. Exit is made to the system with the standard interrupt exit sequence (BL S.EXEC5).

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.EQUS

2.2 Call Monitor (CALM) Interrupt Processor (H.IP0A)

This trap processor is entered when a CALM instruction is executed for emulation of a RTM service. The processor determines the module and entry point number. A register push-down and PSD save is performed and control is transferred to the specified module.

Entry Conditions

Calling Sequence

Occurrence of an interrupt signal at priority level X'27'

Exit Conditions

Return Sequence

LPSD IP27.T1 IP27.T1 = address of MOD., E.P

Registers

Same as upon detection of interrupt

Abort Cases

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

Output Messages

None

External Reference

System Macro

M.EQUS

CALM.TBL

2.3 Real-Time Clock Interrupt Processor (H.IPCL)

The Real-Time Clock Interrupt Processor is directly connected to the specified real-time clock interrupt and therefore is activated upon its occurrence. This processor performs two primary functions:

- maintains the clock interrupt counter (C.INTC) used for time of day calculations
- processes any active DQE timers.

Entry Conditions

Calling Sequence

The processor is entered directly upon the occurrence of the real-time clock interrupt to which it is connected

Exit Conditions

Return Sequence

```
BL      S.EXEC5
```

Registers

R2	address of register save area
R6,R7	PSD

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.EQUS

3 H.IP?? and H.SVC? Trap Processors

3.1 Power Fail Trap and Base Mode Task Dispatch Processor (H.IP00)

Power fail trap processing performed at this level is installation and application dependent; therefore, a minimal routine is provided for this level, which may be easily overridden by a user-supplied routine. The routine saves the general purpose registers (R0-R7) and the base mode registers (B0-B7). The CPU scratchpad is written to low memory to provide the required parameters for a power up-auto-restart.

Entry Conditions

Calling Sequence

Occurrence of interrupt or trap signal at priority level X'00'

Exit Conditions

Return Sequence

None

Abort Cases

None

Output Messages

None

Base mode tasks are dispatched by a section of code in H.IP00. Within H.IP00's code, the base mode instructions are contained in DATAW statements. H.IP00 is assembled by the Macro Assembler and the base mode instructions would not be understood, otherwise.

At the end of the base mode task activation process, the task activation module (H.TAMM) loads an SVC 1,X'55' (M.EXIT) instruction into R7, then branches to BR.DISP in H.IP00. The code at BR.DISP switches H.IP00 into base mode then enters the task by executing a CALL instruction. If the task terminates with a RETURN instruction, control returns to H.IP00, the instruction contained in R7 is executed, and the task terminates.

3.2 Autostart Trap Processor - (H.IPAS)

Processing at this level is application and installation dependent; therefore, a minimal routine is provided which may be easily overridden by a user-supplied routine.

This trap occurs during the power up sequence, provided the following operating conditions are met:

1. The CPU, IPU, and system software traps are enabled.
2. The CPU scratchpad image is contained in dedicated memory locations X'300' through X'6FC'.
3. The memory scratchpad image contains the CPU and IPU scratchpad keys.
4. A successful power down trap has been executed.
5. The integrity of the memory has been preserved.

Note: The system memory configuration must be core and/or MOS memory with a battery backup.

Because the CPU scratchpad key is zeroed when H.IPAS is initialized, condition three is not met. Therefore, H.IPAS is not entered and either an automatic IPL or trap halt is executed.

The H.IPAS trap handler may be replaced with a user-supplied, power-up, auto-restart routine as follows:

1. Specify the new routine in the SYSGEN SYSTRAP directive.
2. Modify the trap vector table H.IPAS entry using the system macro M.TRPINT. Priority level X'01' must be specified as the user-supplied auto-restart trap priority level.
3. Meet the five operating conditions.

Entry Conditions

Calling Sequence

Occurrence of interrupt or trap signal at priority level X'01'

Exit Conditions

Return Sequence

HALT

Abort Cases

None

Output Messages

None

3.3 Memory Parity Trap Processor (H.IP02)

A memory parity error is an indication of total hardware failure and is treated as a fatal system crash. Registers are loaded (for display) with the saved PSD and the instruction resulting in the trap, and the M.KILL macro is invoked. Processing can be continued at the point of interrupt with the registers and PSD intact, but I/O in progress when the HALT was executed is terminated.

Entry Conditions

Calling Sequence

Occurrence of interrupt signal at priority level X'12', or trap signal at level X'02'

Note: Entry into this routine results in registers being loaded for console display as follows:

R0,R1	PSD saved by the hardware when the trap occurred
R2	physical address of the instruction causing the trap
R3	instruction being executed when trap occurred
R4	CPU status word stored when the trap occurred
R5	crash code
R6	address of register save block
R7	ASCII 'TRAP'

Exit Conditions

Return Sequence

M.KILL (crash code = MP01)

Abort Case

MP01 MEMORY ERROR OCCURRED IN A TASK'S LOGICAL ADDRESS SPACE. THIS IS AN INTERNAL OR CPU FAILURE. RERUN TASK.

Output Messages

None

External Reference

System Macro

M.KILL

3.4 Nonpresent Memory Trap Processor (H.IP03)

This routine results in the task currently in execution being aborted. A register push-down and PSD save is made, the abort request is reported to the scheduler, and a standard exit is performed.

Entry Conditions

Calling Sequence

Occurrence of interrupt signal at priority level X'24', or trap signal at level X'03'

Exit Conditions

Return Sequence

```
BL      S.EXEC5A
```

Registers

R2	register save area address
R5	abort code
R6,R7	PSD

Abort Cases

```
NM01    A NONPRESENT MEMORY TRAP ERROR CONDITION HAS OCCURRED.
```

Output Messages

None

3.5 Undefined Instruction Trap Processor (H.IP04)

This routine results in an abort of the task currently in execution.

Entry Conditions

Calling Sequence

Occurrence of an interrupt signal at priority level X'25', or trap signal at level X'04'

Exit Conditions

Return Sequence

BU S.EXEC5A

Registers

R2 register save area address

R5 abort code

R6,R7 PSD

Abort Cases

UI01 UNDEFINED INSTRUCTION TRAP

Output Messages

None

3.6 Privilege Violation Trap Processor (H.IP05)

This routine results in an immediate abort of the task currently in execution.

Entry Conditions

Calling Sequence

Occurrence of an interrupt signal at priority level X'26', or a trap signal at level X'05'

Exit Conditions

Return Sequence

BL S.EXEC5A

R2 register save area address

R5 abort code

R6,R7 PSD

Abort Cases

PV01 PRIVILEGE VIOLATION TRAP.

Output Messages

None

3.7 SVC Trap Processor (H.IP06)

This processor provides the SVC secondary vector table and selects the appropriate processing routine based on SVC type.

Entry Conditions

Calling Sequence

Occurrence of a trap signal by execution of the SVC instruction. (Trap number 6.)

Exit Conditions

Return Sequence

SVC Type	Description
0	M.CALL processor (retain blocked state)
1	SVC Type '1' processor
2	SVC Type '2' processor
3	M.OPEN processor
4	M.RTRN/M.RTNA processor
5	M.SURE when H.SURE is configured
6	M.CALL processor (unblocked interrupts)
7	communications software when configured
8	custom products when configured
9	diagnostics
A-D	event trace when configured
E	available for customer use
F	'CALM' replacement SVC

3.8 M.CALL SVC Processor (H.SVC0, H.SVC6)

This processor provides the means to enter any system module that has a register push down.

The default processing state for the M.CALL SVC processor for code assembled using an MPX-32 PRE file is entry point 6 (H.SVC6). This processes with unblocked interrupts. For code assembled without using an MPX-32 PRE file, entry point 0 (H.SVC0) is the default which processes with blocked interrupts.

The defaults can be overridden with the *state* parameter on individual M.CALL specifications.

Entry Conditions

Calling Sequence

The requestor must be privileged or in the case of SVC0 processing, have the system administrator attribute. Execution of the following code:

M.CALL *mm*,*ee*[,*state*]

mm is the module number being called (bits 20-23 of the SVC instruction)

ee is the entry point or SRID

state is the state of processing:

RETAIN	generates the compatible SVC0 call with blocked interrupts
UEI	generates the optimized SVC6 call with unblocked interrupts

Exit Conditions

Return Sequence

LSPD	SVCO.T1	SVCO.T1 = Address of the entry point within the called module

Registers

All registers remain intact.

Abort Cases

SV01 UNPRIVILEGED TASK ATTEMPTING TO USE M.CALL

3.9 Supervisor Call Trap Processor (H.SVC1, H.SVC2)

This trap processor is entered whenever an SVC type 1 or 2 is executed for an I/O service, Monitor service, or dynamic debug service. The processor determines the module entry point using the SVC1 or SVC2 vector tables. A register push-down and PSD save is performed and control is transferred to the specified module.

Entry Conditions

Calling Sequence

SVC type 1 or 2 instruction is executed.

Exit Conditions

Return Sequence

LPSD SVCO.T1 SVCO.T1 = address of MOD., E.P.

Registers

Same as upon detection of trap signal.

Abort Cases

SV02 INVALID SVC NUMBER

SV03 UNPRIVILEGED TASK ATTEMPTING TO USE A 'PRIVILEGED-ONLY' SERVICE

Output Messages

None

3.10 M.OPEN SVC Processor (H.SVC3)

This routine removes the task context switch inhibit state set by the M.SHUT procedure.

Entry Conditions

Calling Sequence

Execution of the following code:

M.OPEN (SVC type '3' instruction)

Exit Conditions

Return Sequence

BU S.EXEC20

3.11 M.RTRN/M.RTNA SVC Processor (H.SVC4)

This processor provides transfer control that allows system modules to return to the calling program. A register pop-up is performed with specified registers preserved returning control to the location specified or the location following the last CALM or M.CALL.

Entry Conditions

Calling Sequence

The requester must be privileged. Execution of the following code:

SVC type '4' instruction

WAIT

DATAB X'*rr*'

DATAB X'*aa*'

rr bits 0-7 indicate the registers to be preserved through the pop-up. Each register is preserved if its corresponding bit is set.

aa may contain a bit (0-7) set indicating the corresponding register containing the address to which to return.

Exit Conditions

Return Sequence

BU S.EXEC20

3.12 Invalid SVC Type Processor (H.SVCN)

Entry to this processor results in an abort of the currently executing task.

Entry Conditions

Calling Sequence

SVC type '*n*' where *n* is any SVC vector not configured (see SVC Trap Processor H.IP06:)

Exit Conditions

Return Sequence

BU S.EXEC20

Abort Cases

SV04 INVALID SVC TYPE

3.13 Machine Check Trap Processor (H.IP07)

A machine check trap is treated as a fatal system crash. Registers are loaded for display, and the M.KILL macro is invoked. Processing may be continued at the point of interrupt with the registers and PSD intact, but I/O in progress when the HALT was executed is terminated.

Entry Conditions

Calling Sequence

Occurrence of a trap signal at priority level X'07'. Entry into this routine results in registers being loaded for console display as follows:

Register	Definition
R0,R1	PSD saved by the hardware when the trap occurred
R2	physical address of instruction being executed when the trap occurred
R3	instruction being executed when trap occurred
R4	CPU status word stored when the trap occurred
R5	crash code
R6	address of register save block
R7	ASCII 'TRAP'

Exit Conditions

Return Sequence

M.KILL (crash code MC01)

Abort Cases

```
MC01    MACHINE CHECK TRAP
```

Output Messages

None

External Reference

System Macro

M.EQUS

M.KILL

3.14 System Check Trap Processor (H.IP08)

A system check trap is treated as a fatal system crash. Registers are loaded for display, and the M.KILL macro is invoked. Processing may be continued at the point of interrupt with the registers and PSD intact, but I/O in progress when the HALT was executed is terminated.

Entry Conditions

Calling Sequence

Occurrence of a trap signal at priority level X'08'. Entry into this routine results in registers being loaded for console display as follows:

Register	Definition
R0,R1	PSD saved by the hardware when the trap occurred
R2	physical address of instruction being executed when the trap occurred
R3	instruction being executed when trap occurred
R4	CPU status word stored when the trap occurred
R5	crash code
R6	address of register save block
R7	ASCII 'TRAP'

Exit Conditions

Return Sequence

M.KILL (crash code SC01, SC02, SC03, SC04)

Abort Cases

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')

Output Messages

None

External Reference

System Macro

M.EQUS

M.KILL

3.15 Map Fault Trap Processor (H.IP09)

This processor results in the task currently in execution being aborted. A register push down and PSD save is made, the abort request is reported to the scheduler, and a standard exit is performed.

Entry Conditions

Calling Sequence

Occurrence of trap signal at priority level X'09'

Exit Conditions

Return Sequence

```
BL      S.EXEC5A
```

Registers

R2	register save area address
R5	abort code
R6,R7	PSD

Abort Case

```
MF01    A MAP FAULT TRAP HAS OCCURRED.  THIS IS THE RESULT OF A
        BAD MEMORY REFERENCE OUTSIDE OF THE USER'S ADDRESSABLE
        SPACE.
```

Output Messages

None

External Reference

System Macro

M.EQUS

3.16 Address Specification Trap Processor (H.IP0C)

This processor results in an abort of the task currently in execution when the address alignment of the operand of an instruction does not agree with the requirements of the instruction or when the register address of a doubleword operation is not an even number. A register push-down and PSD save is made, the abort request is reported to the scheduler, and a standard exit is performed.

Entry Conditions

Calling Sequence

Occurrence of a trap signal at priority level X'0C'

Exit Conditions

Return Sequence

```
BL      S.EXEC5A
```

Registers

R2	register save area address
R5	abort code
R6,R7	PSD

Abort Cases

```
AD02    ADDRESS SPECIFICATION ERROR OCCURRED WITHIN THE CURRENT
        TASK
```

Output Messages

None

External Reference

System Macro

M.EQUS
M.TBLS

3.17 CPU Halt Trap Processor (H.IPHT)

This processor results in an abort of the privileged task currently in execution when a halt instruction occurs in the user's address space. A register push-down and PSD save is made, the abort request is reported to the scheduler, and a standard exit is performed.

Entry Conditions

Calling Sequence

Occurrence of a trap signal at priority level X'0E'. The CPU must be in privileged mode and the enable halt trap mode bit set with the SETCPU instruction.

Entry Conditions

Calling Sequence

```
BL      S.EXEC5A
```

Registers

R2	register save area address
R5	abort code
R6,R7	PSD

Abort Case

```
HT01    AN ATTEMPT WAS MADE TO EXECUTE A HALT INSTRUCTION IN
        USER'S PROGRAM
```

Output Messages

None

External Reference

System Macro

M.EQUS
M.TBLS

3.18 Arithmetic Exception Trap Processor (H.IP0F)

This processor has two entry points — IP0F and IP0F.0. IP0F is entered and executed by the CPU whenever the CPU detects an arithmetic exception trap. IP0F.0 is entered and executed by the IPU whenever the IPU detects an arithmetic exception trap. In both cases, the current context is saved and a common reentrant section of code is executed.

This section of code performs a set bit in memory instruction (SBM) on the arithmetic exception bit of the user currently in execution. The arithmetic exception bit may then be tested by requesting the Arithmetic Exception Inquiry monitor service.

Entry Conditions

Calling Sequence

Occurrence of an interrupt signal at priority level X'29', or trap signal at priority level X'0F'

Exit Conditions

Return Sequence

Tasks without an exception handler:

```
LPSD    IP0F.OLD  IP0F.OLD=PSD saved by interrupt
```

Tasks with an exception handler:

```
LPSD    T.EXCPAD    T.EXCPAD=task exception handler address
```

Returns to destination registers are handled as follows:

Single Precision

underflow	all zeros
positive overflow	7FFFFFFF
negative overflow	80000001

Double Precision

underflow	all zeros
positive overflow	7FFFFFFFFFFFFFFF
negative overflow	8000000000000001

Abort Cases

None

Output Messages

None

3.19 Cache Memory Parity Error Trap Processor (H.IP10)

This processor results in an abort of the task currently in execution when the hardware detects a failure in the cache memory hardware within the CPU. A register push-down and PSD save is made, the abort request is reported to the scheduler, and a standard exit is performed.

This processor is not applicable to the 32/2000 computer.

Entry Conditions

Calling Sequence

Occurrence of a trap signal at priority level X'10'

Exit Conditions

Return Sequence

```
BL      S.EXEC5A
```

Registers

R2	register save area address
R5	abort code
R6,R7	PSD

Abort Case

```
CP02    CACHE PARITY ERROR OCCURRED IN TASK BODY
```

Output Messages

None

External Reference

System Macro

M.EQUS
M.TBLS

Rapid File Allocation (H.MDT)

MPX-32 Technical Manual

Volume II

Contents

1 Rapid File Allocation (H.MDT)

1.1 General Information

The Rapid File Allocation Module (H.MDT) provides entry points and subroutines that are used by H.VOMM and J.MDTI to manipulate the memory resident descriptor table (MDT). The MDT is a memory-resident copy of permanent file resource descriptors. Because the MDT is memory resident, a file can be allocated more quickly through the MDT than through the resource descriptors.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.MDT,1	AA***	zero the MDT
H.MDT,2	AB***	locate an MDT entry and copy to a buffer
H.MDT,3	AC***	update or create an MDT entry
H.MDT,4	AD***	delete an MDT entry

*** This service is SVC 2,X'*nn*' callable. All others are SVC 1,X'*nn*' callable.

1.3 Subroutine Summary

Subroutine	Description
S.MDT1	locate an MDT entry and copy entry information to the scratchpad
S.MDT2	parse pathname
S.MDT3	hash an MDT entry
S.MDT4	locate an MDT entry through the scratchpad
S.MDT5	locate an MDT directory entry
S.MDT6	move or zero an MDT entry
S.MDT7	build a pathname block vector in the MDT resource descriptor buffer
S.MDT8	zero the MDT
S.MDT9	identify a map block boundary

2 H.MDT Entry Points

2.1 Entry Point 1 - Zero the MDT

This entry point is used by the memory resident descriptor table (MDT) initialization task (J.MDTI) to zero the MDT prior to initialization.

Entry Conditions

Calling Sequence

M.CALL H.MDT,1 (or) SVC 2,X'AA'

Registers

R2	physical address of destination buffer (MDT) to be zeroed
R5	number of words to zero

Exit Conditions

Return Sequence

M.RTRN

Registers

R0-7	unchanged

Abort Cases

None

Output Messages

None

2.2 Entry Point 2 - Locate MDT Entry and Copy to Buffer

This entry point locates a memory resident descriptor table (MDT) entry that corresponds with the specified pathname vector or pathname block vector. If the entry exists, it is copied to the specified buffer. If the entry does not exist, error status is returned in R7.

Entry Point 2 - Locate MDT Entry and Copy to Buffer

Entry Conditions

Calling Sequence

M.CALL H.MDT,2 (or) SVC 2,X'AB'

Registers

R1	pathname vector or pathname block vector
R2	buffer address

Exit Conditions

Return Sequence

M.RTRN (CC1 set indicates error)

R7 zero or error status:

Status	Description
1	invalid pathname
2	pathname consists of volume only
3	volume not mounted
7	resource does not exist
8	resource name in use
60	invalid mode

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.BWORD

System Subroutines

S.VOMM28

S.VOMM29

2.3 Entry Point 3 - Update or Create an MDT Entry

This entry point updates and creates memory resident descriptor table (MDT) entries. To create an entry, an empty MDT entry is allocated and the contents of a buffer are written to the entry. To update an entry, the entry is overwritten by the buffer's contents.

Entry Conditions

Calling Sequence

M.CALL H.MDT,3 (or) SVC 2,X'AC'

Registers

R1 pathname vector or pathname block vector

R2 buffer address (this is not validated)

R7 mode of the call:

Mode	Meaning
0	update
1	create

Exit Conditions

Return Sequence

M.RTRN (CC1 set indicates error)

Registers

R7 zero or error status:

Status	Description
1	invalid pathname
2	pathname consists of volume only
3	volume not mounted
5	file is not a permanent file
7	resource does not exist
8	resource name in use
60	invalid mode

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.BWORD

System Subroutines

S.VOMM23
S.VOMM28
S.VOMM29

2.4 Entry Point 4 - Delete an MDT Entry

This entry point deletes a memory resident descriptor table (MDT) entry. The MDT entry count, C.MDTAV, is updated.

Entry Conditions

Calling Sequence

M.CALL H.MDT,4 (or) SVC 2,X'AD'

Registers

R1 pathname vector or pathname block vector

R2 buffer address containing the resource descriptor of the entry to be deleted, or zero if the VOMM rename service is used

Exit Conditions

Return Sequence

M.RTRN (CC1 set indicates error)

Registers

R7 zero or error status:

Status	Description
1	invalid pathname
2	pathname consists of volume only
3	volume not mounted
7	resource does not exist

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.BWORD

System Subroutines

S.VOMM23
S.VOMM28
S.VOMM29

3 H.MDT Subroutines

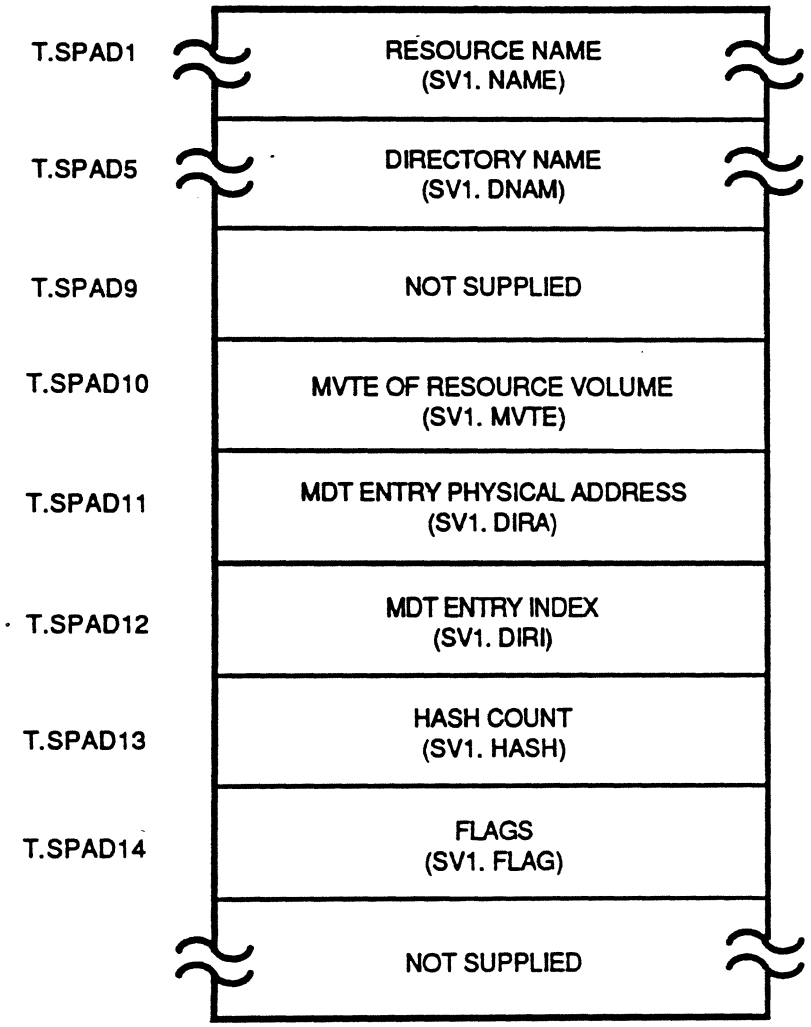
3.1 Subroutine S.MDT1

3.1.1 Locate MDT Entries and Copy Information

This routine locates a memory resident descriptor table (MDT) entry in one of four modes:

- locate/log
- create
- delete
- update

Entry information is returned to the scratchpad in the following format:



87D12T12

Entry Conditions

Calling Sequence

BL S.MDT1

Registers

R1 pathname vector or pathname block vector

R7 mode of the call:

Mode	Meaning
1	locate/log
2	create
4	delete
8	update

Exit Conditions

Return Sequence

BU SV1.XOK (normal)

(or)

BU SV1.XXX (error - CC1 set)

Registers

R0-6 unchanged

R7 unchanged or error status:

Status	Description
1	invalid pathname
2	pathname consists of volume only
3	volume not mounted
7	resource does not exist
8	resource name in use
60	invalid mode

Scratchpad Usage

T.SPAD1-4	SV1.NAME
T.SPAD5-8	SV1.DNAM
T.SPAD10	SV1.MVTE
T.SPAD11	SV1.DIRA
T.SPAD12	SV1.DIRI
T.SPAD13	SV1.HASH
T.SPAD14	SV1.FLAG

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.PUSH

3.2 Subroutine S.MDT2 - Parse Pathname

This routine verifies the pathname. The parsed pathname is returned in the scratchpad.

Entry Conditions

Calling Sequence

BL S.MDT2

Registers

R1 pathname vector or pathname block vector

R7 mode of the call:

Mode	Meaning
1	locate
2	create
4	delete

Exit Conditions

Return Sequence

BU SV1.XOK (Normal)
(or)
BU SV1.XXX (error - CC1 set)

Registers

R1-6 unchanged

R7 unchanged or error status

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.PUSH

3.3 Subroutine S.MDT3 - Hash an MDT Entry

This routine hashes a memory resident descriptor table (MDT) entry using the directory name, resource name, and number of MDT entries.

Entry Conditions

Calling Sequence

```
BL      S.MDT3
```

Input

T.SPAD1-4	resource name
T.SPAD5-8	directory name

Exit Conditions

Return Sequence

```
BU      SV1.XOK
```

Registers

R1-7	unchanged

Scratchpad Usage

T.SPAD11	directory address
T.SPAD12	hash index

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.PUSH

3.4 Subroutine S.MDT4

3.4.1 Locate an MDT Entry through the Scratchpad

This routine locates an available or existing entry depending on the flags stored in the scratchpad. If a new MDT entry is to be created, the available MDT entry is marked as allocated. This prevents another task from allocating the entry before creation is complete.

Entry Conditions

Calling Sequence

BL	S.MDT4

Input

T.SPAD14	flags

Exit Conditions

Return Sequence

BU	SV1.XOK (normal)
	(or)
BU	SV1.XXX (error)

Registers

R1-6	unchanged
R7	unchanged or error status

Status	Description
7	resource does not exist
8	resource name in use
10	MDT entry unavailable

Scratchpad Usage

T.SPAD11	directory address
T.SPAD12	hash index

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.PUSH

3.5 Subroutine S.MDT5 - Locate an MDT Directory Entry

This routine locates a memory resident descriptor table (MDT) by searching a particular directory. The name of the directory to be searched must be stored in the scratchpad.

Entry Conditions

Calling Sequence

BL S.MDT5

Input

R4 flags

R5 mode of the call:

Mode	Meaning
0	locate
1	create
-1	delete

T.SPAD1 resource name

Exit Conditions

Return Sequence

BU SV1.XOK (normal)
(or)
BU SV1.ER07 (error)
(or)
BU SV1.ER08 (error)

Registers

R1-6 unchanged

R7 error status:

Status	Description
7	resource does not exist
8	resource name in use
10	MDT entry unavailable

Scratchpad Usage

T.SPAD11	MDT entry physical address
T.SPAD12	MDT entry index

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.POP

M.PUSH

3.6 Subroutine S.MDT6 - Move or Zero an MDT Entry

This routine moves a block of words from one location to another, or zeros a range of memory locations. The move is performed unmapped.

Warning: If a transfer spans a map block boundary, corruption of another task's logical address space may occur. S.MDT9 can be used to identify map block boundaries.

Entry Conditions

Calling Sequence

BL	S.MDT6

Registers

R1	physical source address
R2	physical destination address
R5	number of words to move, or zero to zero the specified range

Exit Conditions

Return Sequence

BU	SV1.XOK

Registers

R0-R7	unchanged

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.PUSH

3.7 Subroutine S.MDT7

3.7.1 Build a Pathname Block Vector in the MDT

This routine builds a pathname block (PNB) vector in the last 20 words of the memory resident descriptor table's (MDT) resource descriptor buffer. The buffer is located in VOMM's dynamic memory area. The resource descriptor's buffer address is passed to this routine; the address is used in building the PNB vector.

Warning: The address must be for a permanent file.

Entry Conditions

Calling Sequence

BL	S.MDT7

Registers

R2	resource descriptor's buffer address

Exit Conditions

Return Sequence

BU	SV1.XOK

Registers

R1-7	unchanged

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.PUSH

3.8 Subroutine S.MDT8 - Zero the MDT

This routine is used by J.MDTI to zero the memory resident descriptor table (MDT) before the individual entries are initialized.

Entry Conditions

Calling Sequence

BL	S.MDT8

Registers

R2	MDT physical starting address
R5	length of MDT in words

Exit Conditions

Return Sequence

BU	SV1.XOK

Registers

R1-7	unchanged

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.PUSH

3.9 Subroutine S.MDT9 - Identify a Map Block Boundary

This routine identifies a map block boundary, if one exists, in a buffer. If the buffer spans two map blocks, the number of words in each map block is returned.

Warning: Swapping must be inhibited before calling this routine.

Entry Conditions

Calling Sequence

BL	S.MDT9

Registers

R1	buffer address
R5	buffer length

Exit Conditions

Return Sequence

BU	SV1.XOK

Registers

R1-5	unchanged
R6,7	zero if buffer contained within one map block
	(or)
R6	number of words in first map block
R7	number of words in second map block

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.PUSH

Memory Management (H.MEMM)

MPX-32 Technical Manual

Volume II

Contents

Contents

1 H.MEMM Overview

1.1 General Information

The Memory Management Module (H.MEMM) allocates memory and builds memory partitions. This module can reside in extended memory.

1.2 Entry Points

Entry Point	SVC Number	Description
H.MEMM,1	N/A	allocate memory
H.MEMM,2	N/A	deallocate memory
H.MEMM,3	69	get dynamic extended data space
H.MEMM,4	6A	free dynamic extended indexed data space
H.MEMM,5	67	get dynamic task execution space
H.MEMM,6	68	free dynamic task execution space
H.MEMM,7	40*	include memory partition
H.MEMM,8	41*	exclude memory partition
H.MEMM,9	7C*	get dynamic extended discontiguous data space
H.MEMM,10	N/A	reserved
H.MEMM,11	N/A	deallocate memory due to swapping
H.MEMM,12	4B*	get memory in byte increments
H.MEMM,13	4C*	free memory in byte increments
H.MEMM,14	7F*	get extended memory array

* This service is SVC 2,X'*nn*' callable. All others are SVC 1,X'*nn*' callable.

N/A implies reserved for internal use by MPX-32.

1.3 Subroutine Summary

Subroutine	Description
S.MEMM1	reserved
S.MEMM2	locate shared memory table entry
S.MEMM5	update map segment descriptor for memory increase
S.MEMM7	remap user's address space
S.MEMM8	validate buffer address
S.MEMM11	deallocate debugger memory
S.MEMM12	create a protection image
S.MEMM13	reserved
S.MEMM14	update shared memory protection image
S.MEMM15	reserved
S.MEMM16	get system buffer space
S.MEMM17	free system buffer space
S.MEMM18	get map image information
S.MEMM19	update map segment descriptor for memory decrease
S.MEMM20	check new task size
S.MEMM21	create pathname search identifier
S.MEMM22	get specified physical map block
S.MEMM23	deallocate map memory

2 H.MEMM Entry Points

2.1 Entry Point 1 - Allocate Memory

This entry point is called by H.TAMM entry points 2 and 3, and H.MEMM entry points 3, 5, and 12. It is also called by the swapper. It allocates the memory required for the calling task. The memory is returned in the form of map image descriptors (MIDL) and memory attributes (MEML), one MIDL and one MEML per map block. Swappable map counts in the DQE are incremented as needed based on the MEML information. The entries in the memory allocation table are updated to reflect allocation. Protection granules are returned as unprotected.

Entry Conditions

Calling Sequence

M.CALL H.MEMM,1 or H.REMM,4

Registers

R1 address of MIDL or zero

R3 address of MEML or physical map to begin allocation if R1 is zero

R5 right halfword is the number of map blocks required

Left halfword:

Value	Memory Class
1	E
2	H
3	S

left halfword not used if physical request

Exit Conditions

Return Sequence

M.RTRN or M.RTRN R5

Registers

If the request cannot be fully satisfied, no memory is allocated by this service.

CC1 is set if unable to allocate all required memory and R5 contains the number of map blocks that cannot be allocated now.

CC1 is reset if request is successful and R5 is unchanged.

Physical memory definitions in MIDL. Memory attributes returned in MEML.

DQE.CME, DQE.CMH, and DQE.CMS incremented for each swappable map.

2.2 Entry Point 2 - Deallocate Memory

This entry point is called by H.TAMM,4 and H.MEMM entry points 4, 6, 8, and 13. It deallocates memory as directed by the map image descriptor list (MIDL) and the memory attribute list (MEML). Memory can be of mixed types and of mixed swappable characteristics. The swappable count in the DQE is updated according to the information in the MEML. Protection granules are set to show that a protected map hole exists. If R1 is zero then deallocation begins with the physical map number given and the caller is responsible for updating the MIDL, MEML, and DQE counts.

Entry Conditions

Calling Sequence

M.CALL H.MEMM,2 or H.REMM,5

Registers

R1	address of MIDL or zero
R3	address of MEML or physical map number if R1 is zero
R5	number of map blocks to deallocate

Exit Conditions

Return Sequence

M.RTRN

Registers

DQE.CME, DQE.CMH, and DQE.CMS are decremented for each swappable map.

Scratchpad Usage

None

2.3 Entry Point 3 - Get Dynamic Extended Data Space

See M.GD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.4 Entry Point 4 - Free Dynamic Extended Indexed Data Space

See M.FD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.5 Entry Point 5 - Get Dynamic Task Execution Space

See M.GE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.6 Entry Point 6 - Free Dynamic Task Execution Space

See M.FE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.7 Entry Point 7 - Include Memory Partition

See M.INCLUDE or M_INCLUDE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.8 Entry Point 8 - Exclude Memory Partition

See M.EXCLUDE or M_EXCLUDE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.9 Entry Point 9 - Get Dynamic Extended Discontiguous Data Space

See M.GDD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.10 Entry Point 10 - Reserved

2.11 Entry Point 11 - Deallocate Memory Due to Swapping

This entry point is called only by the swapper. It deallocates memory as directed by the map image descriptor list (MIDL) and the memory attribute list (MEML) that are required inputs to this routine. Memory can be of mixed types but all map blocks are swappable. The swappable count in the DQE is updated according to the information in the MEML. The protection granules are not changed.

Entry Conditions

Calling Sequence

```
M.CALL    H.MEMM,11
```

Registers

R1	address of MIDL
R3	address of MEML
R5	number of map blocks to deallocate

Note: DQE.CME, DQE.CMH and DQE.CMS are decremented for each swappable map. Protection registers are unchanged. MEML and MIDL reflect maps deallocated.

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.12 Entry Point 12 - Get Memory in Byte Increments

See M.MEMB or M _GETMEMBYTES in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.13 Entry Point 13 - Free Memory in Byte Increments

See M.MEMFRE or M _FREEMEMBYTES in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.14 Entry Point 14 - Get Extended Memory Array

See the Get Extended Memory Array system service in the MPX-32 Reference Manual Volume I.

3 H.MEMM Subroutines

3.1 Subroutine S.MEMM1 - Reserved

3.2 Subroutine S.MEMM2 - Locate Shared Memory Table Entry

This subroutine finds the first entry in the shared memory table (SMT) that contains the name and task number specified by the caller. It also finds a free entry.

Entry Conditions

Calling Sequence

BL	S.MEMM2

Registers

R1	address of partition RID or pathname shared identifier
R4,R5	8-character owner name (dynamic only)
R6,R7	first eight characters of partition or shared image name if the search parameter in R1 is a pathname identifier; otherwise, registers not used.

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1	address of SMT entry or next available entry if not found
R2	current stack pointer
R3	TSA address
R4-R6	destroyed
R7	error status; otherwise, R7 is destroyed

Status

CC1 set	status = 5	SMT entry not found
CC2 set	status = 58	SMT entry not available
CC3 set	building entry found during scan	

Scratchpad Usage

T.SPAD5	used to save return address

3.3 Subroutine S.MEMM5 - Update Map Segment Descriptor

This subroutine updates the map segment page count contained in DQE.MSD for memory allocations.

Entry Conditions

Calling Sequence

BL	S.MEMM5

Registers

R1	starting T.MIDL address
R5	bytes 0,1 contain the memory type bytes 2,3 contain the number of maps

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2	DQE address
R3	TSA address
R4	destroyed

Status

None

Scratchpad Usage

None

3.4 Subroutine S.MEMM7 - Remap User's Address Space

This subroutine is used during allocation and deallocation of memory to add or delete physical maps from the user's address space. The hardware map registers are reloaded with the new map images.

Entry Conditions

Calling Sequence

```
BL      S.MEMM7
```

Registers

None (DQE.MSD, T.MEML, and T.MIDL have new image)

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

```
R3-R5   destroyed
```

Status

None

Scratchpad Usage

```
T.PSD1  used to perform remap (restored)
```

3.5 Subroutine S.MEMM8 - Validate Buffer Address

This subroutine is used to verify a logical address provided by the user. The following inquiries are made:

- Is the starting address lower than the DSECT start address?
- Do the addresses specified cross a map block boundary?
- Are the addresses specified in a valid map block?
- Are the addresses specified in the extended address space?
- Are the addresses specified in a protected area?

Entry Conditions

Calling Sequence

BL	S.MEMM8

Registers

R6	logical starting address
R7	number of bytes to validate (buffer size). If zero, single address check is made.

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1	R3 is copied to R1 (restores context for IOCS)
R2,R4,R5	destroyed

Status

Code	Meaning if Set
CC4	invalid address. For example, not mapped into user's space.
CC3	locations specified are protected
CC2	buffer crosses map block boundary

3.6 Subroutine S.MEMM11 - Deallocate Debugger Memory

This subroutine deallocates the memory that was dynamically allocated for the loading of the task debugger into the calling task's space. This routine is called by H.REXS,30.

Entry Conditions

Calling Sequence

BL S.MEMM11

Registers

None

Exit Conditions

Return Sequence

TRSW R7 (R7 contains return address)

Registers

All registers are destroyed.

3.7 Subroutine S.MEMM12 - Create a Protection Image

This subroutine creates a protection image for the calling task. The protection granules in the MIDL entry are set. This subroutine protects every map block in the task's logical space. This is allowed because when this subroutine is called, the only memory allocated to the task is the memory for the TSA, which needs to be protected.

Entry Conditions

Calling Sequence

BL S.MEMM12

Registers

R3 TSA address

Exit Conditions

Return Sequence

TRSW R0

Registers

All registers are destroyed.

3.8 Subroutine S.MEMM13 - Reserved

3.9 Subroutine S.MEMM14 - Update Shared Memory Protection Image

This subroutine updates the protection image of the calling task when a shared memory partition is included. The protection granules in the MIDL are updated.

Entry Conditions

Calling Sequence

BL	S.MEMM14

Registers

R1	shared memory table address
R4	logical shared memory address

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2-R7	destroyed

3.10 Subroutine S.MEMM15 - Reserved

3.11 Subroutine S.MEMM16 - Get System Buffer Space

This subroutine is used by system services requiring temporary buffer space. Logical contiguous memory is allocated in the DSECT of the task that called the system service. If insufficient space is available in the DSECT, the memory is allocated in the extended address space. Memory is allocated from low logical addresses to high.

Note: Calls to this subroutine should not be interleaved with calls to the dynamic allocation services (H.MEMM,3 and H.MEMM,5) because holes may develop in the task's address space.

Entry Conditions

Calling Sequence

BL	S.MEMM16

Registers

R5	number of bytes of dynamic space requested

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3	logical start address of dynamic space
R5	number of bytes allocated (number requested is rounded up to the nearest map block boundary)
R1,R2,R4,R6	destroyed
R7	error status; otherwise, R7 is destroyed

Error Condition

CC1	set
R7	error code as follows:

Value	Definition
1	number of bytes requested is negative or zero
2	logical space unavailable in task
4	attempt to expand task beyond limits of physical memory

Abort Cases

None

Scratchpad Usage

T.SPAD22	used for temporary storage

3.12 Subroutine S.MEMM17 - Free System Buffer Space

This subroutine frees space that was allocated by S.MEMM16. Temporary buffer space is returned starting with the high logical address in the task's extended space and working down. If all the extended space allocated by S.MEMM16 is freed and the request is still not satisfied, temporary space in the DSECT is returned, again working from high logical memory toward low.

Note: Services which use this subroutine should not call the dynamic deallocation services (H.MEMM,4 and H.MEMM,6).

Entry Conditions

Calling Sequence

```
BL      S.MEMM17
```

Registers

R3	starting logical address of the space to deallocate
R5	number of bytes of dynamic space to deallocate

Exit Conditions

Return Sequence

Registers

All registers are destroyed.

Error Condition

CC1	invalid system buffer deallocation; only those maps within the deallocation request range marked as system space are deallocated

3.13 Subroutine S.MEMM18 - Get Map Image Information

This subroutine scans a task's MIDL from a given lower bound up to and including a given upper bound. The scan returns information pertinent to the allocation or deallocation of memory.

Entry Conditions

Calling Sequence

BL	S.MEMM18

Registers

R1	lower bound map index (relative to T.MIDL)
R2	upper bound map index (relative to T.MIDL)
R5	number of contiguous maps desired

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1	address of next available MIDL entry
R2	address of last allocated MIDL entry (non-shared)
R4	address of next available MIDL entry (may be discontiguous)
R6,R7	destroyed

Status

CC1	requested space not available

Scratchpad Usage

None

3.14 Subroutine S.MEMM19 - Update Map Segment Descriptor

This subroutine updates the map segment page count in DQE.MSD after memory deallocation.

Entry Conditions

Calling Sequence

```
BL      S.MEMM19
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R2	DQE address
R3	TSA address
R4	new map segment descriptor count
R5	destroyed

Status

None

Scratchpad Usage

None

3.15 Subroutine S.MEMM20 - Check New Task Size

This subroutine is called before memory is allocated. It determines the new size of the task in map blocks and verifies that the new task size does not exceed the total amount of physical memory configured.

Entry Conditions

Calling Sequence

```
BL      S.MEMM20
```

Registers

R3	TSA address
R5	number of map blocks to be added to task space

Exit Conditions

Return Sequence

TRSW	R0

Registers

R4	number of map blocks in excess of physical memory if CC1 is set; otherwise, R4 is destroyed
R5	destroyed

Status

CC1	task size exceeds total memory configured

3.16 Subroutine S.MEMM21 - Create Pathname Search Identifier

This subroutine converts a pathname or pathname block into a one word pathname search identifier. The identifier is used to search the shared memory tables, by pathname, for an entry.

Entry Conditions

Calling Sequence

BL	S.MEMM21

Registers

R1	pathname or pathname block vector to be converted
R2	address of 18 word work space if a pathname is to be converted or zero if a pathname block is to be converted

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1	pathname search identifier
R2	current stack address
R3	TSA address
R4,R5	destroyed
R6,R7	first eight characters of file name

Scratchpad Usage

The following scratchpad locations are only used when R1, on entry, is a pathname block and the file name portion is less than 16 characters:

T.SPAD 3,4	used to save eight characters of file name
T.SPAD 5,6	used for calculation of file name portion of pathname block

3.17 Subroutine S.MEMM22 - Get Specified Physical Map Block

This subroutine acquires physical map blocks for the operating system. The subroutine checks whether the specified physical map block is available. If available, the block is allocated. If not available, an allocation denial is returned. Only one map block can be requested per subroutine call.

Map blocks allocated in this manner are unswappable. If sharing is desired, MEML.SHR must be set. The acquired map blocks can be deallocated normally.

Entry Conditions

Calling Sequence

BL	S.MEMM22

Registers

R1	address of MIDL entry
R3	address of MEML entry
R5	physical map block requested (zero based)

Exit Conditions

Return Sequence

TRSW R0

Registers

All preserved (if CC1 not set)

Status

CC1 not set: allocation performed

CC1 set: allocation denied

Value	Description
R5= X'100'	invalid physical map number
R5< X'100'	status byte MEM.STAT from the memory allocation table (see MPX-32 Technical Manual Volume I, Chapter 2)

Scratchpad Usage

None

3.18 Subroutine S.MEMM23 - Deallocate Map Memory

This subroutine deallocates memory that was previously acquired.

Entry Conditions

Calling Sequence

BL S.MEMM23

Registers

R1	address of MIDL or zero
R3	address of MEML or physical map number if R1 is zero
R5	number of maps to deallocate
R4	number of maps to deallocate or not applicable if R1 is zero

Exit Conditions

Return Sequence

TRSW R0

Registers

R1-R5,R7 destroyed
R6 preserved

Scratchpad usage:

None

Memory Pool Management (H.MEMM2)

MPX-32 Technical Manual

Volume II

Contents

1 H.MEMM2 Overview

1.1 General Information

The Memory Pool Management Module (H.MEMM2) manages the allocation and deallocation of memory pool buffers.

1.2 Subroutine Summary

Subroutine	Description
S.MEMM9	allocate memory pool buffer
S.MEMM9A	allocate IOQ memory pool
S.MEMM9B	allocate MSG memory pool
S.MEMM10	release memory pool buffer
S.MEMM24	update system data structures for memory allocation
S.MEMM25	update system data structures for memory deallocation

2 H.MEMM2 Subroutines

2.1 Subroutine S.MEMM9 - Allocate Memory Pool Buffer

This subroutine is used by system services requiring temporary memory for I/O queue entries, I/O buffering, and messages. Allocated and free lists are maintained to validate subsequent deallocation requests. Buffers are allocated on a doubleword boundary. The operating system uses S.MEMM9 to allocate memory pool as required. Therefore, avoid indiscriminately using this subroutine so that system contention for memory pool does not occur.

Entry Conditions

Calling Sequence

BL	S.MEMM9

Registers

R7	number of words required

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1,R2	destroyed
R3	starting doubleword-bounded address of memory pool
R7	number of words rounded to two word increment

Status

CC1	no memory available and R3 is zero

Scratchpad Usage

None

2.2 Subroutine S.MEMM9A - Allocate IOQ Memory Pool

This subroutine allocates temporary memory for system services from IOQ memory pool. Memory is allocated on a doubleword boundary. If there is no room available in IOQ pool, a branch to S.MEMM9 is made to try the miscellaneous pool. For permanent IOQs, no rollover to the miscellaneous pool is done if the NOROLL option is set. Permanent IOQs are indicated by bottom up allocation.

Entry Conditions

Calling Sequence

BL	S.MEMM9A

Registers

R7	number of words required. If bit 0 set, indicates allocation is from the bottom up.

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3	memory pool buffer address (doubleword bounded)
R7	number of words allocated (modulo 2W)

Status

CC1 set	no memory pool space available
R3	zero

2.3 Subroutine S.MEMM9B - Allocate MSG Memory Pool

This subroutine allocates temporary memory for system services from MSG memory pool. Memory is allocated on a doubleword boundary. If there is no room available in MSGPOOL, a branch to S.MEMM9 is made to try the miscellaneous pool if C.MSGNR (no roll over option) is not set. Otherwise, return with status is processed.

Entry Conditions

Calling Sequence

BL	S.MEMM9B

Registers

R7	number of words required. If bit 0 is set, indicates allocation is from the bottom up.

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3	memory pool buffer address (doubleword bounded)
R7	number of words allocated (modulo 2W)

Status

CC1 set	no memory pool space available
R3	zero

2.4 Subroutine S.MEMM10 - Release Memory Pool Buffer

This subroutine deallocates a previously allocated memory pool buffer if the request matches an entry in the memory pool allocated list.

Entry Conditions

Calling Sequence

BL	S.MEMM10

Registers

R3	starting address of memory pool
R7	number of words to deallocate

Exit Conditions

Return Sequence

TRSW	R0 (with S.EXEC7)

Registers

R1,R2,R4,R7	destroyed

Status

If event trace is on and the system debugger is present, the debugger prompt is displayed on the operator's console and R1 contains one of the following abort conditions:

Code	Definition
1	buffer address in R3 is not in memory pool
2	buffer address in R3 is not allocated
3	invalid byte count is specified in the deallocation request

2.5 Subroutine S.MEMM24 - Update System Structures for Memory Allocate

This subroutine updates the MATA entries to show the map block is being allocated, and updates the MPTL to unlink this map block from the LRU free list. It also updates the PTE if it exists.

Entry Conditions

Calling Sequence

Unmapped Mode:

BL S.MEMM24

Registers

R1 phyiscal address of MIDLs or zero

R2 memory type to allocate

Bit	Memory Type
0	E
1	H
2	S
3	D
4	MP (Multiprocessor shared)

R3 physical address of MEMLs or zero

R7 map block number to allocate or zero

Note: If R7 is a map block number, R2 is ignored.

Exit Conditions

Return Sequence

TRSW R0

Registers

R7 map number allocated

R0,R1,R4,R6 preserved, all others are destroyed

Status

CC1 set if map block requested is not free

CC2 set if no map blocks of type requested are available for allocation

2.6 Subroutine S.MEMM25 - Update Data Structures for Memory Deallocate

This subroutine updates the MATA entries to show the map block is being deallocated, and updates the MPTL to add this map block at the end of the LRU free list. It also updates the PTE if it exists and PST if desired.

Entry Conditions

Calling Sequence

Unmapped Mode:

```
BL      S.MEMM25
```

Registers

R3	address of MEML or zero if no PST is updated.
R7	physical map block number to deallocate

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R0,R1,R3-R7 unchanged

R2	destroyed

System Services (H.MONS)
MPX-32 Technical Manual
Volume II

Contents

Contents

Page

1 H.MONS Overview

1.1 General Information

The system services module (H.MONS) performs the compatible mode monitor system services.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.MONS,1	42	physical device inquiry
H.MONS,2	43	permanent file address inquiry
H.MONS,3	44	memory address inquiry
H.MONS,4	45	create timer entry
H.MONS,5	46	test timer entry
H.MONS,6	47	delete timer entry
H.MONS,7	48	set user status word
H.MONS,8	49	test user status word
H.MONS,9	4A*	change priority level
H.MONS,10	4B	connect task to interrupt
H.MONS,11	4E	time of day inquiry
H.MONS,12	4F	memory dump request
H.MONS,13	50	load overlay segment
H.MONS,14	51	load and execute overlay segment
H.MONS,15	52	activate task
H.MONS,16	53	resume task execution
H.MONS,17	54	suspend task execution
H.MONS,18	55	terminate task execution
H.MONS,19	56	abort specified task
H.MONS,20	57	abort self
H.MONS,21	40	allocate file or peripheral device
H.MONS,22	41	deallocate file or peripheral device
H.MONS,23	4D	arithmetic exception inquiry
H.MONS,24	4C	task option word inquiry
H.MONS,25	58	program hold request
H.MONS,26	60	set user abort receiver address
H.MONS,27	61	submit job from disk file
H.MONS,28	62	abort with extended message
H.MONS,29	63	load and execute interactive debugger
H.MONS,30	N/A	delete interactive debugger
H.MONS,31	5A	delete task
H.MONS,32	64	get task number
H.MONS,33	73	permanent file log
H.MONS,34	74	username specification

* Implies that this service is available to privileged users only. N/A implies reserved for internal use by MPX-32.

Entry Point Summary

Entry Point	SVC Number	Description
H.MONS,35	7A	get message parameters
H.MONS,36	7B	get run parameters
H.MONS,37	7C	wait for any no-wait operation complete, message interrupt or break interrupt
H.MONS,38	5D	disconnect task from interrupt
H.MONS,39	5E	exit from message receiver
H.MONS,40	5F*	parameter task activation
H.MONS,41	65	get address limits
H.MONS,42	66	debug link service
H.MONS,43	6B	receive message link address
H.MONS,44	6C	send message to specified task
H.MONS,45	6D	send run request to specified task
H.MONS,46	6E	break/task interrupt link
H.MONS,47	6F	activate task interrupt
H.MONS,48	70	exit from task interrupt level
H.MONS,49	7D	exit run receiver
H.MONS,50	7E	exit from message end-action routine
H.MONS,51	7F	exit from run request end-action routine
H.MONS,52	N/A	RTM CALM terminate task execution
H.MONS,53	N/A	RTM CALM activate task
H.MONS,54	N/A	RTM CALM suspend task execution
H.MONS,55	N/A	RTM CALM allocation file or device
H.MONS,56	N/A	RTM CALM physical device inquiry
H.MONS,57	2E	disable message task interrupt
H.MONS,58	2F	enable message task interrupt
H.MONS,59	N/A	get physical memory contents
H.MONS,60	N/A	change physical memory contents
H.MONS,61	N/A	RTM CALM permanent file log
H.MONS,62	19	resourcemark lock
H.MONS,63	1A	resourcemark unlock
H.MONS,64	N/A	remove RSM lock on task termination
H.MONS,65	2D	task CPU execution time
H.MONS,66	1E	activate program at given time of day
H.MONS,67	1B	set synchronous task interrupt
H.MONS,68	1C	set asynchronous task interrupt
H.MONS,69		reserved
H.MONS,70	15	date and time inquiry
H.MONS,71	14	get device mnemonic or type code
H.MONS,72	13	enable user break interrupt
H.MONS,73	12	disable user break interrupt
H.MONS,99	N/A	SYSGEN initialization

* Implies that this service is available to privileged users only. N/A implies reserved for internal use by MPX-32.

1.3 RTM System Services Under MPX-32

Generally, RTM CALM's operate under MPX-32 without any change in syntax or function. A few seldom-used CALM's have been deleted, and others may have additional restrictions applied to them. In general, however, the changes to the user's source code should be minimal in the conversion from RTM to MPX-32.

SVC type 15 replaces CALM instructions. During reassembly of a program, the assembler automatically converts CALM instructions to their equivalent SVC 15,X'*nn*' number if option 20 is set.

Also, an address exception trap is generated when a doubleword operation code is used with an incorrectly bounded operand; therefore, coding changes are required when a trap occurs.

Under MPX-32 the following RTM CALM implementation is slightly different from its RTM equivalent:

CALM X'73' Permanent File Log (The file definition is returned in sectors instead of allocation units).

The following RTM CALM's have been deleted in MPX-32:

CALM X'62' Unlink Dynamic Job Queue Entry (not required in MPX-32). M.DDJS or CALL M:UNLKJ

CALM X'63' Activate with Core Append (replaced by memory expansion and contraction services of MPX-32). (M.ACAP was not in RTM run-time)

CALM X'64' Retrieve Address of Appended Core (same as CALM X'63'). (M.APAD was not in RTM run-time)

CALM X'65' Initialize reentrant library pointers (MPX-32 does not support the RTM reentrant run-time library).

All Random Access Calls (MPX-32 does not support DRAH)

CALM X'59' random access OPEN (CALL OPEN)

CALM X'5A' random access READ (CALL READ)

CALM X'5B' random access WRITE (CALL WRITE)

CALM X'5C' random access file (CALL DEFINE)

CALM X'5D' random access file (CALL FIND)

TSS CALM's

MPX-32 replaces TSS with TSM, an on-line support package. Therefore, all TSS CALM's X'80' - X'84' have been deleted.

2 H.MONS Entry Points

2.1 Entry Point 1 - Physical Device Inquiry

See M.PDEV in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Service

H.REMM,27

System Subroutine

S.REXS8

2.2 Entry Point 2 - Permanent File Address Inquiry

See M.FADD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Service

H.FISE,10

2.3 Entry Point 3 - Memory Address Inquiry

See M.ADRS or M_ADRS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.4 Entry Point 4 - Create Timer Entry

See M.SETT or M_SETT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.5 Entry Point 5 - Test Timer Entry

See M.TSTT or M_TSTT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.6 Entry Point 6 - Delete Timer Entry

See M.DLTT or M_DLTT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.7 Entry Point 7 - Set User Status Word

See M.SETS or M_SETS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.8 Entry Point 8 - Test User Status Word

See M.TSTS or M_TSTS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.OPEN

2.9 Entry Point 9 - Change Priority Level

See M.PRIL or M_PRIL in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.OPEN

2.10 Entry Point 10 - Connect Task to Interrupt

See M.CONN or M_CONN in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.11 Entry Point 11 - Time-Of-Day Inquiry

See M.TDAY or M_TDAY in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.12 Entry Point 12 - Memory Dump Request

See M.DUMP or M_DUMP in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.SPAD
M.CALL
M.RTRN

2.13 Entry Point 13 - Load Overlay Segment

See M.OLAY in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

2.14 Entry Point 14 - Load and Execute Overlay

See M.OLAY in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

2.15 Entry Point 15 - Activate Task

See M.ACTV or M_ACTV in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.16 Entry Point 16 - Resume Task Execution

See M.SUME or M_SUME in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN
M.OPEN
M.IOFF
M.IONN

2.17 Entry Point 17 - Suspend Task Execution

See M.SUSP or M_SUSP in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.OPEN
M.IONN
M.IOFF

2.18 Entry Point 18 - Terminate Task Execution

See M.EXIT or M_EXIT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.19 Entry Point 19 - Abort Specified Task

See M.BORT or M_BORT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.IOFF
M.IONN
M.OPEN

2.20 Entry Point 20 - Abort Self

See M.BORT or M_BORT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL

2.21 Entry Point 21 - Allocate File or Peripheral Device

See M.ALOC in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL

System Service

H.ALOC,6

System Subroutine

S.REXS8

2.22 Entry Point 22 - Deallocate File or Peripheral Device

See M.DALC in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.23 Entry Point 23 - Arithmetic Exception Inquiry

See M.TSTE or M_TSTE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.24 Entry Point 24 - Task Option Word Inquiry

See M.PGOW or M_OPTIONWORD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.25 Entry Point 25 - Program Hold Request

See M.HOLD or M_HOLD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.26 Entry Point 26 - Set User Abort Receiver Address

See M.SUAR or M_SUAR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.SPAD
M.CALL
M.RTRN

2.27 Entry Point 27 - Submit Job from Disc File

See M.CDJS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.28 Entry Point 28 - Abort with Extended Message

See M.BORT or M_BORT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.IOFF
M.IONN
M.OPEN

2.29 Entry Point 29 - Load and Execute Interactive Debugger

See M.DEBUG or M_DEBUG in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTNA

2.30 Entry Point 30 - Delete Interactive Debugger

See the H.REXS,30 entry point description in this manual for a detailed description of this entry point.

2.31 Entry Point 31 - Delete Task

See M.DELTSK or M_DELTSK in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.IOFF
M.IONN
M.OPEN

2.32 Entry Point 32 - Get Task Number

See M.ID or M_ID, and M.MYID or M_MYID in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.RTNA

2.33 Entry Point 33 - Permanent File Log

See M.LOG in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Service

H.VOMM,10

System Subroutine

S.REXS8

2.34 Entry Point 34 - Username Specification

See M.USER in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

2.35 Entry Point 35 - Get Message Parameters

See M.GMSGP or M_GMSGP in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.36 Entry Point 36 - Get Run Parameters

See M.GRUNP or M_GRUNP in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.37 Entry Point 37 - Wait for Any No-wait Operation

See M.ANYW or M_ANYWAIT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.38 Entry Point 38 - Disconnect Task from Interrupt

See M.DISCON or M_DISCON in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.39 Entry Point 39 - Exit from Message Receiver

See M.XMSGR or M_XMSGR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.40 Entry Point 40 - Parameter Task Activation

See M.PTSK or M_PTSK in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.41 Entry Point 41 - Get Address Limits

See M.GADRL in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.42 Entry Point 42 - Debug Link Service

See the Debug Link system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTNA

2.43 Entry Point 43 - Receive Message Link Address

See M.RCVR or M_RCVR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.44 Entry Point 44 - Send Message to Specified Task

See M.SMSGR or M_SMSGR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.45 Entry Point 45 - Send Run Request to Specified Task

See M.SRUNR or M_SRUNR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.46 Entry Point 46 - Break/Task Interrupt Link

See M.BRK or M_BRK in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.47 Entry Point 47 - Activate Task Interrupt

See M.INT or M_INT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.48 Entry Point 48 - Exit from Task Interrupt Level

See M.BRKXIT or M_BRKXIT and M.XBRKR or M_XBRKR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.49 Entry Point 49 - Exit Run Receiver

See M.XRUNR or M_XRUNR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.50 Entry Point 50 - Exit from Message End-Action Routine

See M.XMEA or M_XMEA in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.51 Entry Point 51 - Exit from Run Request End-Action Routine

See M.XREA or M_XREA in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.52 Entry Point 52 - RTM CALM Terminate Task Execution

This entry point performs all normal termination functions required of exiting tasks. All devices and memory area are deallocated, related table space is erased, and the task's dispatch queue entry is cleared. If a timer or interrupt level is associated with the task, it is reactivated, connected, and suspended. Resident tasks are merely suspended.

Entry Conditions

Calling Sequence

```
CALM X'55'
M.CALL      H.MONS,52
```

Registers

None

Exit Conditions

Return Sequence

Return to EXEC for sweep of dispatch queue.

Registers

None

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.RTRN

2.53 Entry Point 53 - RTM CALM Activate Task

This entry point is used to activate a task. The task assumes the owner name of the caller.

Entry Conditions

Calling Sequence

CALM X'52'

M.CALL H.MONS,53

Registers

R6,R7 1- to 8-ASCII character left-justified blank-filled program name for which an activation request is to be queued

Exit Conditions

Return Sequence

M.RTRN

Note: If the task being activated is not in the system, an abort indication is not given.

Registers

None

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.RTRN

2.54 Entry Point 54 - RTM CALM Suspend Task Execution

This entry point results in the suspension of the caller or any other task of the same owner name for the specified number of time units or for an indefinite time period, as requested. A task suspended for a time interval results in a one-shot timer entry to resume the task upon time-out of the specified interval. A task suspended for an indefinite time interval must be resumed through the M.SUME or M_SUME system services.

Entry Conditions

Calling Sequence

```
CALM X'54'
M.CALL      H.MONS,54
```

Registers

R7 — zero if suspension for an indefinite time interval is requested, or the negative number of time units to elapse before the caller is resumed

Exit Conditions

Return Sequence

M.RTRN

Registers

None

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.RTRN
M.IOFF
M.IONN
M.OPEN
M.CALL

2.55 Entry Point 55 - RTM CALM Allocate File or Device

This entry point dynamically allocates a peripheral device, a permanent disk file, a temporary disk file, or an SLO or SBO file, and creates a file assignment table (FAT) entry for the allocated unit and specified logical file code (LFC). This entry point may also be used to equate a new LFC with an existing LFC.

Entry Conditions

Calling Sequence

```
CALM X'40'
M.CALL    H.MONS,55
```

Registers

R1 — denial return address

R5 — byte 0 – function code as follows:

- 1 = assign LFC to a user or system permanent file
- 2 = assign LFC to a system file code
- 3 = assign LFC to a peripheral device
- 4 = assign LFC to a defined LFC

bytes 1,2,3 – file code to be assigned

Exit Conditions

Return Sequence

M.RTRN	condition code 1 is set in the program status doubleword if the calling task has read but not write access rights to the specified permanent file
M.RTNA 1	for denial returns if the requested file or device cannot be allocated
M.RTNA 2	condition code 2 is set in the program status doubleword if the calling task does not have read or write access rights to the specified permanent file

Registers

None

Abort Cases

None

Output Messages

None

2.56 Entry Point 56 - RTM CALM Physical Device Inquiry

This entry point returns to the caller physical device information describing the unit to which a specified logical file code is assigned.

Entry Conditions

Calling Sequence

CALM X'42'

M.CALL H.MONS,56

Registers

R5 3-character logical file code for which physical device information is requested in bytes 1, 2, and 3

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 zero, if the specified logical file code is unassigned

M.RTRN 5,6,7

Registers

R5 disk = number of 192-word blocks in file
magnetic tape = reel identifier
all other devices = 0

R6 bytes 0,1 — maximum number of bytes transferrable to device
bytes 2,3 — device mnemonic (2 ASCII characters)

R7 bits 0-5 — device type code
bits 6-15 — device address
bits 16-23 — system file codes as follows:

Value	Definition
0	not a system file
1	SYC file
2	SGO file
3	SLO file
4	SBO file

R7 bits 24-31 — disk = number of 192-word blocks per allocation unit; magnetic tape = volume number (0 if single volume); all other devices = 0

Note: If the file is an SYC or SGO file that is not open, bits 13 through 15 of R7 are returned equal to 1 or 2. All other result bits are not applicable.

Abort Cases

None

Output Messages

None

External Reference

System Macro

M.RTRN

2.57 Entry Point 57 - Disable Message Task Interrupt

See M.DSMI or M_DSMI in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.58 Entry Point 58 - Enable Message Task Interrupt

See M.ENMI or M_ENMI in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.59 Entry Point 59 - Get Physical Memory Contents

See H.REXS,59 for a detailed description of this entry point.

2.60 Entry Point 60 - Change Physical Memory Contents

See H.REXS,60 for a detailed description of this entry point.

2.61 Entry Point 61 - RTM CALM Permanent File Log

This entry point provides a log of currently existing permanent files.

Entry Conditions

Calling Sequence

CALM X'73'

M.CALL H.MONS,61

Registers

R4 contains a byte-scaled value which specifies the type of log to be performed as follows:

Value	Definition
0	specifies a single named system or user file
1	specifies all permanent user files
2	specifies system files only
3	specifies user files
4	specifies a single named system file

Note: If R4 contains zero and a user name is associated with the calling program, an attempt is made to locate the user file directory entry for the given file name. If unsuccessful, the system file directory entry is located, if any. If a user name is not associated with the calling program, the file is assumed to be a system file.

If R4 contains three and the calling program has an associated user name, that user's files are logged or all files are logged if the calling program has no associated user name.

R5 contains the address of an 8-word area where file directory entry is to be stored

Exit Conditions

Return Sequence

M.RTRN 4,5

Registers

R4 if R4 contains zero or four, R4 is destroyed. If R4 contains one, two, or three, this entry point is called repeatedly to obtain all the pertinent file definitions. The type parameter in R4 is specified in the first call only. R4 is returned containing the address of the next directory entry to be returned. The value returned in R4 must be unchanged upon the subsequent call to this service.

R5 contains zero if R4 contains zero or four and the specified file could not be located, or R4 contains one, two, or three, and all pertinent files have been logged. Otherwise, R5 is unchanged.

Abort Cases

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 SPECE.

Output Messages

None

External Reference

System Macro

M.CALL
M.RTRN

2.62 Entry Point 62 - Resourcemark Lock

See M.RSML or M_RSML in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.63 Entry Point 63 - Resourcemark Unlock

See M.RSMU or M_RSMU in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.64 Entry Point 64 - Remove RSM Lock on Task Termination

See H.REXS,64 for a detailed description of this entry point.

2.65 Entry Point 65 - Task CPU Execution Time

See M.XTIME or M_XTIME in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.IOFF
M.IONN

2.66 Entry Point 66 - Activate Program at Given Time of Day

See M.TURNON or M_TURNON in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Service

H.REXS,4

2.67 Entry Point 67 - Set Synchronous Task Interrupt

See M.SYNCH or M_SYNCH in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.68 Entry Point 68 - Set Asynchronous Task Interrupt

See M.ASYNCH or M_ASYNCH in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.69 Entry Point 69 - Reserved

2.70 Entry Point 70 - Date and Time Inquiry

See M.DATE or M_DATE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.OPEN
M.SHUT

2.71 Entry Point 71 - Get Device Mnemonic or Type Code

See M.DEVID or M_DEVID in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.72 Entry Point 72 - Enable User Break Interrupt

See M.ENUB or M_ENUB in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.73 Entry Point 73 - Disable User Break Interrupt

See M.DSUB or M_DSUB in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.74 Entry Point 99 - SYSGEN Initialization

This entry point is for internal use only and is called during SYSGEN. H.MONS sets up its entry point table, then returns to SYSGEN.

Multivolume Magnetic Tape (H.MVMT)

MPX-32 Technical Manual

Volume II

Contents

1 H.MVMT Overview

1.1 General Information

The Multivolume Magnetic Tape Module (H.MVMT) performs all data management operations for multivolume magnetic tape requests.

H.MVMT also recognizes the MPX-32 revision that is being used by the source system. If the system is MPX-32 revision 3.3 or later, bit 2 of DFT.FLGS is set.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.MVOP	N/A	predevice access processing
H.MVPX	N/A	postdevice access processing

N/A implies called only by IOCS

2 H.MVMT Entry Points

2.1 Entry Point H.MVOP - Predevice Access Processing

This entry point performs predevice access processing for multivolume magnetic tape requests.

Entry Conditions

Calling Sequence

```
BL      H.MVOP
```

Registers

R1 FCB address

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1 FCB address

2.2 Entry Point H.MVPX - Postdevice Access Processing

This entry point performs postdevice access processing related to multivolume magnetic tape requests.

Entry Conditions

Calling Sequence

```
BL      H.MVPX
```

Registers

R1 FCB address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1 FCB address

Program Trace (H.PTRAC)

MPX-32 Technical Manual

Volume II

Contents

List of Figures

1 H.PTRAC Overview

1.1 General Information

H.PTRAC is an optional module. It provides:

- an interface between the DBX debugger and MPX-32
- debugger execution control of the target task

Within this section of the manual, the word debugger refers to the DBX debugger or a user-supplied DBX-styled debugger. The words target task refer to the task being debugged.

The H.PTRAC module is included in the system by the SYSGEN PROGRAM or USERPROG directives in the /TRAP section.

1.2 Debugger and Target Task Interaction

The debugger and the target task are individual tasks; the debugger is not mapped into the target task's address space. The two tasks — the debugger and the target — are activated in parallel; however, they do not execute at the same time. One of them is always in the debug-wait state, and execution is interleaved between them through H.PTRAC.

Through H.PTRAC, the debugger can read and write to the target task's memory, and control the target task's execution. Debugger access to the target task's memory is done in the context of the target task. When the target task reaches an appropriate break point, the task's context is relayed to the debugger through H.PTRAC. This allows debugging of:

- tasks that are swapped out of memory and
- tasks whose size plus the debugger's size is greater than the available memory.

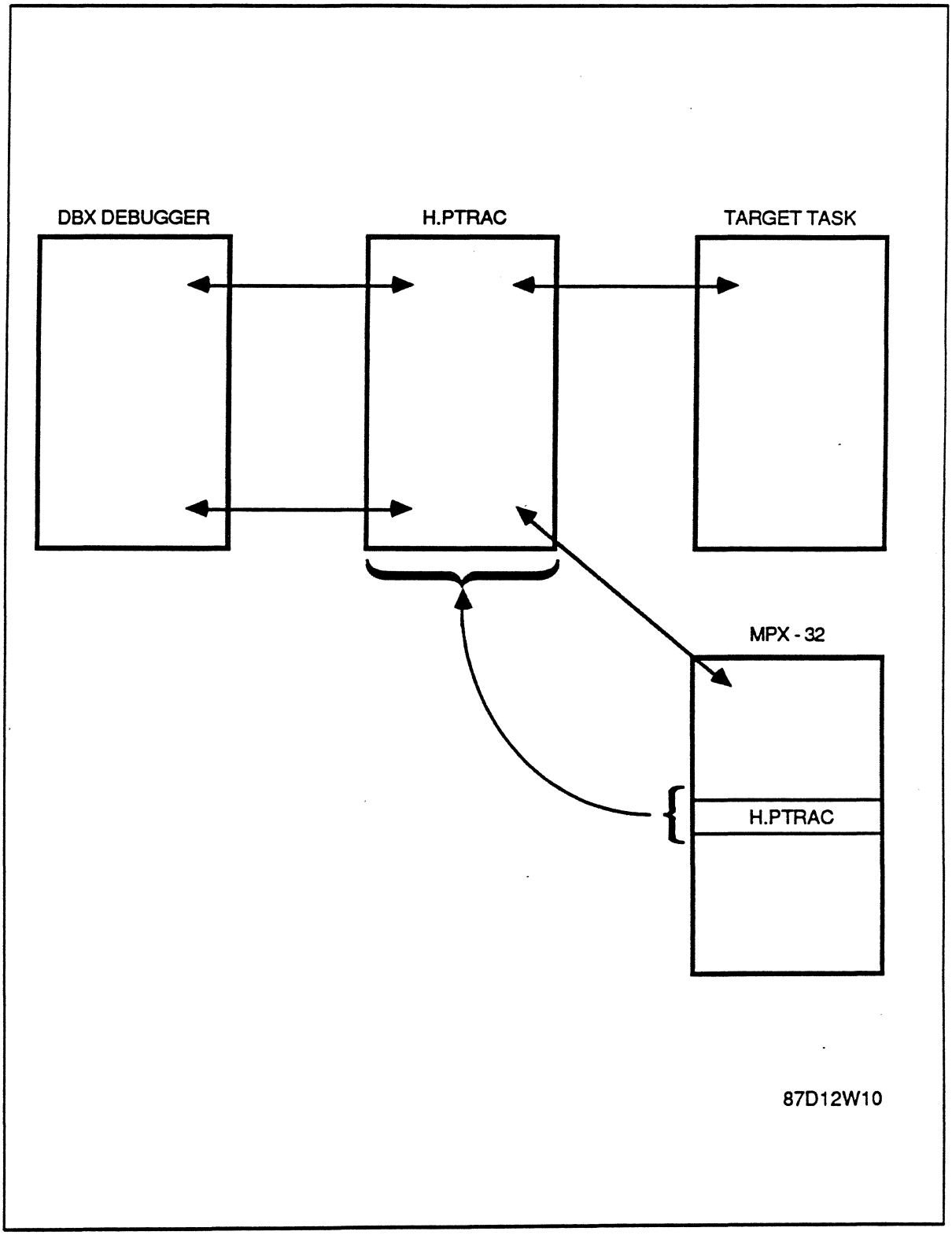


Figure 1-1
H.PTRAC - DBX Interface and Target Task Execution Control

Resource Management (H.REMM)

MPX-32 Technical Manual

Volume II

Contents

Contents

1 H.REMM Overview

1.1 General Information

The Resource Management Module (H.REMM) allocates and assigns all system resources. These functions maintain proper access compatibility and usage rights for resources. This function also coordinates concurrent access to shared resources. This module can reside in extended memory.

Note: Many H.REMM services are alternate entry points and subroutines to the H.TAMM and H.MEMM modules. These services are documented under the appropriate H.TAMM or H.MEMM entry point or subroutine.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.REMM,6	N/A	assign and allocate resource
H.REMM,7	N/A	deassign and deallocate resource
H.REMM,13		reserved
H.REMM,17	49*	mount volume
H.REMM,18		reserved
H.REMM,19	4A*	dismount volume
H.REMM,21	42*	open resource
H.REMM,22	43*	close resource
H.REMM,23	44*	set exclusive resource lock
H.REMM,24	45*	release exclusive resource lock
H.REMM,25	46*	set synchronous resource lock
H.REMM,26	47*	release synchronous resource lock
H.REMM,27	48*	resource inquiry
H.REMM,99	N/A	SYSGEN initialization

* This is SVC 2,X'*nn*' callable. All others are SVC 1,X'*nn*' callable.

N/A implies reserved for internal use by MPX-32.

1.3 Alternate Entry Points

The following are alternate entry points to H.MEMM or H.TAMM entry points. For alternate entry point information, refer to the H.TAMM or H.MEMM entry point listed below.

Alternate Entry Point	Entry Point	SVC Number	Description
H.REMM,1	H.TAMM,2	N/A	construct TSA and DQE
H.REMM,2	H.TAMM,3	N/A	task activation processing
H.REMM,3	H.TAMM,4	N/A	task exit processing
H.REMM,4	H.MEMM,1	N/A	allocate memory
H.REMM,5	H.MEMM,2	N/A	deallocate memory
H.REMM,8	H.MEMM,3	69	get dynamic extended data space
H.REMM,9	H.MEMM,4	6A	free dynamic extended indexed data space
H.REMM,10	H.MEMM,5	67	get dynamic task execution space
H.REMM,11	H.MEMM,6	68	free dynamic task execution space
H.REMM,12	H.MEMM,7	40*	include memory partition
H.REMM,14	H.MEMM,8	41*	exclude memory partition
H.REMM,15	H.MEMM,9	N/A	reserved
H.REMM,16	H.MEMM,10	N/A	reserved
H.REMM,20	H.MEMM,11	N/A	deallocate memory due to swapping
H.REMM,28	H.MEMM,12	4B*	get memory in byte increments
H.REMM,29	H.MEMM,13	4C*	free memory in byte increments

* This sevice is SVC 2,X'*nn*' callable. All others are SVC 1,X'*nn*' callable.

N/A implies reserved for internal use by MPX-32.

1.4 Subroutine Summary

Subroutine	Description
S.REMM4	issue dismount request
S.REMM5	issue mount request
S.REMM6	deallocate all assigned resources
S.REMM6A	deallocate all assigned resources
S.REMM7	find next matching UDT
S.REMM8	deallocate FPT/FAT buffer
S.REMM9	check for resource allocation
S.REMM10	locate FPT/FAT/SMT with preprocessing
S.REMM11	allocate blocking buffer
S.REMM12	locate allocated FPT/FAT
S.REMM14	allocate FPT/FAT
S.REMM23	find associated MVT entry
S.REMM24	uncompress file name
S.REMM25	set any bit in memory

Subroutine	Description
S.REMM26	clear any bit in memory
S.REMM27	test any bit in memory
S.REMM36	check resource compatibility
S.REMM37	caller notification packet M.RTRN processing
S.REMM38	queue for system resource
S.REMM42	gate system prior to ART access
S.REMM43	ungate system after ART access
S.REMM44	check for self-generated resource conflict
S.REMM45	deallocate blocking buffers from the TSA
S.REMM46	allocate a blocking buffer head cell from the TSA
S.REMM47	complete pending dismount of public volume

1.5 Alternate Subroutine Summary

The following are alternate subroutines to H.TAMM or H.MEMM subroutines. For alternate subroutine information, refer to the H.TAMM or H.MEMM subroutine listed below.

Alternate Subroutine	Subroutine	Description
S.REMM1	S.TAMM1	read and verify preamble
S.REMM2	S.TAMM2	deallocate TSA and DQE
S.REMM3	S.MEMM1	reserved
S.REMM13	S.MEMM2	locate shared memory table entry
S.REMM17	S.MEMM5	update map segment descriptor for memory increase
S.REMM19	S.MEMM7	re-map user address space
S.REMM20	S.MEMM8	validate buffer address
S.REMM21	S.MEMM9	allocate memory pool buffer
S.REMM22	S.MEMM10	release memory pool buffer
S.REMM28	S.MEMM11	deallocate debugger memory
S.REMM29	S.TAMM4	load debug overlay
S.REMM30	S.MEMM12	create a protection image
S.REMM31	S.MEMM13	reserved
S.REMM32	S.MEMM14	update shared memory protection image
S.REMM33	S.MEMM15	reserved
S.REMM34	S.MEMM16	get system buffer space
S.REMM35	S.MEMM17	free system buffer space
S.REMM39	S.MEMM18	get map image information
S.REMM40	S.MEMM19	update map segment descriptor for memory decrease
S.REMM41	S.MEMM20	check new task size

2 H.REMM Entry Points

2.1 Entry Point 6 - Assign and Allocate Resource

This entry point assigns and allocates a resource in the manner indicated by the RRS entry supplied as an argument. Compatibility checking is performed to insure proper resource integrity during its allocation.

This entry point is for privileged users only. Nonprivileged users must use the M.ASSN or M_ASSIGN system service described in the MPX-32 Reference Manual Volume I.

Entry Conditions

Calling Sequence

```
M.CALL      H.REMM,6
```

Registers

R1	address of an RRS (type 1 – 6)
R7	address of an CNP or zero

Exit Conditions

Return Sequence with CNP

```
M.RTRN      R5
(or)
M.RTNA      R5 (CC1 set)
```

Return Sequence without CNP

```
M.RTRN      R5
(or)
M.RTRN      R5,R7 (CC1 set)
```

Registers

R5	allocation index or zero
R7	status if an error and CNP is not supplied

Status

CC1 set	Error Code	Definition
	1	unable to locate resource
	2	specified access mode not allowed
	3	FAT/FPT space not availa ble
	4	blocking buffer space not available
	5	shared memory table (SMT) entry not found
	6	volume assignment table (VAT) not available
	7	static assignment to dynam ic common
	8	unrecoverable I/O error to volume
	9	invalid usage specification
	11	invalid RRS entry
	12	LFC logically equated to unassigned LFC
	13	assigned device not in system
	14	resource already allocated by requesting task
	15	SGO or SYC assignment by real-time task
	16	shared memory conflicts with task's address space
	17	duplicate LFC assignment attempted
	18	invalid device specification
	19	invalid resource ID
	20	specified volume not assigned or access not allowed
	22	resource is marked for deletion
	23	assigned device is marked off-line
	24	segment definition allocation by unprivileged task
	25	random access not allowed for this access mode
	26	user attempting to open SYC file in a write mode
	27	resource already opened in a different access mode
	28	invalid access specification at open
	29	unassigned LFC in FCB or invalid FCB address
	38	time out occurred while waiting for resource to become available
	46	unable to obtain resource descriptor lock - multiport only
	50	resource is exclusively locked by another task
	51	shareable resource is allocated in an incompatible access mode
	52	volume space is not available
	53	assigned device is not available
	54	unable to allocate resource for specified usage
	55	allocated resource table (ART) space not available
	56	reserved

Note: Status values 25-29 are returned only when auto-open is indicated.

Scratchpad Usage

T.SPAD1-2	directory back link for pathname
T.SPAD3	MVTE (volume resource) or UDT (device) address
T.SPAD4	used to build first word of FAT
T.SPAD5	used to build FAT access value
T.SPAD6	used for temporary storage
T.SPAD7	FPT address for error cleanup
T.SPAD8	temporary file resource descriptor block for error cleanup
T.SPAD9-20	used to build new RRS for temporary SGO, SLO, SBO assignment from TSA information
T.SPAD21	used by S.REMM7 and S.REMM36

2.2 Entry Point 7 - Deassign and Deallocate Resource

This entry point performs the deassignment of a resource by detaching it from the associated system structures and deallocating its related TSA structures.

This entry point is for privileged users only. Nonprivileged users must use the M.DASN or M_DEASSIGN system service described in the MPX-32 Reference Manual Volume I.

Entry Conditions

Calling Sequence

M.CALL H.REMM,7

Registers

R1	32-bit allocation index or a FCB address
R7	address of a CNP or zero

Exit Conditions

Return Sequence with CNP

M.RTRN
(or)
M.RTNA (CC1 set)

Return Sequence without CNP

M.RTRN
(or)
M.RTRN R7 (CC1 set)

Registers

R7 status if an error and CNP not supplied

Status

CC1 set

Error Code	Definition
8	unrecoverable I/O error
29	LFC not assigned
30	invalid allocation index
46	unable to obtain resource descriptor lock (multiprocessor only)

Scratchpad Usage

T.SPAD1-2 used to build RID for resource deletion
T.SPAD3-8 used by S.TSM4 (spooled files only)
T.SPAD1-19 used by S.REMM4 (unformatted media only)

2.3 Entry Point 13 - Reserved

2.4 Entry Point 17 - Mount Volume

See M.MOUNT or M_MOUNT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point in reference to wait mode mount requests.

To request a mount in no-wait mode, the following conditions must be met:

- bit 12 of RR.GPTS must be set to indicate no-wait mode mount
- the task must be privileged
- a CNP must be supplied and it must contain the PSB address in CP.FCBA
- an end-action address must be supplied in CP.NADDR
- if a MUTE address is to be returned, bit 0 of CP.OPTS must be set.

2.5 Entry Point 18 - Reserved

2.6 Entry Point 19 - Dismount Volume

See M.DMOUNT or M_DISMOUNT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.7 Entry Point 21 - Open Resource

See M.OPENR or M_OPENR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.8 Entry Point 22 - Close Resource

See M.CLOSER or M_CLOSER in the MPX-32 Reference Manual I for a detailed description of this entry point.

2.9 Entry Point 23 - Set Exclusive Resource Lock

See M.LOCK or M_LOCK in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.10 Entry Point 24 - Release Exclusive Resource Lock

See M.UNLOCK or M_UNLOCK in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.11 Entry Point 25 - Set Synchronous Resource Lock

See M.SETSYNC or M_SETSYNC in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.12 Entry Point 26 - Release Synchronous Resource Lock

See M.UNSYNC or M_UNSYNC in the MPX-32 Reference Manual I for a detailed description of this entry point.

2.13 Entry Point 27 - Resource Inquiry

See M.INQUIRY or M _INQUIRER in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.14 Entry Point 99 - SYSGEN Initialization

This entry point is for internal use only and is called during SYSGEN. H.REMM sets up its entry point table, then returns to SYSGEN.

3 H.REMM Subroutines

3.1 Subroutine S.REMM4 - Issue Dismount Request

This subroutine issues a no-wait run request to J.MOUNT for the purpose of performing a physical dismount of a formatted volume or issuing a dismount message to the operator for an unformatted volume. Operator response is not required.

Entry Conditions

Calling Sequence

BL	S.REMM4

Registers

R3	FAT address

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2,R6,R7	destroyed
R3	FAT address

Status

CC1 set	run request failed; no message issued

Scratchpad Usage

T.SPAD1-8	used to build PSB for run request

3.2 Subroutine S.REMM5 - Issue Mount Request

This subroutine issues a mount request for an unformatted medium or a formatted volume by invoking J.MOUNT.

Entry Conditions

Calling Sequence

BL S.REMM5

Registers

R1 FCB address (for an unformatted medium) or a RRS address with bit 0 set (for a formatted volume)

T.R7 CNP address or zero

Exit Conditions

Return Sequence

TRSW R0

Registers

R1 FCB or RRS address

R2,R3 destroyed

R4 mount device type code (for an unformatted medium) or undefined (for a formatted volume)

R5-R7 destroyed

Status

CC1 set

Error Code	Definition
8	unrecoverable I/O error (J.MOUNT)
20	unable to initialize volume (J.MOUNT)
21	J.MOUNT run request failed
42	abort request
43	hold request
	Device reallocation status can also be returned when hold is requested.

Scratchpad Usage

T.SPAD1-8 used to build PSB for J.MOUNT and to construct RRS for device reallocation when hold is requested

3.3 Subroutine S.REMM6 - Deallocate All Assigned Resources

This subroutine deallocates all assigned FPT/FAT's.

Entry Conditions

Calling Sequence

```
BL      S.REMM6
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R2	current stack frame pointer
R3	TSA address
R4-R7	destroyed

Status

None

Scratchpad Usage

T.SPAD22	used to save caller's register one

3.4 Subroutine S.REMM6A - Deallocate All Assigned Resources

This subroutine deallocates all mount assignments in the volume assignment table (VAT).

Entry Conditions

Calling Sequence

```
BL      S.REMM6A
```

Registers

None

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2	current stack frame pointer
R3	TSA Address
R4-R5	unchanged
R6-R7	destroyed

Status

None

Scratchpad Usage

T.SPAD22	used to save caller's register one

3.5 Subroutine S.REMM7 - Find Next Matching UDT

This subroutine locates the next UDT whose device address matches that portion of the supplied device specification indicated by the mask in R4. CC1 is set to indicate an unsuccessful scan, and the resulting status is returned in R7.

Entry Conditions

Calling Sequence

BL	S.REMM7

Registers

R2	UDT address from last call
R4	device mask
R6	device specification as follows:

Byte	Definition
1	device type code
2	logical channel number
3	logical subchannel number

Exit Conditions

Return Sequence

TRSW R0

Registers

R2	UDT address of next matching device (if successful); otherwise, R2 is unchanged
R4	device mask
R5	destroyed
R6	device specification
R7	status if an error; otherwise, R7 is destroyed

Status

CC1 set

Error Code	Definition
13	device not in system
14	device already allocated by caller
23	device is marked off-line
50	device is exclusively locked

Scratchpad Usage

T.SPAD20	used to save the last matching UDT
T.SPAD21	used to save caller's R1
T.SPAD22	used to store initial UDT address

3.6 Subroutine S.REMM8 - Deallocate FPT/FAT Buffer

This subroutine deallocates the specified FPT and FAT. The assignment count in the associated VAT is updated and, if extendible, the segment definition area is marked available.

Entry Conditions

Calling Sequence

BL S.REMM8

Registers

R2 FPT address

Exit Conditions

Return Sequence

TRSW R0

Registers

R2	FPT address
R3	FAT address
R4,R5	destroyed

Status

None

Scratchpad Usage

None

3.7 Subroutine S.REMM9 - Check for Resource Allocation

This subroutine checks the allocated resource table (ART) to determine if the specified resource is currently allocated. If a match is found, CC2 is set and the address of the associated ART entry is returned. If a match is not found, the address of the next available ART entry or zero is returned depending on whether or not any ART entries are available.

Entry Conditions

Calling Sequence

BL S.REMM9

Assumptions

Context switching is inhibited.

Registers

R2 search function code. Byte 0 contains:

Bit	Meaning if Set
4	segment definition (AR.SPACE)
5	partition (AR.PART)
6	device (AR.DEVC)
	zero for all other resources

Bytes 1, 2 and 3 contain zero.

R4 resource allocation key with following byte significance:

Volume Resource

Byte 0 contains the UDT index of the volume on which the resource resides

Bytes 1, 2 and 3 contain the absolute block address of the resource descriptor

Partition

Byte 0 contains the associated SMT index.

Bytes 1, 2 and 3 contain the associated SMT entry address.

Device

Bytes 0 and 1 contain the associated UDT index. Bytes 2 and 3 are zero.

Exit Conditions

Return Sequence

TRSW R0

Registers

R2 address of the associated ART entry if found, or the next available entry if not found, or zero if not found and ART is full

R4 resource allocation key

R5,R6 destroyed

R7 status if an error; otherwise, R7 is destroyed

Status

CC1 set — error (status in R7)

CC2 set — resource allocated:

Error Code	Definition
55	ART is full

Scratchpad Usage

T.SPAD22 — used for temporary storage

3.8 Subroutine S.REMM10 - Locate FPT/FAT/SMT

This subroutine preprocesses the search for an allocated FPT/FAT or partition along with clearing CC1 in the current stack frame.

Entry Conditions

Calling Sequence

BL — S.REMM10

Registers

R5 — 32-bit allocation index or a FCB address

Exit Conditions

Return Sequence

TRSW — R0 (with S.REMM12)

Registers

R1 — current stack frame address

R2 — FPT address (volume resource) or destroyed (partition)

R3 — FAT address (volume resource) or ART address (partition)

R4 — destroyed

R5 — destroyed if allocation index supplied; otherwise, R5 is unchanged (volume resource or partition)

R7 — status if an error; otherwise, R7 is unchanged

Status

CC1 set

Error Code	Definition
29	LFC not assigned
30	invalid allocation index

CC2 set resource is a memory partition

Scratchpad Usage

None

3.9 Subroutine S.REMM11 - Allocate Blocking Buffer

This subroutine allocates a free blocking buffer for the caller. The control word in the blocking buffer is cleared and the buffer empty bit is set. The buffer is marked allocated. The blocking buffer address is inserted in the FAT and the blocking buffer active bit is set in the status word. If the FAT address provided is the system FAT, the system blocking buffer is unconditionally allocated.

Entry Conditions

Calling Sequence

BL S.REMM11

Registers

R3 FAT address

Exit Conditions

Return Sequence

TRSW R0

Registers

CC1 set no blocking buffer found (R5=0)
R3 FAT address
R4 destroyed
R5 blocking buffer address (0 if no allocation)

3.10 Subroutine S.REMM12 - Locate Allocated FPT/FAT

This subroutine locates the FPT/FAT pair associated with a given logical file code (LFC) or allocation index.

Entry Conditions

Calling Sequence

BL	S.REMM12

Registers

R5	32-bit allocation index, or right-justified LFC (bit 8 set indicates the system FPT/FAT)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2	FPT address
R3	FAT address
R4	destroyed
R5	destroyed if allocation index supplied; otherwise, R5 is unchanged
R7	status if an error; otherwise, R7 is unchanged

Status

CC1 set

Error Code	Definition
29	LFC not assigned
30	invalid allocation index

Scratchpad Usage

None

3.11 Subroutine S.REMM14 - Allocate FPT/FAT

This subroutine locates an available FPT/FAT pair and initializes the FPT with the supplied logical file code. The FAT is zeroed and the FPT linked to the FAT. Unsuccessful completion is indicated by condition code settings. If bit 0 is set in register five, a search is made for an associated pseudo-FAT in place of the FPT/FAT allocation. This FAT was allocated to the task when the temporary file was created.

Entry Conditions

Calling Sequence

```
BL      S.REMM14
```

Registers

R1 resource descriptor address if bit 0 set in R5

R5 logical file code (right-justified) or zero:

Bit	Meaning if Set
0	find pseudo-FAT for temporary file
1	deallocate dedicated file system FPT/FAT
8	allocate system FPT/FAT

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R2 FPT address

R3 FAT address

R4,R6 destroyed

R5 logical file code (byte 0 cleared)

R7 status if an error; otherwise, R7 is destroyed

Status

CC1 set

Error Code	Definition
3	FPT/FAT space unavailable
17	duplicate LFC assignment

Scratchpad Usage

None

3.12 Subroutine S.REMM23 - Find Associated MVT Entry

This subroutine locates the mounted volume table (MVT) entry associated with the volume name passed to it as an argument. The address of the MVT entry is returned as the result of the call, if found. Otherwise, an error status is posted.

Entry Conditions

Calling Sequence

BL	S.REMM23

Registers

R1	address of a volume name (left-justified and blank-filled to 16 characters)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1	address of volume name
R2	address of MVT entry, or zero if not found
R4,R6	destroyed
R5	address of the associated VAT entry, or zero if public volume
R7	status if an error; otherwise, R7 is destroyed

Status

CC1 set	volume is not assigned (error code is 20)

Scratchpad Usage

None

3.13 Subroutine S.REMM24 - Uncompress File Name

This subroutine converts 6-bit ASCII coded characters to 8-bit ASCII coded characters. A hexadecimal 20 is added to each 6-bit value to get the new 8-bit value.

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2-R5	destroyed
R6,R7	8-bit ASCII coded name

3.14 Subroutine S.REMM25 - Set Any Bit In Memory

This subroutine sets any bit in memory.

Entry Conditions

Calling Sequence

BL	S.REMM25

Registers

R2	base address of bit string
R4	relative bit number (0-2**20)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R4,R5	destroyed

3.15 Subroutine S.REMM26 - Clear Any Bit In Memory

This subroutine clears any bit in memory.

Entry Conditions

Calling Sequence

BL	S.REMM26

Registers

R2	base address of bit string
R4	relative bit number (0-2**20)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R4,R5	destroyed

3.16 Subroutine S.REMM27 - Test Any Bit In Memory

This subroutine tests the status of any bit in memory.

Entry Conditions

Calling Sequence

BL	S.REMM27

Registers

R2	base address of bit string
R4	relative bit number (0-2**20)

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2	unchanged
R4,R5	destroyed

Status

CC1 set	bit tested is set

3.17 Subroutine S.REMM36 - Check Resource Compatibility

This subroutine determines if the requested allocation access and usage are compatible with the existing allocation status of the resource. If compatible, the allocated resource table (ART) entry is updated appropriately. If not compatible, the task is enqueued or denied as appropriate.

Entry Conditions

Calling Sequence

BL	S.REMM36

Assumptions

Context switching is inhibited.

Registers

R2	ART address
R3	FAT address
T.R7	address of a caller notification packet or zero

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2	ART address
R3	FAT address
R4-R6	destroyed
R7	status if an error or denial; otherwise, R7 is destroyed

Status

CC1 set

Error Code	Definition
38	time out occurred during queue
50	resource is exclusively locked
51	incompatible access mode (implicit shared)
54	incompatible usage mode
56	reserved

Scratchpad Usage

T.SPAD21-22 used for temporary storage

3.18 Subroutine S.REMM37 - Caller Notification Packet

This subroutine performs the error/denial return processing for an entry point called with an optional caller notification packet (CNP) and no return registers.

Entry Conditions

Calling Sequence

BL S.REMM37

Registers

R1 address of current stack frame
R7 error/denial status

Exit Conditions

Return Sequence

Return Sequence with CNP

M.RTRN
M.RTNA

Return Sequence without CNP

M.RTRN R7

Registers

R7 status (if CNP not supplied)

Status

CC1 is always set. Status is the contents of R7.

Scratchpad Usage

None

3.19 Subroutine S.REMM38 - Queue for System Resource

This subroutine places the caller on the general queue for the resource specified by the supplied function code. If a CNP is present and indicates no-wait, the task is not queued.

Entry Conditions

Calling Sequence

BL S.REMM38

Assumptions

Context switching inhibited.

Registers

R3 CNP address or zero

R5 enqueue function code:

Code	Definition
QVRES	volume resource
QART	allocated resource table
QSMT	shared memory table
QMVT	mounted volume table
QSRL	synchronous resource lock
QMNT	volume mount in progress

R6 enqueue ID:

ID	Queuing for
zero	system table space
ART	address volume resource
MVTE	address volume mount

Exit Conditions

Return Sequence

Context switching enabled.

TRSW R0

Registers

R3,R4,R7 destroyed

Status

CC1 set returning due to enqueue time out (status code posted in R7)

CC2 set CNP indicated no queue

Scratchpad Usage

None

3.20 Subroutine S.REMM42 - Gate System Prior to ART Access

This subroutine ensures mutual exclusion whenever the system must read and/or modify an allocated resource table (ART) entry to update the allocation status of a resource. For a multiprocessor, shared volume resource, the resource descriptor (RD) is locked, and the memory-resident ART entry updated with information from the descriptor. Context switching is always inhibited as a result of calling this routine.

Entry Conditions

Calling Sequence

BL S.REMM42

Registers

R2 ART entry address

T.R7 CNP address or zero

Exit Conditions

Return Sequence

Context switching inhibited.

TRSW R0

Registers

R4 destroyed

R7 status, if error; otherwise, R7 is unchanged

Status

CC1 set

Error Code	Definition
46	unable to obtain RD lock (multiport resources only)

Scratchpad Usage

T.SPAD9-10 saves RID for modify RD
T.SPAD11-13 builds CNP
T.SPAD15-22 saves caller's registers

3.21 Subroutine S.REMM43 - Ungate System After ART Access

This subroutine ensures that an allocated resource table (ART) entry is made available to other tasks whenever the system has completed its access to it. If the access was performed on a multiprocessor, shared volume resource, the ART information from the memory-resident ART entry is posted in the resource descriptor (RD), and the RD lock is released. Context switching is always enabled as a result of calling this routine.

Entry Conditions

Calling Sequence

```
BL      S.REMM43
```

Registers

R2 ART entry address

Assumptions

Context switching inhibited. T.RDBUFA contains the appropriate RD (multiprocessor resources only).

Exit Conditions

Return Sequence

Context switching enabled.

```
TRSW    R0
```

Registers

R4 destroyed
R7 status, if error; otherwise, R7 is unchanged

Status

CC1 set

Error Code	Definition
46	unable to release RD lock (multiprocessor resources only)

CC4 set assign count for current port ID equals zero (multiprocessor resources only)

Scratchpad Usage

T.SPAD15-20 saves caller's registers

3.22 Subroutine S.REMM44 - Self-Generated Resource Conflict

This subroutine is called before a task is queued when a required resource is incompatible with the calling task's usage/access rights. The task's FAT and FPT areas are searched for two FATs pointing to the same allocated resource table (ART). If two are found, an error condition is returned to the caller.

Entry Conditions

Calling Sequence

BL S.REMM44

Registers

R2 ART address
R3 FAT address

Exit Conditions

Return Sequence

TRSW R0

Normal Return

R2 ART address
R3 FAT address

Abnormal Return

CC1 set	error
R2	ART address
R3	FAT address
R5	allocation index of previously assigned LFC
R7	abort code

Abort Cases

RM14	RESOURCE ALREADY ALLOCATED BY TASK

3.23 Subroutine S.REMM45 - Deallocate Blocking Buffers

This subroutine deallocates a blocking buffer, or all blocking buffers within a large blocking buffer, and the associated head cell from the task's TSA.

Entry Conditions

Calling Sequence

BL	S.REMM45

Registers

R3	FAT address

Exit Conditions

Return Sequence

TRSW	R0

Registers

R4	destroyed

Status

CC1 set	blocking buffer or any buffer within a large blocking buffer is already deallocated

3.24 Subroutine S.REMM46 - Allocate a Blocking Buffer Head

This subroutine locates a free blocking buffer head cell in the task's TSA and clears the first word to allocate it. The blocking buffer free to allocate bit is reset.

Entry Conditions

Calling Sequence

BL	S.REMM46

Registers

R7	destroyed

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3	head cell address is available; otherwise, R3 is destroyed
R4	destroyed

Status

CC1 set	no free head cells are found

3.25 Subroutine S.REMM47 - Dismount of Volume

This subroutine is called by H.REMM,7 when the last user/assigner on a volume resource deallocates that resource.

The UDT index is used to determine whether the resource is a public volume with pending dismount. If it is, and there are no other ART entries allocated for it, a physical dismount is immediately attempted.

Entry Conditions

Calling Sequence

BL	S.REMM47

Registers

R3	FAT address
R5	UDT index

Exit Conditions

Return Sequence

TRSW	R0

Registers

R4-R7	destroyed

Resident Executive Services (H.REXS)

MPX-32 Technical Manual

Volume II

Contents

Contents

Page

3 H.REXS Subroutines

Contents

1 H.REXS Overview

1.1 General Information

The Resident Executive Services Module (H.REXS) performs resident executive services. After an H.REXS entry point or system subroutine accepts a task request, H.REXS can pass the task request to other resident modules for processing. Resident executive services include timer management, date and time inquiry, conversion routines, and task and controls communications such as task activate, suspend, resume, abort, send, and receive. This module can reside in extended memory.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.REXS,1		reserved
H.REXS,2		reserved
H.REXS,3	44*	memory address inquiry
H.REXS,4	45	create timer entry
H.REXS,5	46	test timer entry
H.REXS,6	47	delete timer entry
H.REXS,7	48	set user status word
H.REXS,8	49	test user status word
H.REXS,9	4A**	change priority level
H.REXS,10	4B	connect task to interrupt
H.REXS,11	4E*	time-of-day inquiry
H.REXS,12	4F	memory dump request
H.REXS,13	50	load overlay segment
H.REXS,14	51	load and execute overlay segment
H.REXS,15	52	activate task
H.REXS,16	53	resume task execution
H.REXS,17	54	suspend task execution
H.REXS,18	55	terminate task execution
H.REXS,19	56	abort specified task
H.REXS,20	57	abort self
H.REXS,21	52***	assign and allocate resource

* This service can be executed by the IPU.

** This service is available to privileged users only.

*** This service is SVC 2,X'*nn*' callable. All others are SVC 1,X'*nn*' callable.

Entry Point Summary

Entry Point	SVC Number	Description
H.REXS,22	53***	deassign and deallocate resource
H.REXS,23	4D*	arithmetic exception inquiry
H.REXS,24	4C*	task option word inquiry
H.REXS,25	58	program hold request
H.REXS,26	60	set user abort receiver address
H.REXS,27	55***	batch job entry
H.REXS,28	62	abort with extended message
H.REXS,29	63	load and execute interactive debugger
H.REXS,30	56***	delete interactive debugger
H.REXS,31	5A	delete task
H.REXS,32	64	get task number
H.REXS,33	59***	validate address range
H.REXS,34		reserved
H.REXS,35	7A	get message parameters
H.REXS,36	7B	get run parameters
H.REXS,37	7C	wait for any no-wait operation complete, message interrupt or break interrupt
H.REXS,38	5D	disconnect task from interrupt
H.REXS,39	5E	exit from message receiver
H.REXS,40	5F**	parameter task activation
H.REXS,41	65	get address limits
H.REXS,42	66	debug link service
H.REXS,43	6B	receive message link address
H.REXS,44	6C	send message to specified task
H.REXS,45	6D	send run request to specified task
H.REXS,46	6E	break/task interrupt link
H.REXS,47	6F	activate task interrupt
H.REXS,48	70	exit from task interrupt level
H.REXS,49	7D	exit run receiver
H.REXS,50	7E	exit from message end-action routine
H.REXS,51	7F	exit from run request end-action routine
H.REXS,52		reserved
H.REXS,53		reserved
H.REXS,54		reserved
H.REXS,55		reserved
H.REXS,56		reserved
H.REXS,57	2E*	disable message task interrupt
H.REXS,58	2F	enable message task interrupt
H.REXS,59	N/A	get physical memory contents
H.REXS,60	N/A	change physical memory contents
H.REXS,61		reserved
H.REXS,62	19	resourcemark lock

* This service can be executed by the IPU.

** This service is available to privileged users only.

*** This service is SVC 2,X'*nn*' callable. All others are SVC 1,X'*nn*' callable.

N/A implies reserved for internal use by MPX-32.

Entry Point	SVC Number	Description
H.REXS,63	1A	resourcemark unlock
H.REXS,64	N/A	remove RSM lock on task termination
H.REXS,65	2D	task CPU execution time
H.REXS,66	1E	activate program at given time of day
H.REXS,67	1B*	set synchronous task interrupt
H.REXS,68	1C	set asynchronous task interrupt
H.REXS,69		reserved
H.REXS,70	15*	date and time inquiry
H.REXS,71	14*	get device mnemonic or type code
H.REXS,72	13*	enable user break interrupt
H.REXS,73	12*	disable user break interrupt
H.REXS,74	50****	acquire current date/time in ASCII format acquire current date/time in binary format acquire current date/time in byte binary format acquire system date/time in any format get current date/time
H.REXS,75	51****	convert ASCII date/time to byte binary format convert ASCII date/time to standard binary convert binary date/time to ASCII format convert binary date/time to byte binary convert byte binary date/time to ASCII convert byte binary date/time to binary convert system date/time format convert time
H.REXS,76	54***	reformat RRS entry
H.REXS,77		reserved
H.REXS,78	57***	reinstate privilege mode to privilege task
H.REXS,79	58***	change task to unprivileged mode
H.REXS,80	7B***	get address limits
H.REXS,81	79***	set exception return address
H.REXS,82	5B***	set IPU bias
H.REXS,83	5C***	set exception handler
H.REXS,84	5D***	get base register task address limits
H.REXS,85	5E***	get task environment
H.REXS,86	5F***	exit with status
H.REXS,87	N/A	load overlay in position
H.REXS,88	61***	get command line
H.REXS,89	62***	move data to the user address
H.REXS,90	0E***	get real physical address

* This service can be executed by the IPU.

*** This service is SVC 2,X'*nn*' callable. All others are SVC 1,X'*nn*' callable.

**** This service is SVC 2,X'*nn*' callable. This service can be executed by the IPU.

N/A implies reserved for internal use by MPX-32.

Entry Point	SVC Number	Description
H.REXS,91	7D****	get TSA address
H.REXS,92	75***	return to the operating system
H.REXS,93	7E****	physical memory read
H.REXS,94	AF****	physical memory write
H.REXS,95	C0****	get task option words 1 and 2
H.REXS,99	N/A	SYSGEN initialization

* This service can be executed by the IPU.

*** This service is SVC 2,X'*nn*' callable. All others are SVC 1,X'*nn*' callable.

**** This service is SVC 2,X'*nn*' callable. This service can be executed by the IPU.

N/A implies reserved for internal use by MPX-32.

1.3 Subroutine Summary

Subroutine	Description
S.REXS1	locate specified task in memory
S.REXS2	delete timers for current task
S.REXS3	get DQE address for specified task
S.REXS4	validate resourcemark index
S.REXS5	get DQE address from task number
S.REXS6	reserved
S.REXS7	zero buffer
S.REXS8	clear scratchpad in current stack frame
S.REXS9	create system pathname in word 10
S.REXS10	test LFC for read only or read/write access
S.REXS11	create system pathname in word 24

2 H.REXS Entry Points

2.1 Entry Point 1 - Reserved

2.2 Entry Point 2 - Reserved

2.3 Entry Point 3 - Memory Address Inquiry

See M.ADRS or M_ADRS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.4 Entry Point 4 - Create Timer Entry

See M.SETT or M_SETT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.5 Entry Point 5 - Test Timer Entry

See M.TSTT or M_TSTT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.6 Entry Point 6 - Delete Timer Entry

See M.DLTT or M_DLTT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.7 Entry Point 7 - Set User Status Word

See M.SETS or M_SETS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.8 Entry Point 8 - Test User Status Word

See M.TSTS or M_TSTS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.OPEN

2.9 Entry Point 9 - Change Priority Level

See M.PRIL or M_PRIL in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.OPEN

2.10 Entry Point 10 - Connect Task to Interrupt

See M.CONN or M_CONN in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.11 Entry Point 11 - Time-of-Day Inquiry

See M.TDAY or M_TDAY in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.12 Entry Point 12 - Memory Dump Request

See M.DUMP or M_DUMP in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.SPAD
M.CALL
M.RTRN

2.13 Entry Point 13 - Load Overlay Segment

See M.OLAY in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL

M.RTRN

2.14 Entry Point 14 - Load and Execute Overlay Segment

See M.OLAY in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL

M.RTRN

2.15 Entry Point 15 - Activate Task

See M.ACTV or M_ACTV in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.16 Entry Point 16 - Resume Task Execution

See M.SUME or M_SUME in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.IOFF
M.RTRN
M.IONN
M.OPEN

2.17 Entry Point 17 - Suspend Task Execution

See M.SUSP or M_SUSP in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.IONN
M.CALL
M.IOFF
M.OPEN

2.18 Entry Point 18 - Terminate Task Execution

See M.EXIT or M_EXIT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.19 Entry Point 19 - Abort Specified Task

See M.BORT or M_BORT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.IOFF
M.IONN
M.OPEN

2.20 Entry Point 20 - Abort Self

See M.BORT or M_BORT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL

2.21 Entry Point 21 - Assign and Allocate Resource

See M.ASSN or M_ASSIGN in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.22 Entry Point 22 - Deassign and Deallocate Resource

See M.DASN or M_DEASSIGN in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.23 Entry Point 23 - Arithmetic Exception Inquiry

See M.TSTE or M_TSTE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.24 Entry Point 24 - Task Option Word Inquiry

See M.PGOW or M_OPTIONWORD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.25 Entry Point 25 - Program Hold Request

See M.HOLD or M_HOLD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.26 Entry Point 26 - Set User Abort Receiver Address

See M.SUAR or M_SUAR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.SPAD
M.CALL
M.RTRN

2.27 Entry Point 27 - Batch Job Entry

See M.BATCH or M_BATCH in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Services

H.VOMM,13
H.REXS,22
H.REXS,45
H.IOCS,23

System Subroutines

S.REXS8
S.VOMM23
S.REMM12
S.REMM37

2.28 Entry Point 28 - Abort with Extended Message

See M.BORT or M_BORT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.IOFF
M.IONN
M.OPEN

2.29 Entry Point 29 - Load and Execute Interactive Debugger

See M.DEBUG or M_DEBUG in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTNA

2.30 Entry Point 30 - Delete Interactive Debugger

This entry point is used only by the interactive debugger to disassociate itself from a task.

Entry Conditions

Calling Sequence

SVC 2, X '56'

or

```
M.CALL      H.REXS,30
```

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.31 Entry Point 31 - Delete Task

See M.DELTSK or M_DELTSK in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.CALL
M.IOFF
M.IONN
M.OPEN

2.32 Entry Point 32 - Get Task Number

See M.ID or M_ID, and M.MYID or M_MYID in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.RTNA

2.33 Entry Point 33 - Validate Address Range

See M.VADDR or M_VADDR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.34 Entry Point 34 - Reserved

2.35 Entry Point 35 - Get Message Parameters

See M.GMSGP or M_GMSGP in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.36 Entry Point 36 - Get Run Parameters

See M.GRUNP or M_GRUNP in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.37 Entry Point 37 - Wait for Any No-Wait Operation Complete

See M.ANYW or M_ANYWAIT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.38 Entry Point 38 - Disconnect Task from Interrupt

See M.DISCON or M_DISCON in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.39 Entry Point 39 - Exit from Message Receiver

See M.XMSGR or M_XMSGR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.40 Entry Point 40 - Parameter Task Activation

See M.PTSK or M_PTSK in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.41 Entry Point 41 - Get Address Limits

See M.GADRL in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.42 Entry Point 42 - Debug Link Service

See the Debug Link system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTNA

2.43 Entry Point 43 - Receive Message Link Address

See M.RCVR or M_RCVR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.44 Entry Point 44 - Send Message to Specified Task

See M.SMSGR or M_SMSGR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.45 Entry Point 45 - Send Run Request to Specified Task

See M.SRUNR or M_SRUNR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.46 Entry Point 46 - Break/Task Interrupt Link

See M.BRK or M_BRK in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.47 Entry Point 47 - Activate Task Interrupt

See M.INT or M_INT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.48 Entry Point 48 - Exit from Task Interrupt Level

See M.BRKXIT or M_BRKXIT and M.XBRKR or M_XBRKR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.49 Entry Point 49 - Exit Run Receiver

See M.XRUNR or M_XRUNR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.50 Entry Point 50 - Exit from Message End-Action Routine

See M.XMEA or M_XMEA in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.51 Entry Point 51 - Exit from Run Request End-Action Routine

See M.XREA or M_XREA in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.52 Entry Point 52 - Reserved

2.53 Entry Point 53 - Reserved

2.54 Entry Point 54 - Reserved

2.55 Entry Point 55 - Reserved

2.56 Entry Point 56 - Reserved

2.57 Entry Point 57 - Disable Message Task Interrupt

See M.DSMI or M_DSMI in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.58 Entry Point 58 - Enable Message Task Interrupt

See M.ENMI or M_ENMI in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.59 Entry Point 59 - Get Physical Memory Contents

This entry point forces the specified physical address to an 8-word boundary and returns the memory contents of that 8-word block to the callers 8-word buffer area, which must be on an 8-word boundary.

Entry Conditions

Calling Sequence

M.CALL H.REXS,59

Registers

R1	physical address of memory
R2	caller's buffer address, must be on an 8-word boundary

Exit Conditions

Return Sequence

M.RTRN

Registers

None

Abort Cases

None

Output Messages

None

2.60 Entry Point 60 - Change Physical Memory Contents

This entry point stores a given value at the physical address specified by the caller.

Entry Conditions

Calling Sequence

M.CALL H.REXS,60

Registers

R1	physical address to change
R7	value to be stored

Exit Conditions

Return Sequence

M.RTRN

Registers

None

Abort Cases

None

Output Messages

None

2.61 Entry Point 61 - Reserved

2.62 Entry Point 62 - Resourcemark Lock

See M.RSML or M_RSML in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.63 Entry Point 63 - Resourcemark Unlock

See M.RSMU or M_RSMU in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.64 Entry Point 64 - Remove RSM Lock on Task Termination

This entry point searches the resourcemark table for the calling task's program number. If found, locks belonging to the task are cleared, the task is dequeued, and the lock is given to the next task waiting for that resource.

Entry Conditions

Calling Sequence

M.CALL H.REXS,64

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

None

Output Messages

None

External Reference

System Services

H.EXEC,36

2.65 Entry Point 65 - Task CPU Execution Time

See M.XTIME or M_XTIME in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN
M.IOFF
M.IONN

2.66 Entry Point 66 - Activate Program at Given Time of Day

See M.TURNON or M_TURNON in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.CALL
M.RTRN

System Services

H.REXS,4

2.67 Entry Point 67 - Set Synchronous Task Interrupt

See M.SYNCH or M_SYNCH in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.68 Entry Point 68 - Set Asynchronous Task Interrupt

See M.ASYNCH or M_ASYNCH in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.69 Entry Point 69 - Reserved

2.70 Entry Point 70 - Date and Time Inquiry

See M.DATE or M_DATE in the MPX-32 Reference Manual Volume I for a detailed description of this entry.

External Reference

System Macro

M.RTRN

2.71 Entry Point 71 - Get Device Mnemonic or Type Code

See M.DEVID or M_DEVID in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.72 Entry Point 72 - Enable User Break Interrupt

See M.ENUB or M_ENUB in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.73 Entry Point 73 - Disable User Break Interrupt

See M.DSUB or M_DSUB in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.74 Entry Point 74 - Acquire Date/Time Services

See the MPX-32 Reference Manual Volume I for a detailed description of the following services:

M.BBTIM	acquire current date/time in byte binary format
M.BTIM	acquire current date/time in binary format
M.GTIM	acquire system date/time in any format
M.QATIM	acquire current date/time in ASCII format
M _GETTIME	get current date/time

External Reference

System Macro

M.RTRN

2.75 Entry Point 75 - Conversion Services

See the MPX-32 Reference Manual Volume I for a detailed description of the following services:

M.CONABB	convert ASCII date/time to byte binary format
M.CONASB	convert ASCII date/time to standard binary
M.CONBAF	convert binary date/time to ASCII format
M.CONBBA	convert byte binary date/time to ASCII
M.CONBBY	convert binary date/time to byte binary
M.CONBYB	convert byte binary date/time to binary
M.CTIM	convert system date/time format
M _CONVERTTIME	convert time

External Reference

System Macro

M.RTRN

2.76 Entry Point 76 - Reformat RRS Entry

See M.NEWRRS in the MPX-32 Reference Manual Volume for a detailed description of this entry point.

2.77 Entry Point 77 - Reserved

2.78 Entry Point 78 - Reinstate Privilege Mode to Privilege Task

See M.PRIV or M_PRIVMODE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.79 Entry Point 79 - Change Task to Unprivileged Mode

See M.UPRIV or M_UNPRIVMODE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.80 Entry Point 80 - Get Address Limits

See M.GADRL2 in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External Reference

System Macro

M.RTRN

2.81 Entry Point 81 - Set Exception Return Address

See M_SETERA in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.82 Entry Point 82 - Set IPU Bias

See M.IPUBS or M_IPUBS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.83 Entry Point 83 - Set Exception Handler

See M_SETEXA in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.84 Entry Point 84 - Get Base Register Task Address Limits

See M_LIMITS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.85 Entry Point 85 - Get Task Environment

See M.ENVRMT or M_ENVRMT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.86 Entry Point 86 - Exit With Status

See M_EXTSTS in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.87 Entry Point 87 - Load Overlay in Position

This entry point enables the symbolic debugger to load or load and execute an overlay at an address other than T.BIAS. The entry point specifies a load address and whether to execute the overlay after loading. The named segment to overlay must be previously defined to the Cataloger as a multiple file overlay.

Entry Conditions

Calling Sequence

M.CALL H.REXS,87

Registers

R5 byte 0 contains the following flags:

Bits	Definition
0	set indicates load and execute overlay
1	set indicates ignore TSA overlay index table and overlay in separate file format
2-7	reserved

bytes 1 through 3 specify the logical load address for the overlay

R6,R7 overlay name (1- to 8-ASCII character, left-justified, blank-filled system file name)

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 unchanged, if the load and execute option was specified. Otherwise, contains transfer address of the overlay segment.

Abort Cases

RX07,LD01-08	CANNOT LOAD OVERLAY SEGMENT DUE TO SOFTWARE CHECKSUM OR DATA ERROR LD*nn* code indicates specific failure.
RX10	OVERLAY HAS AN INVALID PREAMBLE
RX11	AN UNRECOVERALBE I/O ERROR HAS OCCURRED DURING OVERLAY LOADING
RX33	OVERLAY LINKAGES HAVE BEEN DESTROYED BY LOADING A LARGER OVERLAY

Output Messages

None

2.88 Entry Point 88 - Get Command Line

See M.CMD or M_CMD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.89 Entry Point 89 - Move Data to the User Address

See M.MOVE or M_MOVE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.90 Entry Point 90 - Get Real Physical Address

See M.RADDR or M_RADDR in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.91 Entry Point 91 - Get TSA Start Address

See M.GTSAD or M_GTSAD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.92 Entry Point 92 - Return to the Operating System

See M.RTRNOS or M_RTRNOS in the MPX-32 Technical Manual Volume I for a detailed description of this entry point.

2.93 Entry Point 93 - Physical Memory Read

See M.OSREAD or M_OSREAD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.94 Entry Point 94 - Physical Memory Write

See M.OSWRIT or M_OSWRIT in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.95 Entry Point 95 - Get Task Option Word 1 and 2

See M.PGOD or M_OPTIONDWORD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.96 Entry Point 99 - SYSGEN Initialization

This entry point is for internal use only and is called during SYSGEN. H.REXS sets up its entry point table, then returns to SYSGEN.

3 H.REXS Subroutines

3.1 Subroutine S.REXS1 - Locate Specified Task in Memory

This subroutine locates the specified task in memory, then returns its dispatch queue entry (DQE) address in R3 and its task number in R7. A task number must be supplied as input for tasks that are multicopied or shared.

Entry Conditions

Calling Sequence

BL	S.REXS1

Registers

R6,R7	1- to 8-ASCII character task name, left-justified, blank-filled
(or)	
R6,R7	zero if current task
(or)	
R6	zero
R7	task number

Exit Conditions

Return Sequence

Context switch inhibited (M.SHUT) if specified task is not the current task.

TRSW	R0	if an error
TRSW	R0+1W	if successful

Registers

R1,R5	saved
R2,R4,R6	destroyed
R3	DQE address
R7	task number

3.2 Subroutine S.REXS2 - Delete Timers for Current Task

This subroutine searches the timer table for a timer that is allocated to the calling task. If a match is found, the timer is marked available.

Entry Conditions

Calling Sequence

```
BL      S.REXS2
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1,R5, R6,R7	saved
R2,R4	destroyed
R3	DQE address

3.3 Subroutine S.REXS3 - Get DQE Address for Specified Task

This subroutine is used to determine if a specified task is currently activated. If the task is not already activated, an attempt is made to activate it. Once the task is activated, it is linked to the suspend state queue.

Entry Conditions

Calling Sequence

```
BL      S.REXS3
```

Registers

R6,R7	1- to 8-ASCII character task name, left-justified, blank-filled
(or)	
R6,R7	zero if current task
(or)	
R6	zero
R7	task number
(or)	
R6	pathname vector or RID vector
R7	zero

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

Context switch inhibited (M.SHUT) if specified task is not the current task.

R1,R4,R5	saved
R6,R7	saved
R2	destroyed
R3	DQE address or zero if task not found

3.4 Subroutine S.REXS4 - Validate Resourcemark Index

This subroutine validates a resourcemark for a nonprivileged caller.

Entry Conditions

Calling Sequence

BL S.REXS4

Registers

R6 resourcemark index

Exit Conditions

Return Sequence

TRSW R0 if successful

Registers

R7 zero

X1 address of byte in resourcemark table

(or)

Return Sequence

M.RTRN R7 if invalid index

Registers

R7 = 1 range exceeded
= 2 less than minimum range

3.5 Subroutine S.REXS5 - Get DQE Address from Task Number

This subroutine returns the DQE address of the task identified by the task number supplied in R7.

Entry Conditions

Calling Sequence

BL S.REXS5

Registers

R7 task number

Exit Conditions

Return Sequence

Context switch inhibited (M.SHUT) if specified task is not the current task.

TRSW	R0	if an error
TRSW	R0+1W	if successful

Registers

R2,R6,R7	destroyed
R3	DQE address

3.6 Subroutine S.REXS6 - Reserved

3.7 Subroutine S.REXS7 - Zero Buffer

This subroutine is used to zero any contiguous memory buffer in word increments.

Entry Conditions

Calling Sequence

BL	S.REXS7

Registers

R2	address of buffer to zero
R7	number of words to zero

Exit Conditions

Return Sequence

TRSW	R0

Registers

R2,R7	destroyed

3.8 Subroutine S.REXS8 - Clear Scratchpad in Current Stack Frame

This subroutine determines the location of the current push stack level in the caller's TSA. All scratchpad words, except for words 0 through 9 which are the register save area and the return PSD, are then zeroed. All condition codes located in scratchpad word 8, which is the first word of the return PSD, are also cleared.

Entry Conditions

Calling Sequence

```
BL      S.REXS8
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

```
R2,R7   destroyed
```

Status

All condition codes are cleared.

3.9 Subroutine S.REXS9 - Create System Pathname in Word 10

This subroutine is used to create a system pathname starting at scratchpad word ten in the current stack. The pathname is built using the filename supplied in scratchpad words six and seven in the current stack.

Entry Conditions

Calling Sequence

```
BL      S.REXS9
```

Registers

None

Scratchpad Usage

Words 6 and 7 must contain the file (partition) name.

Exit Conditions

Return Sequence

TRSW R0

Registers

R1	stack frame pointer
R3	TSA address
R6,R7	file (partition) name

Scratchpad Usage

Words 10 through 17 contain the 8-word pathname.

3.10 Subroutine S.REXS10 - Test LFC Read Only or Read/Write Access

This subroutine searches the file pointer table (FPT) entries in the caller's TSA searching for an LFC to match the LFC supplied as input. Once a match is found in the FPT, the file assignment table (FAT) is searched to determine if the disk file associated with the specified LFC is read only or read/write restricted.

Entry Conditions

Calling Sequence

BL S.REXS10

Registers

R5	bytes 1 through 3 contain the left-justified, blank-filled, 1 to 3 ASCII character LFC

Exit Conditions

Return Sequence

TRSW R0

Registers

R3	FAT address or zero if an error
R4,R7	destroyed
R5	bit 0 set if read only bytes 1 through 3 unchanged if LFC found; otherwise, R5 is zero

3.11 Subroutine S.REXS11 - Create System Pathname

This subroutine is used to create a system pathname starting at scratchpad word 24 in the current stack. The pathname is built using the file name supplied in scratchpad words 6 and 7 in the current stack.

Entry Conditions

Calling Sequence

```
BL      S.REXS11
```

Registers

None

Scratchpad Usage

Words 6 and 7 must contain the file (partition) name.

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1	stack frame pointer
R3	TSA address
R6,R7	file (partition) name

Scratchpad Usage

Words 24 to 31 contain the 8-word pathname.

Optimized Context Switch Time Module (H.SURE)

MPX-32 Technical Manual

Volume II

Contents

1 H.SURE Overview

1.1 General Information

The Optimized Context Switch Time Module (H.SURE) performs a fast context switch by suspending the calling task and resuming the target task in a single SVC. This service applies to real time and time distribution tasks in base or nonbase mode. The tasks must be CPU only or unbiased. This service is not recommended for two IPU biased tasks.

Optimum performance for the service is achieved when all of the following conditions are met and no errors are encountered:

- the target task is not outswapped
- the target task is of equal or higher priority than the calling task
- the target task is in the suspend state
- the target task is a real time task
- no owner/access violations occur
- in IPU configurations, at least one task is CPU only or unbiased and eligible to run in the CPU at the time the service is called (i.e., the IPU is idle or the task in the IPU is of higher priority then the target task)
- no outstanding system action or task interrupt requests are present (i.e., ABORT, HOLD, DELETE, etc.)
- real time task accounting is turned off (using the OFRA option to the SYSGEN or OPCOM MODE directives or the CATALOG ENVIRONMENT directive)

The H.SURE module is entered unmapped, retaining current maps, and with interrupts blocked. The code sequences supporting the optimized case are sequential such that no branches are taken to break the CPU pipeline. When an unoptimized case is encountered, the map registers are reloaded and standard MPX-32 services (S.EXEC14 and S.EXEC5) are used. The optimized case extracts the physical address of the TSA for both the calling task and the target task and executes a LPSDCM to the target task.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.SURE	00***	Suspend calling task and resume target task

*** This service is SVC 5, X'*nn*' callable.

2 H.SURE Entry Points

2.1 Entry Point H.SURE - Suspend Calling Task and Resume Target Task

Refer to M.SURE or M_SURE in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

Task Management (H.TAMM)

MPX-32 Technical Manual

Volume II

Contents

1 H.TAMM Overview

1.1 General Information

The Task Management Module (H.TAMM) builds and activates new tasks.

1.2 Entry Point Summary

Entry Point	Description
H.TAMM,1	load task into memory
H.TAMM,2	construct TSA and DQE
H.TAMM,3	task activation processing
H.TAMM,4	task exit processing
H.TAMM,5	load base task into memory

1.3 Subroutine Summary

Subroutine	Description
S.TAMM1	read and verify preamble
S.TAMM2	deallocate TSA and DQE
S.TAMM4	load debug overlay
S.TAMM5	calculate the nonbase mode task size
S.TAMM6	calculate the base mode task size
S.TAMM7	set VOMM stack and buffers FPT and FAT in the TSA
S.TAMM9	change the EXTDMPX environment
S.TAMM10	move H.TAMM's working map block
S.TAMM11	activate the program trace debugger

2 H.TAMM Entry Points

2.1 Entry Point 1 - Load Task Into Memory

This entry point loads a task into memory based on information contained in the load module preamble. The task service area (TSA) scratchpad area contains the preamble load information.

Entry Conditions

Calling Sequence

M.CALL H.TAMM,1

Registers

None

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 loading status as follows:

Value	Definition
0	successful loading
1	CSECT loading error
2	CSECT checksum error
3	CSECT relocation matrix loading error
4	CSECT relocation matrix checksum error
5	DSECT loading error
6	DSECT checksum error
7	DSECT relocation matrix loading error
8	DSECT relocation matrix checksum error

2.2 Entry Point 2 - Construct TSA and DQE

This entry point initializes a primitive TSA and DQE for task activation. This is achieved in the following manner:

- Assign the load module with PN vector, PNB vector, or RID vector to support multisegmented load modules.
- Open the load module.
- Read the load module's preamble.
- Scan active DQEs and identify single-copy load module with RID.
- Initialize the child's DQE and save the load module's RID in the child's DQE.
- Link the child's DQE to the preactivation chain.
- Initialize the child's TSA.
- Save the load module's FAT, segment definition, and static resource requirement in the child's task through the child's TSA.
- Set the child's task to execute phase two, H.TAMM3.
- Return to caller.

Special Cases:

- Any load module starting with the letters 'SYSG' are treated as the SYSGEN task, which requires special loading.
- Setting bit 0 of word 0 in the parameter block indicates passing of line buffer to child task (for ICS only).

Entry Conditions

Calling Sequence

```
M.CALL  H.TAMM,2
```

Registers

R1	address of parameter block, or zero if none
R2	pathname, pathname block, or resource ID vector

Exit Conditions

Return Sequence

M.RTRN R6,R7

Registers

R6 return status as follows:

Value	Definition
0	successful preactivation
1	attempt to multicopy unique load module
2	load module not found
3	unable to allocate load module
4	resource is not a load module
5	no more DQE's available
6	I/O error on resource descriptor or device
7	I/O error on reading load module preamble
8	insufficient logical/physical address space for task activation

R7 DQE address of new task or zero

Scratchpad Usage

T.SPAD1	used to hold new TSA address
T.SPAD2	used to hold MIDL counter
T.SPAD3-11	used by S.REMM1
T.SPAD13-16	used by S.REMM1
T.SPAD22	used to save original MSD count

2.3 Entry Point 3 - Task Activation Processing

This entry point is entered on behalf of a new task being activated. It performs all necessary functions to complete the introduction of the new task to the system. This is accomplished by the following sequence:

- Build a temporary FPT/FAT and VOMM stack in the TSA.
- Build a temporary load module FCB and FPT.
- Read the Catalog RRS if an RRS is indicated in the load module's preamble.
- Read the overlay index table if one is indicated in the load module's preamble.
- Save the overlay index table into the TSA.
- Initialize T.IDXA and set byte 0 to indicate the number of single-file format overlays.
- Build the TSA.
- Build a permanent load module FPT and FAT.
- Assign all required resources.
- Load the code/data section using the task loader, H.TAMM1.
- Load the task debugger, if required, using the debugger loader.
- Dispatch control to the user's task.

Note: The common error code return paths for the resource manager are found in this entry point.

Entry Conditions

Calling Sequence

Entered by pop of TSA stack built by H.TAMM,2.

Registers

All registers are zero.

Exit Conditions

Return Sequence

Dispatch to transfer address or to H.REXS,20 with abort code in R5.

Scratchpad Usage

T.SPAD1-2	used for temporary storage
T.SPAD4-8	used to build CNP
T.SPAD9-16	used to reformat old RRS entry
T.SPAD17-20	used for temporary storage

2.4 Entry Point 4 - Task Exit Processing

This entry point performs task exit processing. The abort code, if any, is output. The task clean-up includes the deallocation of all peripherals, volume space, memory and memory pool. Finally the TSA and DQE are deallocated and a return is made to the scheduler with S.EXEC20.

Entry Conditions

Calling Sequence

M.CALL H.TAMM,4

Registers

None

Exit Conditions

Return Sequence

BU S.EXEC20 (CPU scheduler routine)

2.5 Entry Point 5 - Load Base Task Into Memory

This entry point loads a base image (task or shared image) into memory based on information contained in the load module preamble. The task service area (TSA) spad area and T.LDATTR contain the preamble load information.

Entry Conditions

Calling Sequence

M.CALL H.TAMM,5

Registers

R0 bit 0 set: indicates read/write writeback section is used, therefore no checksum for that section

bit 0 not set: checksum all sections

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 loading status as follows:

Value	Definition
0	successful loading
1	CSECT loading error
2	CSECT checksum error
3	CSECT relocation matrix loading error
4	CSECT relocation matrix checksum error
5	DSECT loading error
6	DSECT checksum error
7	DSECT relocation matrix loading error
8	DSECT relocation matrix checksum error

3 H.TAMM Subroutines

3.1 Subroutine S.TAMM1 - Read and Verify Preamble

This subroutine reads the preamble of a load module into the system buffer. The preamble is then verified for integrity. If error conditions exist, status is returned and CC1 is set. This subroutine is used for activation and overlay functions.

Entry Conditions

Calling Sequence

BL S.TAMM1

Registers

R2 pathname, pathname block, or resource ID vector

Exit Conditions

Return Sequence

TRSW R0

Registers

R1 address of preamble (T.BBUFA)
R2 current register pointer (T.REGP)
R3 address of TSA (C.REGS)
R4-R6 destroyed
R7 error status; otherwise, R7 is destroyed

Status

CC1 set
R7 loading status as follows:

Value	Definition
0	successful completion
2	load module not found
3	unable to allocate load module
4	invalid preamble (not a load module)
6	I/O error on resource descriptor/device
7	I/O error on resource

CC2 set shared image resource

Scratchpad Usage

T.SPAD3-8	used to build RRS entry
T.SPAD9-11	used to build a CNP for load module allocation
T.SPAD13-14	used to save resource owner name
T.SPAD15-16	used to save load module name

3.2 Subroutine S.TAMM2 - Deallocate TSA and DQE

This subroutine deallocates all TSA map blocks and updates the memory allocation table. It also clears the tasks DQE and relinks it to the DQE free list.

Entry Conditions

Calling Sequence

BL S.TAMM2

Registers

None

Exit Conditions

Return Sequence

Return to S.EXEC20

Registers

None

3.3 Subroutine S.TAMM4 - Load Debug Overlay

This subroutine performs all memory management and set up requirements for loading the debug overlay. The user's context is copied to T.CONTXT prior to dispatching control to the debug overlay. DQE.ADM, DQE.DBAT, and T.DBHAT are all initialized. T.CSOR points to the start of the debug overlay.

Entry Conditions

Calling Sequence

BL S.TAMM4

Registers

None

Note: This subroutine is called by H.TSM,6, H.REXS,29, or H.TAMM,3.

Exit Conditions

Return Sequence

TRSW R0

Registers

R1-R6 destroyed

R7 transfer address of debug overlay or error status as follows:

Value	Definition
2	debugger load module not found
4	unable to load debugger (inhibit set or sufficient contiguous maps not available)
5	insufficient task space for loading
6	I/O error on resource descriptor
7	I/O error on resource
8	loading error

Status

CC1 set

Scratchpad Usage

T.PSD1	replaced with debugger transfer address
T.SPAD3-11	used by S.TAMM1
T.SPAD12	used to save return address
T.SPAD13-16	used by S.TAMM1
T.SPAD17-22	used to build a pathname vector
T.SPAD1-18	next stack frame used to store preamble data for H.TAMM,1

3.4 Subroutine S.TAMM5 - Calculate Nonbase Mode Task Size

This subroutine calculates the DSECT and CSECT size of the nonbase mode task.

Entry Conditions

Calling Sequence

BL S.TAMM5

Registers

R3 TSA address (C.REGS)

Exit Conditions

Return Sequence

TRSW R0

Registers

R1	unchanged
R2	address of working map (T.WKADR)
R3	TSA address (C.REGS)
R4-R6	destroyed
R7	unchanged

Abort Cases

RM64	DSECT overlaps TSA
RM65	CSECT and DSECT overlap, or task too large for logical address space

3.5 Subroutine S.TAMM6 - Calculate the Base Mode Task Size

This subroutine calculates the size of the base mode task.

Entry Conditions

Calling Sequence

BL S.TAMM6

Registers

None

Exit Conditions

Return Sequence

TRSW R0

Registers

R1	destroyed
R2	unchanged
R3	address of TSA (C.REGS)
R4-R5	destroyed
R6-R7	unchanged

Status

CC1 set	logical address space not available

3.6 Subroutine S.TAMM7 - Set VOMM Stack, FPT, FAT Buffers

This subroutine establishes the temporary VOMM stack, FPT, and FAT areas in the TSA. It also establishes the temporary system buffers in the TSA.

Entry Conditions

Calling Sequence

```
BL      S.TAMM7
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1,R2,R4,R6
destroyed

R3 TSA address (C.REGS)

3.7 Subroutine S.TAMM9 - Change the EXTDMPX Environment

This subroutine changes the EXTDMPX environment when the EXTDMPX partition changes position in the task's logical address space. S.TAMM9 reloads the map registers and revises the contents of T.MPXBR and the current base register stack frame. It also reloads the base registers with new address values.

Entry Conditions

Calling Sequence

```
M.IOFF
BL      S.TAMM9
```

Registers

R0 return address

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R0	return address
R1	destroyed
R2	TSA address
R3-R4	unchanged
R5-R6	destroyed
R7	unchanged

Scratchpad Usage

T.SPAD7-8 used to build PSD

3.8 Subroutine S.TAMM10 - Move Working Map Block

This subroutine moves H.TAMM's working map block within the task's logical address space, and out of the way of the task's address space. This prevents collision between the TSA or other parts of the task being built and H.TAMM's working map block. The working map block is deallocated after the task is completely built and before it is dispatched.

Entry Conditions

Calling Sequence

```
BL      S.TAMM9
```

Registers

R1	target address
R3	TSA address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1	destroyed
R2	current stack address (T.REGP)
R3	TSA address (C.REGS)
R4-R6	destroyed

Status

CC1 set if no logical address space is available

Scratchpad Usage

T.SPAD2 used to save return address

3.9 Subroutine S.TAMM11 - Activate Program Trace Debugger

This subroutine activates the program trace debugger by calling the M.PTSK service. It takes the debugger name from the preamble. The pathname vector and PTASK parameter block are built in the current stack frame SPAD area.

Entry Conditions

Calling Sequence

BL S.TAMM10

Registers

R3 TSA address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1	destroyed
R2	current stack address (T.REGP)
R3	TSA address (C.REGS)
R4-R6	destroyed
R7	status if an error; otherwise, R7 is destroyed

Status

CC1 set The error code is as follows:

Error Code	
2	invalid debugger name
PTASK error code	an error from M.PTSK call

Scratchpad Usage

T.SPAD1-8	used to build debugger pathname
T.SPAD9-21	used to build PTASK parameter block

Terminal Services (H.TSM)

MPX-32 Technical Manual

Volume II

Contents

1 H.TSM Overview

1.1 General Information

The Terminal Services Module (H.TSM) provides services for interfacing TSM terminals to MPX-32.

1.2 H.TSM Entry Point Summary

Entry Point	SVC Number	Description
H.TSM,1	N/A	terminal I/O interface
H.TSM,2	5B	syntax scanner
H.TSM,3	20	TSM task detach
H.TSM,4	N/A	user task abort
H.TSM,5	59	set user tab positions
H.TSM,6	5C	break processing entry
H.TSM,7	28*	convert ASCII decimal to binary
H.TSM,8	29*	convert ASCII hexadecimal to binary
H.TSM,9	2A*	convert binary to ASCII decimal
H.TSM,10	2B*	convert binary to ASCII hexadecimal
H.TSM,11	N/A	relink queued entry by priority
H.TSM,12	N/A	remove and deallocate run request queue entry
H.TSM,13	77	set lower option
H.TSM,14	78	reset lower option
H.TSM,15	7A***	get terminal function definition
H,TSM,16		prepare an SWTI task for execution
H,TSM,17	AE***	TSM procedure call
H,TSM,18		prepare SWSM task for execution
H,TSM,19		activate J.TSM
H.TSM,99	N/A	SYSGEN initialization

* This service can be executed by the IPU.

*** This service is SVC 2,X'*nn*' callable. All others are SVC 1,X'*nn*' callable.

N/A implies reserved for internal use by MPX-32.

1.3 H.TSM Subroutine Summary

Subroutine	Description
S.TSM01	format abort message
S.TSM02	convert binary to ASCII decimal
S.TSM03	report task completion/suspension
S.TSM04	queue real-time SLO/SBO
S.TSM05	build MRRQ and link to J.TSM with no current task

2 H.TSM Entry Points

2.1 Entry Point 1 - Terminal I/O Interface

This entry point performs the special case processing required to support the SYC command files and TSM terminal I/O. H.IOCS transfers to H.TSM for any I/O operation if either the SYC bit or the TSM bit is set in the caller's FAT. H.IOCS performs general validation and set-up of the caller's FCB prior to entering this entry point. H.TSM,1 transfers control to the appropriate I/O routine based on the opcode found in byte 0 of the FCB.

Open Logic

A line buffer is allocated for TSM tasks, and the address of the initialized buffer is placed in T.LINBUF.

Multiple open requests are ignored if T.LINBUF is not equal to zero.

Read Logic

A determination is made whether to read from the terminal or SYC command file.

For terminal I/O, input is buffered through the line buffer to allow the task to be swapped while in SWTI (a prompt is generated if this option is in effect). The actual read request is reissued using the system FPT and FCB. Lower case input is allowed if that option is in effect.

For command line recall and edit terminal reads, a break is sent to J.TSM. J.TSM then performs a read in the same manner as for SYC command file input.

For SYC command file input, a message is sent to J.TSM where the actual read is performed into the caller's line buffer. J.TSM also performs any necessary macro processing and $ detection. If the TEXT option is in effect, the record is echoed to the terminal for interactive tasks or the SLO file for batch tasks.

After I/O is complete, including post I/O processing, the contents of the input buffer are moved back to the caller's buffer. All characters from the carriage return (if any) to the end of the buffer are blank-filled. The transfer count in the FCB is updated to reflect the actual number of characters entered before the carriage return. Finally, the scheduler is informed that terminal input is complete and return is made back to the caller unless a break was detected during the read, then H.MONS,47 is called.

Write Logic

First the line count (T.LINNO) is checked to see if a screen size has been specified. If so, the current position of the cursor is checked against the maximum count. A prompt message is written when the bottom of the screen is reached. After a user response, the count is reset to top of screen. Next, the caller's TCW is clamped with the maximum transfer specified in UDT.CHAR. Break detection is enabled and the scheduler is informed that terminal output is in progress. Finally, the write service is reissued to IOCS with a new FCB (from T.BFCB).

When I/O is complete, the scheduler is informed and a test for break is made. If a break occurred, this service calls H.MONS,47. Otherwise, return is made to the instruction following the original IOCS call.

Close Logic

The line buffer pointed to by T.LINBUF is deallocated by S.REMM22.

Rewind Logic

The cursor position in the line buffer is reset to the first input character.

Entry Conditions

Calling Sequence

M.CALL H.TSM,1

Registers

R1 address of user's FCB

Output Conditions

Updated FCB

External Reference

System Macro

H.IOCS,1
H.IOCS,3
H.IOCS,4
H.IOCS,19
H.IOCS,23
H.EXEC,7
S.REMM22

Terminal Messages

```
ENTER CR FOR MORE
prompt>
```

where *prompt* is the 3-character task abbreviation

2.2 Entry Point 2 - Syntax Scanner

See M.TSCAN in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.3 Entry Point 3 - TSM Task Detach

This entry point is entered by the scheduler as part of the exit process for online or batch tasks. The terminal line buffer is deallocated and an exit message is sent to J.TSM via H.TSM.

This entry point will also detach a user task from TSM if called through SVC 1,X'20'. If this entry point is called from a user task through the SVC, LFC UT and all assignments to LFC UT must be deallocated first.

Entry Conditions

Calling Sequence

M.CALL H.TSM,3

Registers

None

Exit Conditions

Return Sequence

M.RTRN
(or)
M.RTRN and CC1 set if not a TSM task

Registers

None

2.4 Entry Point 4 - User Task Abort

This entry point is entered by the scheduler when an online or batch task aborts. For interactive tasks, the abort code and PSW address are written to the user's terminal. If the terminal is malfunctioning, the abort code is written to the system console device. Then this entry point merges with entry point 3.

Entry Conditions

Calling Sequence

M.CALL H.TSM,4

Registers

None

Exit Conditions

Return Sequence

M.RTRN

Registers

None

2.5 Entry Point 5 - Set User Tab Positions

This entry point is used by the editor to pass the user's specified tab positions to the UDT. The user's tabs are examined by the terminal handler during formatted I/O processing and the cursor is adjusted as appropriate.

Entry Conditions

Calling Sequence

(caller must be a TSM task)

LD	R6, TABS
SVC	1,X'59' (or) M.CALL H.TSM,5
TABS	is a doubleword containing up to eight tab positions. These positions must be in ascending order with zero indicating no more tabs.

Exit Conditions

Return Sequence

M.RTRN

(or)

M.RTRN and CC1 set if no terminal found

Registers

None

2.6 Entry Point 6 - Break Processing Entry

See M.TBRKON in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.7 Entry Point 7 - Convert ASCII Decimal to Binary

See M.CONADB or M_CONADB in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.8 Entry Point 8 - Convert ASCII Hexadecimal to Binary

See M.CONAHB or M_CONAHB in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.9 Entry Point 9 - Convert Binary to ASCII Decimal

See M.CONBAD or M_CONBAD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.10 Entry Point 10 - Convert Binary to ASCII Hexadecimal

See M.CONBAH or M_CONBAH in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.11 Entry Point 11 - Relink Queued Entry by Priority

This entry point is used to relink a queued entry according to its new priority. It is used by the REDIRECT, REMOVE and URGENT directives as well as by J.TSM and J.SOEX.

Entry Conditions

Calling Sequence

```
M.CALL    H.TSM,11
```

Registers

R2	address of MRRQ
R3	head cell address
R7	new priority (zero implies relink to top of queue)

Exit Conditions

Return Sequence

```
M.RTRN
```

Registers

All unchanged

2.12 Entry Point 12 - Remove and Deallocate Run Request Queue Entry

This entry point is used to unlink and deallocate a MRRQ which does not contain call back processing. This entry point is used by J.TSM and J.SOEX.

Entry Conditions

Calling Sequence

M.CALL H.TSM,12

Registers

R2 address of MRRQ

R3 head cell address

Exit Conditions

Return Sequence

M.RTRN

Registers

All registers are unchanged.

2.13 Entry Point 13 - Set Lower Option

See M.SOPL or M_SOPL in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.14 Entry Point 14 - Reset Lower Option

See M.ROPL or M_ROPL in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.15 Entry Point 15 - Get Terminal Function Definition (TERMDEF)

See M.GETDEF in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.16 Entry Point 16 - Prepare SWTI Task for Execution

This entry point removes a task from the SWTI wait state and prepares it for execution. This entry point is called by J.TSM when terminal input for a task is complete.

Entry Conditions

Calling Sequence

M.CALL H.TSM,16

Registers

R2 DQE address of the SWTI task that is to be prepared

Exit Conditions

Return Sequence

M.RTRN

2.17 Entry Point 17 - TSM Procedure Call

See M.TSMPC or M_TSMPC in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

2.18 Entry Point 18 - Prepare SWSM Task for Execution

This entry point removes a task from the SWSM wait state and prepares it for execution. This entry point is called by J.TSM when performing TSM procedure calls (M.TSMPC/M_TSMPC).

Entry Conditions

Calling Sequence

M.CALL H.TSM,18

Registers

R2 DQE address of the SWSM task that is to be prepared

Exit Conditions

Return Sequence

M.RTRN

2.19 Entry Point 19 - Activate J.TSM

This entry point activates J.TSM and is called due to the following conditions:

- If TSM exit is specified in the SYSGEN directives, this entry point is called to activate J.TSM
  - when a user attempts to log on to the system

  or
  - when an independent task performs an M.DEFT or M.BATCH.
- If TSM exit is not specified in the SYSGEN directives, this entry point is called to activate J.TSM by SYSINIT
  - when the system is booted.

This entry point is intended for use only by the MPX-32 operating system.

Entry Conditions

Calling Sequence

```
M.CALL      H.TSM,19
```

Registers

None

Exit Conditions

Return Sequence

M.RTRN

(or)

M.RTRN and CC1 set if M.PTSK error

Registers

R7	status from M.PTSK

2.20 Entry Point 99 - SYSGEN Initialization

There is no SYSGEN initialization required. Return is made with TRSW through R0.

3 H.TSM Subroutines

3.1 Subroutine S.TSM01 - Format Abort Message

This subroutine formats the abort message for terminals and real-time tasks. The task is assumed to be in the abort or delete sequence.

Entry Conditions

Calling Sequence

```
BL      S.TSM01
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R5 abort message length

R6 address of abort message

Message Format

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

3.2 Subroutine S.TSM02 - Convert Binary to ASCII Decimal

This subroutine is functionally the same as H.TSM,9. See M.CONBAD or M_CONBAD in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

3.3 Subroutine S.TSM03 - Report Task Completion/Suspension

This subroutine is called in the exit, abort, delete, or hold sequence by TSM interactive and batch tasks. The routine sends a wait message to its parent, typically J.TSM, passing the IPU and CPU time, abort PSD and bias.

Entry Conditions

Calling Sequence

```
BL      S.TSM03
```

Registers

R5	Value	Meaning
	2	task is terminating
	4	task is entering a hold state

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

All registers are destroyed.

3.4 Subroutine S.TSM04 - Queue Real-Time SLO/SBO

This subroutine is called during the deallocation of real-time SLO or SBO. The routine sends a no-wait run request to J.SOEX. The run request format is described in MPX-32 Reference Manual, Volume I Chapter 2. In addition, the resource descriptor of the file is modified from temporary to spool type.

Entry Conditions

Calling Sequence

```
BL      S.TSM04
```

Registers

R3 FAT address

Exit Conditions

Return Sequence

TRSW R0

Registers

All registers are destroyed.

3.5 Subroutine S.TSM05 - Build MRRQ and Link to J.TSM

This subroutine is called by the operating system when a message is to be displayed on the system console, but there is no current task; therefore, a stack is not available for performing an M.CALL. The routine builds a message request queue in memory pool and links it to the message receiver head cell in J.TSM's DQE. The format of the message request queue is described in MPX-32 Reference Manual Volume I, Chapter 2.

Entry Conditions

Calling Sequence

LA	R1,*addr*
BL	S.TSM05
addr	is the address of a message buffer containing the character count in byte 0

Exit Conditions

Return Sequence

TRSW R0

Abort Cases

CC1 is set if message is not linked.

Volume Management Module (H.VOMM)

MPX-32 Technical Manual

Volume II

Contents

Page

3 H.VOMM Subroutines

1 H.VOMM Overview

1.1 General Information

The Volume Management Module (H.VOMM) is responsible for the dynamic manipulation of the disk data structures in MPX-32. The basic disk volume layout is defined by the Volume Formatter (J.VFMT) when a disk volume is initialized; this layout is verified prior to access by the mount program (J.MOUNT).

H.VOMM provides the following facilities:

- disk resource creation and deletion
- resource logging and location
- file space management (extension and truncation)
- resource attribute modification

H.VOMM deals with memory resident tables (the mounted volume table (MVT)) and disk-resident structures (file space allocation maps, directories and descriptors). The MVT is set up at volume mount time; the disk-resident structures are accessed on an as needed basis. The latter requirement means some H.VOMM entries must obtain dynamic memory on a per entry basis from the callers address space.

The integrity of the space allocation map is crucial in preventing file overlaps. If H.VOMM detects potential overlap areas, H.VOMM halts the system. For further information, refer to the MPX-32 Reference Manual, Volume III, Chapter 6.

H.VOMM helps maintain the integrity of the disk file system space. During execution, H.VOMM inhibits the completion of any task's asynchronous abort or delete requests for all entry points which modify the allocation of disk space. This includes RD and file space. Any request is deferred until leaving the respective entry point.

In addition, H.VOMM helps protect the integrity of critical disk data structures (SMAP, DMAP, RDs, and directories) from I/O errors that may occur during their manipulation. This protection is handled through the MVT, which maintains a count of any such errors. The count is used later, during volume dismount processing, and again, during the subsequent volume mount attempt.

All interlocks within H.VOMM, for example, update access to the file space allocation map, are provided by Resource Management Module (H.REMM) allocation/open calls, thus obtaining exclusive access only to required structures. This gating mechanism allows concurrent H.VOMM activities on a 'per volume' basis.

All entry points within H.VOMM specify their entry conditions by calling S.VOMM28. This routine creates the requested environment for that entry point, thus providing (dynamic) buffers and FCBs as required, and setting up parameters for the generated normal and abnormal exit sequences.

The dynamic memory region is accessed through T.BASEP, which contains the entry depth count in bits 0 through 7 and the address of the memory in bits 8 through 24.

H.VOMM subroutines utilize a register save stack, which is located in the caller's task service area (TSA). The macro M.PUSH saves the caller's registers, returning the address of the pushed registers in R3, and M.POP restores the last-pushed register set. The stack operates from high to low addresses, thus allowing positive displacement access in extended addressing mode. The pointer in the TSA (T.FSSP) is copied into the communication region (C.FSSP) whenever a task is scheduled, and points to the next frame area (each frame is eight words long).

H.VOMM gains access to the various disc structures in one of two ways:

- Short resource ID (SRID) – this mechanism is used to access structures in a file-like manner, and is used for directories and allocation maps.
- Space definition – this mechanism is used to access descriptors.

H.VOMM uses either read mode (for examination) to allow access to other readers, or update mode (for content modification), an exclusive use mode. By convention, H.REMM does not allow access by resource ID (RID) to a structure whose resource descriptor (RD) is held open in update mode by space definition.

If a resource is located on a multiprocessor volume and it is shared on MPX-32 Revision 3.3 or later, then H.VOMM uses the multiprocessor resource descriptor format; otherwise, H.VOMM uses the dual-processor resource descriptor format.

1.2 Entry Point Summary

Entry Point	SVC Number	Description
H.VOMM,1	20*	create permanent file
H.VOMM,2	21*	create temporary file
H.VOMM,3	22*	create memory partition
H.VOMM,4	23*	create directory
H.VOMM,5	24*	delete resource
H.VOMM,6	25*	extend file
H.VOMM,7	26*	truncate file
H.VOMM,8	27*	change defaults
H.VOMM,9	28*	change temporary file to permanent file
H.VOMM,10	29*	log resource or directory
H.VOMM,11	2A*	modify descriptor
H.VOMM,12	2B*	rewrite descriptor
H.VOMM,13	2C*	read descriptor
H.VOMM,14	2D*	rename file
H.VOMM,15	2E*	convert pathname to pathname block
H.VOMM,16	2F*	reconstruct pathname
H.VOMM,17	N/A	allocate resource descriptor
H.VOMM,18	N/A	deallocate resource descriptor
H.VOMM,19	N/A	allocate file space
H.VOMM,20	N/A	deallocate file space
H.VOMM,21	N/A	reserved
H.VOMM,22	N/A	reserved
H.VOMM,23	30*	replace permanent file
H.VOMM,24	N/A	create temporary file

Entry Point	SVC Number	Description
H.VOMM,25	N/A	read/write authorization file
H.VOMM,26	31*	modify descriptor user area
H.VOMM,27	32*	rewrite descriptor user area
H.VOMM,99	N/A	SYSGEN initialization

* This service is SVC 2,X'*nn*' callable. N/A implies reserved for internal use by MPX-32.

1.3 Subroutine Summary

Subroutine	Description
S.VOMM1	get directory entry
S.VOMM2	get next pathname item
S.VOMM3	get next pathname block item
SV1.S30	calculate hash index
S.VOMM4	restore original priority for multiport
S.VOMM5	change priority for multiport
S.VOMM6	read resource descriptor for modification
S.VOMM7	validate privilege
S.VOMM8	get resource descriptor
S.VOMM9	return resource descriptor
S.VOMM10	validate address
S.VOMM11	allocate file space
S.VOMM12	get file space
S.VOMM13	deallocate file space
S.VOMM14	read resource descriptor by resource specification
S.VOMM15	delete directory entry
S.VOMM16	release file space
S.VOMM17	build resource descriptor
S.VOMM18	zero resource space if requested
S.VOMM19	create directory entry
S.VOMM20	return resource ID to caller
S.VOMM21	reserved
S.VOMM22	reserved
S.VOMM23	determine resource specification type
S.VOMM24/25	push and pop
S.VOMM26	assign/open for read only
S.VOMM27	assign/open for read/write
S.VOMM28	create VOMM environment
S.VOMM29	delete VOMM environment
S.VOMM30	write
S.VOMM31	read
S.VOMM32	merge RCB
S.VOMM33	set ART flags
S.VOMM34	increment critical I/O error count in MVT

2 H.VOMM Entry Points

2.1 Entry Point 1 - Create Permanent File

See M.CPERM or M_CREATEP in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

Asynchronous abort and delete are inhibited for execution of this entry point.

External References

System Service

H.VOMM,24

2.2 Entry Point 2 - Create Temporary File

See M.TEMP or M_CREATET in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

Asynchronous abort and delete are inhibited for execution of this entry point.

External References

System Service

H.VOMM,24

2.3 Entry Point 3 - Create Memory Partition

See M.MEM or M_MEM in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

Asynchronous abort and delete are inhibited for execution of this entry point.

External References

System Service

H.VOMM,24

System Subroutine

S.REMM14

2.4 Entry Point 4 - Create Directory

See M.DIR or M_DIR in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

Asynchronous abort and delete are inhibited for execution of this entry point.

External References

System Service

H.VOMM,24

2.5 Entry Point 5 - Delete Resource

See M.DELR or M_DELETER in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

Asynchronous abort and delete are inhibited for execution of this entry point.

External References

System Services

H.VOMM,18
H.VOMM,20

2.6 Entry Point 6 - Extend File

See M.EXTD or M_EXTENDFILE in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

Asynchronous abort and delete are inhibited for exection of this entry point.

External References

System Service

H.VOMM,19

System Subroutine

S.REMM9

2.7 Entry Point 7 - Truncate File

See M.TRNC or M_TRUNCATE in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

Asynchronous abort and delete are inhibited for exection of this entry point.

External References

System Service

H.VOMM,20

Scratchpad Usage

T.SPAD1
T.SPAD2
T.SPAD3

2.8 Entry Point 8 - Change Defaults

See M.DEFT or M_DEFT in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

External References

System Services

H.VOMM,13
H.VOMM,25

2.9 Entry Point 9 - Change Temporary File to Permanent File

See M.TEMPER or M_TEMPFILETOPERM in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

2.10 Entry Point 10 - Log Resource or Directory

See M.LOGR or M_LOGR in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

External References

System Service

H.REMM,7

2.11 Entry Point 11 - Modify Descriptor

See M.MOD or M_MOD in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

External References

System Service

H.REMM,7

2.12 Entry Point 12 - Rewrite Descriptor

See M.REWRIT or M_REWRIT in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

External References

System Service

H.REMM,7

System Subroutine

S.REMM12

2.13 Entry Point 13 - Read Descriptor

See M.LOC or M_READD in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

2.14 Entry Point 14 - Rename File

See M.RENAM or M_RENAME in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

2.15 Entry Point 15 - Convert Pathname to Pathname Block

See M.PNAMB or M_PNAMB in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External References

Scratchpad Usage

T.SPAD2
T.SPAD3

2.16 Entry Point 16 - Reconstruct Pathname

See M.PNAM or M_CONSTRUCTPATH in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

External References

System Service

H.REMM,7

Scratchpad Usage

T.SPAD1
T.SPAD2
T.SPAD3
T.SPAD4

2.17 Entry Point 17 - Allocate Resource Descriptor

See the Allocate Resource Descriptor system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External References

System Service

H.REMM,7

2.18 Entry Point 18 - Deallocate Resource Descriptor

See the Deallocate Resource Descriptor system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External References

System Service

H.REMM,7

2.19 Entry Point 19 - Allocate File Space

See the Allocate File Space system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External References

System Service

H.REMM,7

2.20 Entry Point 20 - Deallocate File Space

See the Deallocate File Space system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External References

System Service

H.REMM,7

2.21 Entry Point 21 - Reserved

2.22 Entry Point 22 - Reserved

2.23 Entry Point 23 - Replace Permanent File

See M.REPLAC or M_REPLACE in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

External References

System Service

H.REMM,7

System Subroutine

S.REMM9

2.24 Entry Point 24 - Create Temporary File

This entry point creates a temporary file for system calls. Resource descriptor for the created file will be in the system buffer.

Entry Conditions

Calling Sequence

M.CALL H.VOMM,24

Registers

R4	start block; or zero
R5	number of blocks required
R6	MVTE or zero

Exit Conditions

Return Sequence

TRSW R0

Registers

Contiguous creation:

R4	start block
R5	size in blocks
R6	MVTE
R7	RD disk address

(or)

Noncontiguous creation:

R4	number of segments created
R5	end of medium block number created
R6	MVTE
R7	RD disk address

CC1 is set if error.

External References

System Services

H.REMM,7
H.VOMM,17
H.VOMM,18
H.VOMM,19
H.REXS,74

2.25 Entry Point 25 - Read/Write Authorization File

See the Read/Write Authorization File system service in the MPX-32 Reference Manual Volume I for a detailed description of this entry point.

External References

System Services

H.MONS,44
H.FISE,8

2.26 Entry Point 26 - Modify Descriptor User Area

See M.MODU or M_MODU in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

2.27 Entry Point 27 - Rewrite Descriptor User Area

See M.REWRTU or M_REWRTU in the MPX-32 Reference Manual, Volume I for a detailed description of this entry point.

2.28 Entry Point 99 - SYSGEN Initialization

This entry point is for internal use only and is called during SYSGEN. H.VOMM sets up its entry point table, then returns to SYSGEN.

3 H.VOMM Subroutines

3.1 Subroutine S.VOMM1 - Get Directory Entry

This subroutine parses a pathname (PN) or a pathname block (PNB) in order to return a directory entry. This subroutine is callable in three different modes to perform three distinct functions as described below.

The lookup mode determines the existence or nonexistence of a given resource.

The create mode is called after the lookup mode to perform hashing of the last item in the PN/PNB into the parent directory, updating hash counts as it goes. The active and free entry counts for the directory are also updated.

The delete mode is called after the lookup mode to verify the existence of the resource to be deleted. It updates the hash counts and entry counts for the directory, but the user must set the delete bit in the entry and rewrite the entry to the directory.

S.VOMM1 requires two registers as input. R1 contains the address of a PN or PNB to be parsed. R7 contains a bit pattern to define the operation to be performed. If the resource is to be looked up, only the directory is opened read only. If the user indicates a create or a delete is to follow, S.VOMM1 opens the directory in the update mode. In any case, the directory is assigned and opened upon return to the caller. The caller must close and deallocate the directory.

S.VOMM1 returns all parameters except the error code in the current scratchpad area. See below for details.

Entry Conditions

Calling Sequence

```
BL      S.VOMM1
```

Registers

R1 address of PB/PNB to parse

R7

Bit	Meaning if Set
29	indicates delete to follow
30	indicates create to follow
31	indicates lookup mode

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1	pathname or pathname block vector
R7	status code

Output format (in scratchpad)

SV1.NAM1	T.SPAD1	name (0-3)
	T.SPAD2	name (4-7)
	T.SPAD3	name (8-11)
	T.SPAD4	name (12-15)
SV1.DNAM	T.SPAD5	directory (0-3)
	T.SPAD6	directory (4-7)
	T.SPAD7	directory (8-11)
	T.SPAD8	directory (12-15)
SV1.RDAD	T.SPAD9	directory resource descriptor address
SV1.MVTE	T.SPAD10	path mounted volume table entry
SV1.DIRI	T.SPAD12	directory Index
SV1.HASH	T.SPAD13	hash count
SV1.FLAG	T.SPAD14	flags
SV1.TYPE	T.SPAD15	type code
SV1.FCB	T.SPAD16	file control block address
SV1.DIRR	T.SPAD17	directory buffer ending address
SV1.DIRE	T.SPAD18	directory size (bytes)
SV1.BLOK	T.SPAD19	directory starting block number

Flag bits are assigned as follows:

Bit	Meaning if Set
24	if set, causes S.VOMM27 to supply a default time-out value when assigning the directory for multiport files (SV1.TMO)
25	log service call (SV1.LOG)
26	volume only scan required (SV1.VOLO)
27	pathname block input (reset for pathname) (SV1.PNB)
28	update access (SV1.CRE + SV1.DELE) (SV1.UPD)
29	delete mode (SV1.DELE)
30	create mode (SV1.CRE)
31	lookup mode (SV1.LOOK)

Internal subroutines used by S.VOMM1:

SV1.S10	get next pathname/pathname block item
SV1.S20	move string (R7) to destination (R3) (16 characters blank-filled if needed)
SV1.S30	see subroutine SV1.S30
SV1.S40	scan directory for matching name
	Input Registers
	R4 0 for scan or delete, or create flag for create
	R5 0 for scan (first scan only)
	1 for create (second scan only)
	-1 for delete (second scan only)
SV1.S50	a second entry into SV1.S30 (Entry into directory open for update. Save registers and jumps in after the open to allow hash setup, etc.)
SV1.S60	find mounted volume table entry (MVTE) for volume
S.VOMM2	see subroutine S.VOMM2
S.VOMM3	see subroutine S.VOMM3

3.2 Subroutine S.VOMM2 - Get Next Pathname Item

This subroutine examines the specified pathname (PN) to return the next item to the caller. Item length and item type are also returned. If an item length exceeds the allowed limit, a syntax error is returned.

The user must update the address of the next unprocessed character in the PN (add the item size) and must not change the number of characters left to process or change the previous item type for repetitive calls.

Entry Conditions

Calling Sequence

BL	S.VOMM2

Registers

R1	address of next unprocessed character in PN
R5	state (0 first call, S.VOMM2 reply subsequent)
R6	pathname vector

Exit Conditions

Return Sequence

TRSW R0

Registers

R1

Value	Definition
0	named item
4	current volume or directory
8	system volume or directory
12	root directory

R5 state (byte 0) and state flags (bytes 1-3)

R6 updated PN vector for next call

R7 replied name vector

3.3 Subroutine S.VOMM3 - Get Next Pathname Block Item

This subroutine examines the specified pathname block (PNB) to return the next item to the caller. Item length and item type are also returned. If a syntax error is detected, the address of the next PNB item points to the beginning of the item being processed when the error was detected.

The user must update the address of the next unprocessed character in the PN (add the item size) and must not change the number of characters left to process or change the previous item type for repetitive calls.

Entry Conditions

Calling Sequence

BL S.VOMM3

Registers

R1 address of next unprocessed character in PNB

R5 state (0 first call, S.VOMM3 reply subsequent)

R6 pathname block vector

Exit Conditions

Return Sequence

TRSW R0

Registers

R1

Value	Definition
0	named item
4	current volume or directory
8	system volume or directory
12	root directory

R5 state (byte 0) and state flags (bytes 1-3)

R6 updated PNB vector for next call

R7 replied name vector

3.4 Subroutine SV1.S30 - Calculate Hash Index

This subroutine hashes the specified 16-character resource or directory name in order to produce an index which will be used to locate a directory entry corresponding to the name. A hash count (indicating the current number of unsuccessful attempts to locate the desired directory entry) is supplied by the caller for the proper hashing algorithm to be determined as follows:

16 Character Name

Word	0	A
	1	B
	2	C
	3	D

Hash Count	Hashing Algorithm
0	*((a+b)/n)*s* = hash count
.	where:
.	*a* = 1 bit circulated to right of word 0
.	+ = exclusive OR
n	b = word 1
	n = total number of entries
	s = entry size 64 bytes
	Words 2 and 3 are ignored

Entry Conditions

Calling Sequence

BL SV1.S30

Registers

R2 stack area address of 16-character name

R4 flag (bit 2 is on for create)

R5

Value	Definition
0	scan (first scan only)
1	create (second scan only)
-1	delete (second scan only)

Exit Conditions

Return Sequence

TRSW R0

3.5 Subroutine S.VOMM4 - Restore Priority for Multiport

This subroutine restores priority and swap status for the multiport environment if S.VOMM5 was called. If S.VOMM5 was not called, this subroutine does not perform any action.

Entry Conditions

Calling Sequence

BL S.VOMM4

Registers

None

Exit Conditions

Return Sequence

TRSW R0

Registers

T.DPINFO (1 word) contains the following:

Bytes 0-2 Priority

Bit	Meaning if set
30	task is swappable (T.SWP)
31	real-time task (T.RT)

3.6 Subroutine S.VOMM5 - Change Priority for Multiport

This subroutine changes priority and locks the resource descriptor for the multiport environment.

Entry Conditions

Calling Sequence

BL S.VOMM5

Registers

R1 FCB address

Exit Conditions

Return Sequence

TRSW R0

Registers

CC1 set error

T.DPINFO (1 word) contains the following:

Bytes 0-2 Priority

Bit	Meaning if set
30	task is swappable (T.SWP)
31	real-time task (T.RT)

3.7 Subroutine S.VOMM6 - Read Resource Descriptor

This subroutine is intended to be called only by H.VOMM,11 (M.MOD or M _MOD) and H.VOMM,26 (M.MODU or M _MODU). Its function is to set up the appropriate VOMM environment dependent on the caller. The resource descriptor is read into the system buffer if the call is from M.MODU or M_MODU; otherwise, the resource descriptor is read into the user buffer.

Entry Conditions

Calling Sequence

```
BL      S.VOMM6
```

T.BIT2 is set in the task service area (TSA) if the call is from H.VOMM,26.

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1	address of FCB that read the resource descriptor
R3	address of buffer containing the resource descriptor

3.8 Subroutine S.VOMM7 - Validate Privilege

This subroutine is comprised of two additional subroutines, S.VOMM7A and S.VOMM7B. These subroutines check to see if the caller has sufficient access to a resource. The test (any bit match) is made against owner, project group and other rights depending on the current caller.

This subroutine can also check to see if a call originated as a system call. In this case, enter with R4 equal to zero. Upon exit, an error (CC1 set) is produced if the call is not a system call.

Entry Conditions

Calling Sequence

BL	S.VOMM7	resource descriptor is in VOMM's dynamic buffer
BL	S.VOMM7A	resource descriptor is in the system buffer
BL	S.VOMM7B	resource descriptor address is in R2

Registers

R4 access request bits as follows:

Value	Description
0	check if call is from system routine or system administrator
-1	check if call is from system routine only

Exit Conditions

Return Sequence

TRSW R0

Registers

CC1 set	no bits in R4 match access bits in the resource descriptor
CC1 reset	any bit in R4 matches access bits in the resource descriptor

3.9 Subroutine S.VOMM8 - Get Resource Descriptor

This subroutine examines the specified resource descriptor allocation map (DMAP) to locate a zero bit corresponding to an available resource descriptor (RD). When a zero bit is located, the bit is set to one, and the DMAP bit number in this segment is returned to the caller. If no RDs are available, condition code one is set prior to returning to the caller. S.VOMM8 assumes the FCB has been assigned and opened prior to being called and contains the DMAP buffer address.

Entry Conditions

Calling Sequence

BL	S.VOMM8

Registers

R1	FCB address
R4	buffer length (bits)

Exit Conditions

Return Sequence

TRSW	R0

Registers

CC1 set	RD unavailable
R4	DMAP bit number in this DMAP segment

3.10 Subroutine S.VOMM9 - Return Resource Descriptor

This subroutine examines the specified resource descriptor allocation map (DMAP) bit number for this DMAP segment to reset the corresponding bit in the DMAP. If the resource descriptor is not currently allocated, condition code 1 is set prior to returning to the caller.

Entry Conditions

Calling Sequence

BL	S.VOMM9

Registers

R1	FCB address
R4	DMAP bit number in this DMAP segment
R6	mounted volume table entry (MVTE) address

Exit Conditions

Return Sequence

TRSW	R0

Registers

CC1 set	RD was not allocated

3.11 Subroutine S.VOMM10 - Validate Address

This subroutine validates the address range input. S.REMM20 is called to check the address range and its response is checked.

Entry Conditions

Calling Sequence

BL	S.VOMM10

Registers

R6	address to be checked
R7	size to check

Exit Conditions

Return Sequence

TRSW	R0

Registers

CC1 set	address range is invalid

All registers are unchanged.

3.12 Subroutine S.VOMM11 - Allocate File Space

This subroutine marks the space map as allocated and writes it to SMAP using the FCB given.

The FCB is expected to contain the address and size of the memory buffer. It is assumed exclusively open on SMAP to provide SMAP lock against other users. The last portion of SMAP is expected to be available in the buffer, with FCB showing values having read that section last.

Entry Conditions

Calling Sequence

BL	S.VOMM11

Registers

R1	FCB address
R4	bit position relative to SMAP start of the last +1 bit to be allocated

Exit Conditions

Return Sequence

TRSW	R0

Registers

CC1 set	write error
R7	contains the error codes as follows:

Value	Definition
17	SMAP write error
52	SMAP bit double-set

3.13 Subroutine S.VOMM12 - Get File Space

This subroutine scans the file space allocation map (SMAP) to locate the requested free allocation units (free allocation units are indicated by zero bit settings). The necessary SMAP file manipulation is provided with the exception of allocation/open processing. A starting allocation unit can be specified. If it is not available, a failure indication is given.

The FCB is expected to contain the address and size of the memory buffer and it is assumed exclusively open on SMAP to provide SMAP lock against other users.

Entry Conditions

Calling Sequence

BL	S.VOMM12

Registers

R1	FCB address
R4	starting bit number of relative SMAP or zero if anywhere
R6	mounted volume table entry (MVTE) address
R7	number of allocation units required

Exit Conditions

Return Sequence

TRSW	R0

Registers

CC1 reset	if space found
R4	contains bit number in SMAP of next bit after section used
R7	unchanged
	Memory buffer holds the last or only portion of SMAP containing the requested free space.
(or)	
CC1 set	space not found
R4,R5	unchanged
R7	contains the error code as follows:

Value	Definition
11	no space available
16	SMAP read error

Internal Subroutines

SV12.100 check space map section for unallocated bit string

Input Registers

R2	address of starting word
R5	bit number in word to start at
R7	allocation unit requested (buffer bound)

Exit Registers

R2	address of next word to process
R5	next allocation unit bit
R7	bits remaining (may be -VE or 0 if allocation unit found)
R3-R6	destroyed

SV12.200 check space map section for allocated bit string

Input Registers

R2	address of starting word
R5	bit number in word to start at
R7	buffer boundary

Exit Registers

R2	address of next word to process
R5	next allocation unit bit
R7	bits remaining (may be -VE or 0 if allocation unit found)
R3-R6	destroyed

SV12.300 scan map buffer for length of zero bits at high end of buffer or zero bit string anywhere of requested size

Input Registers

R1	file control block (FCB) address
R5	number of bits to scan
R7	allocation unit requested (buffer bound)

Exit Registers

CC1 reset	string found
R2	starting bit address of requested string (first occurrence)
R4	length of zero bits at high end of buffer
CC1 set	string not found
R4	length of zero bits at high end of buffer

SV12.400 mark space map section as allocated

Input Registers

R1	file control block (FCB) address
R4	relative bit number from beginning of buffer
R7	number of allocation units

Exit Registers

All registers are preserved.

3.14 Subroutine S.VOMM13 - Deallocate File Space

This subroutine deallocates the space map and writes it to the file space allocation map (SMAP) using FCB #3. FCB #3 is expected to contain the address and size of the memory buffer and is assumed to be exclusively opened on SMAP to provide SMAP lock against other users.

Entry Conditions

Calling Sequence

```
BL      S.VOMM13
```

Registers

R4	bit position relative to SMAP start of the first bit to be deallocated
R7	number of allocation units to be deallocated

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

CC1 set	error
R7	contains the error code as follows:

Value	Description
16	SWAP read error
17	SMAP write error

3.15 Subroutine S.VOMM14 - Read Resource Descriptor

This subroutine reads a resource descriptor given a pathname (PN), pathname block (PNB), resource ID (RID), LFC or FCB as input. If desired, the FCB that reads the resource descriptor is left open. If the read is successful, the mounted volume table Entry (MVTE) address is given.

Entry Conditions

Calling Sequence

BL	S.VOMM14

Registers

R1	PN, PNB, RID, LFC or FCB address
R2	FCB address

Bit	Meaning if set
1	FCB is left assigned and opened read/write. If multiport, resource descriptor is locked and priority is changed to one.
2	directory is assigned and opened read/write
3	resource descriptor is not locked for multiport

Exit Conditions

Return Sequence

TRSW	R0

Registers

R5	type of resource specification
R7	contains MVTE and resource descriptor is in specified FCB buffer
(or)	
CC1 set	error condition
R7	contains the error code

3.16 Subroutine S.VOMM15 - Delete Directory Entry

This subroutine locates and deletes a directory entry given a pathname (PN) or pathname block (PNB) as input.

Entry Conditions

Calling Sequence

BL	S.VOMM15

Registers

R1	PN or PNB address

Exit Conditions

Return Sequence

TRSW	R0

Registers

CC1 set	error conditions
R7	contains error code
(or)	
CC1 reset	delete is successful

3.17 Subroutine S.VOMM16 - Release File Space

This subroutine returns file space to the free allocation area. It truncates a file to EOF or releases all file space as directed by the input parameter.

Entry Conditions

Calling Sequence

BL	S.VOMM16

Registers

R1	resource descriptor address
R6	mounted volume table entry (MVTE) address
R7	specifies operation to take place as follows:

- negative indicates delete file
- positive or 0 indicates truncate

Exit Conditions

Return Sequence

TRSW	R0

Registers

CC1 reset	successful operation. All registers unchanged.
(or)	
CC1 set	error condition
R7	contains an error code

3.18 Subroutine S.VOMM17 - Build Resource Descriptor

This subroutine builds a resource descriptor for a permanent or temporary file in the system buffer. This subroutine expects a resource descriptor for a system temporary file to be present in the VOMM resource descriptor buffer and an appropriate resource create block (RCB) to be present in B.RCB. If these conditions are present, the resource descriptor in the system buffer is modified to reflect the attributes of a specific resource, such as directory, file, and memory partition.

Entry Conditions

Calling Sequence

BL	S.VOMM17

Registers

R7	resource type code

Exit Conditions

Return Sequence

TRSW	R0

Registers

None

3.19 Subroutine S.VOMM18 - Zero Resource Space

This subroutine zeroes the disk space associated with a resource. It assumes the presence of a valid working resource descriptor in B.RDBUF.

Entry Conditions

Calling Sequence

```
BL      S.VOMM18
```

Registers

R6 mounted volume table entry (MVTE) address

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

None

3.20 Subroutine S.VOMM19 - Create Directory Entry

This subroutine processes a pathname (PN) for the purpose of creating a directory entry for the resource defined by the PN and the specified identifier. This subroutine is called when a permanent file, directory, or COMMON is created, a temporary file is made permanent, or files are renamed.

Entry Conditions

Calling Sequence

```
BL      S.VOMM19
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

CC1 set if error return
R7 error status

3.21 Subroutine S.VOMM20 - Return Resource ID to Caller

This subroutine transfers a resource ID (RID) for a resource to the address specified in the incoming resource create block (RCB). The RCB is assumed in the RCB area in the VOMM environment.

Entry Conditions

Calling Sequence

```
BL          S.VOMM20
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW        R0
```

Registers

All registers are unchanged.

3.22 Subroutine S.VOMM21 - Reserved

3.23 Subroutine S.VOMM22 - Reserved

3.24 Subroutine S.VOMM23 - Determine Resource Specification

This subroutine examines the resource specification parameter to determine its format: pathname, pathname block (PNB), resource ID (RID), short RID (SRID), or FCB.

Entry Conditions

Calling Sequence

BL S.VOMM23

Registers

R1 resource specification

Note: An FCB or LFC will have its top byte clear as it is either a G'*xxx*' or an address. It is as convenient to provide an FCB address as to provide the contents of word 0 of an FCB. Other forms of the resource specification contain a byte count in byte 0 followed by the address of the pathname, pathname block, RID, or SRID.

Exit Conditions

Return Sequence

TRSW R0

Registers

R7 resource specification code as follows:

Value	Definition
0	pathname
1	pathname block
2	RID
3	SRID
4	FCB address
5	LFC
6	allocation index

If the reply code is 4 or 5, R1 contains the FPT address of the allocated FCB or LFC.

(or)

CC1 set failure condition (all registers unchanged)

3.25 Subroutine S.VOMM24/25 - Push and Pop

This subroutine is used by M.PUSH and M.POP to push/pop the VOMM stack to save all registers or restore all registers. C.FSSP points to the current stack level pointer and it is decremented by M.PUSH and incremented by M.POP.

Entry Conditions

Calling Sequence

```
BL      S.VOMM24
BL      S.VOMM25
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R3 current stack level pointer

3.26 Subroutine S.VOMM26 - Assign/Open Read Only

This subroutine assigns a resource for read-only operations.

Entry Conditions

Calling Sequence

```
BL      S.VOMM26
```

Registers assignment by SRID

R1 address of FCB to perform request
R5 zero
R6 resource descriptor block number
R7 mounted volume table entry (MVTE) address

Registers assignment by space definition

R1 address of FCB to perform request
R5 UDT index
R6 starting block number
R7 number of blocks

Exit Conditions

Return Sequence

TRSW R0

Registers

R7 unchanged if no abnormal conditions
(or)
CC1 set abnormal conditions detected
R7 contains abnormal condition value as follows:

Value	Definition
44	REMM error occurred

3.27 Subroutine S.VOMM27 - Assign/Open Read Write

This subroutine assigns a resource for read and write operations.

Entry Conditions

Calling Sequence

BL S.VOMM27

Registers assignment by SRID

R1 address of FCB to perform request
R5 zero
R6 resource descriptor block number
R7 mounted volume table entry (MVTE) address. If bit 0 is set then apply default timeout on resource assignment.

Registers assignment by space definition

R1 address of FCB to perform request
R5 UDT index
R6 starting block number
R7 number of blocks

Exit Conditions

Return Sequence

TRSW	R0
R7	unchanged if no abnormal conditions
(or)	
CC1 set	abnormal conditions detected
R7	contains the REMM error number

3.28 Subroutine S.VOMM28 - Create VOMM Environment

This subroutine gets dynamic system memory if the context bit is on in the control word and has not been previously allocated. To save R0, which contains the address of the control word, one level of stack is pushed.

Entry Conditions

Calling Sequence

BL	S.VOMM28

Registers

None

Exit Conditions

Return Sequence

TRSW	R0

Registers

R3	address of caller's TSA
T.BASEP	address of dynamic storage area in TSA
C.FSSP	address of stack in communication region

3.29 Subroutine S.VOMM29 - Delete VOMM Environment

This subroutine releases the service entry push frame, deallocates all FCBs specified at the service entry, and releases dynamic memory.

Entry Conditions

Calling Sequence

```
BL      S.VOMM29
```

Registers

None

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

CC1 set	deallocation error occurs
R1	contains T.REGP
R3	points to the context control word
R5	contains error code

3.30 Subroutine S.VOMM30 - Write

This subroutine issues an IOCS write for the caller. It monitors the I/O status and returns CC1 if an error occurs.

Entry Conditions

Calling Sequence

```
BL      S.VOMM30
```

Registers

R1	address of FCB to perform request

Exit Conditions

Return Sequence

TRSW R0

Registers

R7 unchanged if no abnormal conditions

(or)

CC1 set abnormal conditions detected

R7 contains abnormal condition value as follows:

Value	Definition
1	short transfer
2	unrecoverable I/O error

3.31 Subroutine S.VOMM31 - Read

This subroutine issues an IOCS read for the caller. It monitors the I/O status and returns CC1 if an error occurs.

Entry Conditions

Calling Sequence

BL S.VOMM31

Registers

R1 address of FCB to perform request

Exit Conditions

Return Sequence

TRSW R0

Registers

R7 unchanged if no abnormal conditions

(or)

CC1 set abnormal conditions detected

R7 contains abnormal condition value as follows:

Value	Definition
1	short transfer
2	unrecoverable I/O error
3	end-of-medium detected
4	end-of-file detected

3.32 Subroutine S.VOMM32 - Merge RCB

This subroutine forms an RCB by merging the entry RCB, which may be absent, with default values. Volume name and owner name defaults are applied from the TSA. The RCB is formed in B.RCB in dynamic memory.

Entry Conditions

Calling Sequence

BL S.VOMM32

Registers

R2 users RCB address or zero if not specified

R3 default RCB address

Exit Conditions

Return Sequence

TRSW R0

Registers

CC1 set address error on RCB

R2 address of working RCB in dynamic memory buffer

3.33 Subroutine S.VOMM33 - Set ART Flags

This subroutines sets AR.DELET and/or AR.TRUNC flags in the ART if the resource to be deleted or truncated meets one of the following conditions:

- The resource is currently allocated.
- A multiported resource is assigned by another CPU.

Entry Conditions

Calling Sequence

BL	S.VOMM33

Registers

R1	RD address
R6	MVTE address
R7	negative value indicates delete file; positive or zero indicates truncate file.

Exit Conditions

Return Sequence

TRSW	R0

Registers

CC2 set	AR.DELET or AR.TRUNC is set in the ART

3.34 Subroutine S.VOMM34 - Increment Critical I/O Error Count

This routine is called every time an error code of 12 through 19 is returned. These codes indicate an I/O error has occurred during a read or write of a critical data structure on the disk (directory, RD, SMAP, DMAP).

Entry Conditions

Calling Sequence

BL	S.VOMM34

Registers

R1	FCB address used in read or write attempt

Exit Conditions

Return Sequence

TRSW R0

Registers

All registers are preserved.

A System Macros Cross-Reference

A.1 System Macros Cross-Reference

Macro name	Description	Reference
HMP.INIT	MIOP Initialization	Vol. I, Ch. 1
IB.INIT	MIOP Initialization	Vol. I, Ch. 1
M.ACTV and M_ACTV	Activate Task	H.REXS,15; H.MONS,15
M.ADRS and M_ADRS	Memory Address Inquiry	H.REXS,3; H.MONS,3
M_ADVANCE	Advance Record or File	H.IOCS,7;H.IOCS,8
M.ALOC	Allocate File or Peripheral Device	H.MONS,21
M.ANYW and M_ANYWAIT	Wait for any No-Wait Operation Complete, Message Interrupt or Break	H.REXS,37;H.REMM,6 H.MONS,37
M_ASSIGN and M.ASSN	Assign/Allocate Resource	H.REXS,21; H.REMM,6
M.ASYNCH and M_ASYNCH	Set Asynchronous Task Interrupt	H.REXS,68
M_AWAITACTION	End Action Wait	H.EXEC,40
M.BACK and M_BACKSPACE	Backspace File or Record	H.IOCS,9; H.IOCS,19 Vol. I, Ch. 1
M.BATCH and M_BATCH	Batch Job Entry	H.REXS,27
M.BBTIM and M_BBTIM	Acquire Current Date/Time in Byte Binary Format	H.REXS,74
M.BORT and M_BORT	Abort Specified Task	H.REXS,19; H.MONS,19
	Abort Self	H.REXS,20; H.MONS,20
	Abort with Extended Message	H.REXS,28; H.MONS,28
M.BRK and M_BRK	Break/Task Interrupt Link	H.REXS,46; H.MONS,46
M.BRKXIT and M_BRKXIT	Exit from Task Interrupt Level	H.REXS,48; H.MONS,48
M.BTIM and M_BTIM	Acquire Current Date/Time in Binary Format	H.REXS,74
M.CALL	Call to System Module	Vol. I, Ch. 1
M.CDJS	Submit Job from Disc File	H.MONS,27
M.CLOSER and M_CLOSER	Close Resource	H.REMM,22

Macro name	Description	Reference
M.CLSE and M_CLSE	Close File	H.IOCS,2; H.IOCS,5 H.IOCS,23
M.CMD and M_CMD	Get Command Line	H.REXS,88
M.CONABB and M_CONABB	Convert ASCII Date/Time to Byte Binary Format	H.REXS,75
M.CONADB and M_CONADB	Convert ASCII Decimal to Binary	H.TSM,7
M.CONAHB and M_CONAHB	Convert ASCII Hexadecimal to Binary	H.TSM,8
M.CONASB and M_CONASB	Convert ASCII Date/Time to Standard Binary	H.REXS,75
M.CONBAD and M_CONBAD	Convert Binary to ASCII Decimal	H.TSM,9
CONBAF and M_CONBAF	Convert Binary Date/Time to ASCII Format	H.REXS,75
M.CONBAH and M_CONBAH	Convert Binary to ASCII Hexadecimal	H.TSM,10
M.CONBBA and M_CONBBA	Convert Byte Binary Date/Time to ASCII	H.REXS,75
M.CONBBY and M_CONBBY	Convert Binary Date/Time to Byte Binary	H.REXS,75
M.CONBYB and M_CONBYB	Convert Byte Binary Date/Time to Binary	H.REXS,75
M.CONN and M_CONN	Connect Task to Interrupt	H.REXS,10; H.MONS,10
M_CONSTRUCTPATH	Reconstruct Pathname	H.VOMM,16
M_CONVERTTIME	Convert Time	H.REXS,75
M.CPERM	Create Permanent File	H.VOMM,1
M.CREATE	Create Permanent File	H.FISE,12
M_CREATEP	Create Permanent File	H.VOMM,1
M_CREATET	Create Temporary File	H.VOMM,2; H.VOMM,24
M.CTIM and M_CTIM	Convert System Date/Time Format	H.REXS,75
M.CWAT and M_CWAT	System Console Wait	H.IOCS,26
M.DALC	Deallocate File or Peripheral Device	H.MONS,22
M.DASN	Deassign/Deallocate Resource	H.REXS,22; H.REMM,7

Macro name	Description	Reference
M,DATE and M_DATE	Date and Time Inquiry	H.REXS,70
M_DEASSIGN	Deassign and Deallocate Resource	H.REXS,22;H.REMM,7
M.DEBUG and M_DEBUG	Load and Execute Interactive Debugger	H.REXS,29; H.MONS,29
M.DEFT and M_DEFT	Change Defaults	H.VOMM,8
M.DELETE	Delete Permanent File or Non-SYSGEN Memory Partition	H.FISE,14
M_DELETER and M.DELR	Delete Resource	H.VOMM,5
M.DELTSK and M_DELTSK	Delete Task	H.REXS,31; H.MONS,31
M.DEVID and M_DEVID	Get Device Mnemonic or Type Code	H.REXS,71
M.DFCB	Create a File Control Block (FCB)	Vol. I, Ch. 1
M.DFCBE	Create an Expanded File Control Block (FCB)	Vol. I, Ch. 1
M.DIR and M_DIR	Create Directory	H.VOMM,4
M.DISCON and M_DISCON	Disconnect Task from Interrupt	H.REXS,38; H.MONS,38
M_DISMOUNT	Dismount Volume	H.REMM,19
M.DLTT and M_DLTT	Delete Timer Entry	H.REXS,6; H.MONS,6
M.DMOUNT	Dismount Volume	H.REMM,19
M.DSMI and M_DSMI	Disable Message Task Interrupt	H.REXS,57
M.DSUB and M_DSUB	Disable User Break Interrupt	H.REXS,73
M.DUMP and M_DUMP	Memory Dump Request	H.REXS,12; H.MONS,12
M.EAWAIT	End Action Wait	H.EXEC,40
M.EIR	Resident System Module Initialization Entry Macro	Vol. I, Ch. 1
M.ENMI and M_ENMI	Enable Message Task Interrupt	H.REXS,58
M.ENUB and M_ENUB	Enable User Break Interrupt	H.REXS,72

Macro name	Description	Reference
M.ENVRMT and M_ENVRMT	Get Task Environment	H.REXS,85
M.EXCL	Free Shared Memory (EXCLUDE)	H.ALOC,14
M. EXCLUDE and M_EXCLUDE	Exclude Memory Partition	H.MEMM,8; H.REMM,14
M.EXIT and M_EXIT	Terminate Task Execution	H.REXS,18; H.MONS,18
M.EXTD and M_EXTENDFILE	Extend File	H.VOMM,6
M_EXTSTS	Exit With Status	H.REXS,86
M.FADD	Permanent File Address Inquiry	H.MONS,2
M.FCBEXP	Create a File Control Block (FCB) for Execute Channel Program	Vol. I, Ch. 1
M.FD	Free Dynamic Extended Indexed Data Space	H.MEMM,4; H.REMM,9 H.ALOC,9
M.FE	Free Dynamic Task Execution Space	H.MEMM,6; H.REMM,11 H.ALOC,11
M.FILE	Open File	H.IOCS,1
M_FREEMEMBYTES	Free Memory in Byte Increments	H.MEMM,13; H.REMM,29
M.FSLR	Release Synchronization File Lock	H.FISE,25
M.FSLS	Set Synchronization File Lock	H.FISE,24
M.FWRD	Advance File or Record	H.IOCS,7; H.IOCS,8 Vol. I, Ch. 1
M.FXLR	Release Exclusive File Lock	H.FISE,23
M.FXLS	Set Exclusive File Lock	H.FISE,22
M.GADRL	Get Address Limits	H.REXS,41; H.MONS,41
M.GD	Get Dynamic Extended Data Space	H.MEMM,3; H.REMM,8
M.GDD	Get Dynamic Extended Discontiguous Data Space	H.MEMM,9
M.GE	Get Dynamic Task Execution Space	H.MEMM,5; H.REMM,10 H.ALOC,10
M_GETCTX	Get User Context	H.EXEC,41
M.GETDEF	Get Terminal Function Definition	H.TSM,15
M_GETMEMBYTES	Get Memory in Byte Increments	H.MEMM,12
M_GETTIME	Get Current Date and Time	H.REXS,74

Macro name	Description	Reference
M.GMSGP and M_GMSGP	Get Message Parameters	H.REXS,35; H.MONS,35
M.GRUNP and M_GRUNP	Get Run Parameters	H.REXS,36; H.MONS,36
M.GTIM and M_GTIM	Acquire System Date/Time in any Format	H.REXS,74
M.GTSAD and M_GTSAD	Get TSA Start Address	H.REXS,91
M.HOLD and M_HOLD	Program Hold Request	H.REXS,25; H.MONS,25
M.ID and M_ID	Get Task Number	H.REXS,32; H.MONS,32
M.INCL	Get Shared Memory (INCLUDE)	H.ALOC,13
M.INCLUDE and M_INCLUDE	Include Memory Partition	H.MEMM,7; H.REMM,12
M.INIT	Initialize Device Handler Parameters	Vol. I, Ch. 1
M.INITX	Initialize Device Handler Parameters	Vol. I, Ch. 1
M_INQUIRER and M.INQUIRY	Resource Inquiry	H.REMM,27
M.INT and M_INT	Activate Task Interrupt	H.REXS,47; H.MONS,47
M.IOFF	Inhibit Interrupt Signals	Vol. I, Ch. 1
M.IONN	Allow Interrupt Signals	Vol. I, Ch. 1
M.IPUBS and M_IPUBS	Set IPU Bias	H.REXS,82
M.IPUOFF	IPU Off Line	Vol. I, Ch. 1
M.IPUON	IPU On Line	Vol. I, Ch. 1
M.IPURTN	IPU Return	Vol. I, Ch. 1
M.IVC	Connect Entry Point to Interrupt Vector Location	Vol. I, Ch. 1
M.KILL	Halt Computer	Vol. I, Ch. 1
M_LIMITS	Get Base Mode Task Address Limits	H.REXS,84
M.LOC	Read Descriptor	H.VOMM,13
M.LOCK and M_LOCK	Set Exclusive Resource Lock	H.REMM,23

Macro name	Description	Reference
M.LOG	Permanent File Log	H.FISE,15; H.MONS,33
M.LOGR and M_LOGR	Log Resource or Directory	H.VOMM,10
M.MEM and M_MEM	Create Memory Partition	H.VOMM,3
M.MEMB	Get Memory in Byte Increments	H.MEMM,12
M.MEMFRE	Free Memory in Byte Increments	H.MEMM,13
M.MOD and M_MOD	Modify Descriptor	H.VOMM,11
M.MODT	Build Module Address Table Entry	Vol. I, Ch. 1
M.MODU and M_MODU	Modify Descriptor User Area	H.VOMM,26
M.MOUNT and M_MOUNT	Mount Volume	H.REMM,17
M.MOVE and M_MOVE	Move Data to User Address	H.REXS,89
M.MYID and M_MYID	Get Task Number	H.REXS,32; H.MONS,32
M.NEWRRS	Reformat RRS Entry	H.REXS,76
M.OLAY	Load Overlay Segment Load and Execute Overlay Segment	H.REXS,13; H.MONS,13 H.REXS,14; H.MONS,14
M.OPEN	Allow Context Switching	Vol. I, Ch. 1
M.OPENR and M_OPENR	Open Resource	H.REMM,21
M_OPTIONDWORD	Task Option Doubleword Inquiry	H.REXS,95
M_OPTIONWORD	Task Option Word Inquiry	H.REXS,24; H.MONS,24
M.OSREAD and M_OSREAD	Physical Memory Read	H.REXS,93
M.OSWRIT and M_OSWRIT	Physical Memory Write	H.REXS,94
M.PGOD	Task Option Doubleword Inquiry	H.REXS,95
M.PDEV	Physical Device Inquiry	H.MONS,1
M.PERM	Change Temporary File to Permanent	H.FISE,13
M.PGOW	Task Option Word Inquiry	H.REXS,24; H.MONS,24
M.PNAM	Reconstruct Pathname	H.VOMM,16

Macro name	Description	Reference
M.PNAMB and M_PNAMB	Convert Pathname to Pathname Block	H.VOMM,15
M.PRIL and M_PRIL	Change Priority Level	H.REXS,9; H.MONS,9
M.PRIV and M_PRIVMODE	Reinstate Privileged Mode to Privileged Task	H.REXS,78
M.PTSK and M_PTSK	Parameter Task Activation	H.REXS,40; H.MONS,40
M_PUTCTX	Put User Context	H.EXEC,42
M.QATIM and M_QATIM	Acquire Current Date/Time in ASCII Format	H.REXS,74
M.RADDR and M_RADDR	Get Real Physical Address	H.REXS,90
M.RCVR and M_RCVR	Receive Message Link Address	H.REXS,43; H.MONS,43
M.READ and M_READ	Read Record	H.IOCS,3
M_READD	Read Descriptor	H.VOMM,13
M.RELP and M_RELP	Release Dual Ported Disc	H.IOCS,27
M.RENAM and M_RENAME	Rename File	H.VOMM,14
M.REPLAC and M_REPLACE	Replace File	H.VOMM,23
M.RESP and M_RESP	Reserve Dual Ported Disc	H.IOCS,24
M_REWIND	Rewind File	H.IOCS,2
M.REWRIT and M_REWRIT	Rewrite Descriptor	H.VOMM,12
M.REWRTU and M_REWRTU	Rewrite Descriptor User Area	H.VOMM,27
M.ROPL and M_ROPL	Reset Option Lower	H.TSM,14
M.RRES and M_RRES	Release Channel Reservation	H.IOCS,13

Macro name	Description	Reference
M.RSML and M_RSML	Resourcemark Lock	H.REXS,62
M.RSMU and M_RSMU	Resourcemark Unlock	H.REXS,63
M.RSRV and M_RSRV	Reserve Channel	H.IOCS,12
M.RTNA	Return to Specified Address	Vol. I, Ch. 1
M.RTRN	Return In-Line	Vol. I, Ch. 1
M.RTRNOS	Return to Operating System	H.REXS,92
M.RWND	Rewind File	H.IOCS,2
M_SETERA	Set Exception Return Address	H.REXS,81
M_SETEXA	Set Exception Handler	H.REXS,83
M.SETS and M_SETS	Set User Status Word	H.REXS,7; H.MONS,7
M.SETSYNC and M_SETSYNC	Set Synchronous Resource Lock	H.REMM,25
M.SETT and M_SETT	Create Timer Entry	H.REXS,4; H.MONS,4
M.SHARE	Share Memory with Another Task	H.ALOC,12
M.SHUT	Inhibit Context Switching	Vol. I, Ch. 1
M.SMSGR and M_SMSGR	Send Message to Specified Task	H.REXS,44; H.MONS,44
M.SMULK	Unlock and Dequeue Shared Memory	H.ALOC,19
M.SOPL and M_SOPL	Set Option Lower	H.TSM,13
M.SPAD	Scratchpad Reference	Vol. I, Ch. 1
M.SRUNR and M _SRUNR	Send Run Request to Specified Task	H.REXS,45; H.MONS,45
M.SUAR and M_SUAR	Set User Abort Receiver Address	H.REXS,26; H.MONS,26
M.SUME and M_SUME	Resume Task Execution	H.REXS,16; H.MONS,16
M.SUSP and M_SUSP	Suspend Task Execution	H.REXS,17; H.MONS,17

Macro name	Description	Reference
M.SVCP	Build SVC Type 1 Protect Bits	Vol. I, Ch. 1
M.SVCP2	Build SVC Type 2 Protect Bits	Vol. I, Ch. 1
M.SVCT	Build SVC Type 1 Table Entry	Vol. I, Ch. 1
M.SVCT2	Build SVC Type 2 Table Entry	Vol. I, Ch. 1
M.SYNCH and M_SYNCH	Set Synchronous Task Interrupt	H.REXS,67
M.TBRKON	Break Processing Entry	H.TSM,6
M.TDAY and M_TDAY	Time-of-Day Inquiry	H.REXS,11; H.MONS,11
M.TEMP	Create Temporary File	H.VOMM,2; H.VOMM,24
M.TEMPER and M_TEMPFILETOPERM	Change Temporary File to Permanent File	H.VOMM,9
M.TRAC	System Trace	Vol. I, Ch. 1
M.TRNC and M_TRUNCATE	Truncate File	H.VOMM,7
M.TSCAN	Syntax Scanner	H.TSM,2
M.TSTE and M_TSTE	Arithmetic Exception Inquiry	H.REXS,23; H.MONS,23
M.TSTS and M_TSTS	Test User Status Word	H.REXS,8; H.MONS,8
M.TSTT and M_TSTT	Test Timer Entry	H.REXS,5; H.MONS,5
M.TURNON and M_TURNON	Activate Program at Given Time of Day	H.REXS,66
M.TYPE and M_TYPE	System Console Type	Vol. I, Ch. 1
M.UNLOCK and M_UNLOCK	Release Exclusive Resource Lock	H.REMM,24
M_UNPRIVMODE	Change Task to Unprivileged Mode	H.REXS,79
M.UNSYNC and M_UNSYNC	Release Synchronous Resource Lock	H.REMM,26
M.UPRIV	Change Task to Unprivileged Mode	H.REXS,79
M.UPSP and M_UPSP	Upspace	H.IOCS,20
M.USER	Username Specification	H.MONS,34
M.USHUT	Inhibit User Task Context Switching	Vol. I, Ch. 1

Macro name	Description	Reference
M.VADDR and M_VADDR	Validate Address Range	H.REXS,33
M.WAIT and M_WAIT	Wait I/O	H.IOCS,25
M.WEOF	Write End of File (EOF)	H.IOCS,5
M.WRIT and M_WRITE	Write Record	H.IOCS,4
M_WRITEEOF	Write EOF	H.IOCS,5
M.XBRKR and M_XBRKR	Exit from Task Interrupt Level	H.REXS,48; H.MONS,48
M.XIEA and M_XIEA	No-wait I/O End-Action Return	H.IOCS,34
M.XIR	Resident System Module Initialization Exit Macro	Vol. I, Ch. 1
M.XMEA and M_XMEA	Exit from Message End-Action Routine	H.REXS,50; H.MONS,50
M.XMSGR and M_XMSGR	Exit from Message Receiver	H.REXS,39; H.MONS,39
M.XREA and M_XREA	Exit from Run Request End-Action Routine	H.REXS,51; H.MONS,51
M.XRUNR and M_XRUNR	Exit Run Receiver	H.REXS,49; H.MONS,49
M.XTIME and M_XTIME	Task CPU Execution Time	H.REXS,65

MPX-32 ™

Resident Handlers

Revision 3.5

Technical Manual Volume II(B)

April 1990

H.DCSCI
H.DCXIO
H.DPXIO
H.F8XIO
H.GPMCS
H.HSDG
H.IBLG
H.MDXIO
H.MTSCI
H.XIOS*

322-001552-500

ENCORE

Multi-Function Processor Disk Handler (H.DCSCI)

MPX-32 Technical Manual

Volume II

Contents

Contents

List of Tables

1 H.DCSCI Overview

1.1 Introduction

The multi-function processor (MFP) disk handler (H.DCSCI) provides user tasks with an I/O path to small computer system interface (SCSI) disks connected to an MFP. H.DCSCI performs the following:

- builds the SCSI command data blocks (CDBs)
- issues channel programs
- collects and reports status about the I/O operation to the user task and MPX-32
- queues I/O operations for a particular disk and issues the next queued I/O operation

This section discusses the modules used by H.DCSCI and the SYSGEN considerations.

1.2 Modules Used by H.DCSCI

H.DCSCI calls the following modules: H.IFXIO, XIO.SUB, and H.IOCS. H.IFXIO is the interrupt fielder and corresponds one-to-one with the channel. XIO.SUB is the extended I/O common subroutine package that performs extended I/O functions. H.IOCS performs device-independent I/O request management. This includes preprocessing and postprocessing of I/O requests and I/O queue (IOQ) management. For more information about H.IFXIO, XIO.SUB, and H.IOCS refer to the H.XIOS and H.IOCS sections in this manual.

1.3 SYSGEN Considerations

To include H.DCSCI as part of MPX-32, specify H.DCSCI in the SYSGEN directive file. The SYSGEN CONTROLLER and DEVICE directives are used to specify the MFP and H.DCSCI, respectively. Each SCSI disk drive has a unique device subaddress that is specified with the DEVICE directive. H.DCSCI is system re-entrant, which means that only one copy should be configured into MPX-32. For more information about the SYSGEN directive file, refer to the System Generation (SYSGEN) chapter in Volume III of the MPX-32 Reference Manual.

Following are examples of the CONTROLLER and DEVICE directives that configure MPX-32 for SCSI disk support:

```
CONTROLLER=DM76,PRIORITY=10,CLASS=F,MUX=MFP,SUBCH=0
DEVICE=00,DTC=DM,SHR,DISC=ANY,HANDLER=H.DCSCI
```

2 H.DCSCI Structures and Entry Points

2.1 Introduction

This section describes the data structures and the entry points that H.DCSCI uses to perform I/O.

2.2 Data Structures

This section describes the following data structures:

- device context area (DCA)
- small computer system interface (SCSI) command data block (CDB)
- handler address table (HAT)
- status doubleword

Other data structures that H.DCSCI uses to perform I/O are the unit definition table (UDT), controller definition table (CDT), I/O queue (IOQ), file control block (FCB), and file assignment table (FAT). For more information about these data structures, refer to the MPX-32 Technical Manual Volume I, Chapter 2.

2.2.1 DCA

The DCA stores subchannel operation information. It contains a common section (words 0 through 35) and a device-dependent section. A DCA must be specified for each subchannel. Table 2-1 lists the H.DCSCI device-dependent information. For more information about the DCA common section, refer to the MPX-32 Technical Manual Volume I, Chapter 2.

Table 2-1
H.DCSCI Device-Dependent DCA Information

Word	Hex Byte	Meaning
36	90	time out (DCA.MAX)
37	94	handler flag word (DCA.CDBF)
38/39	98/9C	read capacity buffer (DCA.RCAP)
40	A0	sectors per track (DCA.SPT)
41	A4	current IOCD address (DCA.A1)
42	A8	error queue (DCA.ERRQ)
43	AC	EOF buffer (DCA.EOFB)
44/48	B0/B4	local handler storage (DCA.SAVR)

2.2.2 SCSI CDB

H.DCSCI uses the SCSI CDB to communicate I/O requests to SCSI device controllers. For more information about the SCSI CDB, refer to ANSI SCSI Committee Working Document X3T9.2/82.2.

2.2.3 HAT

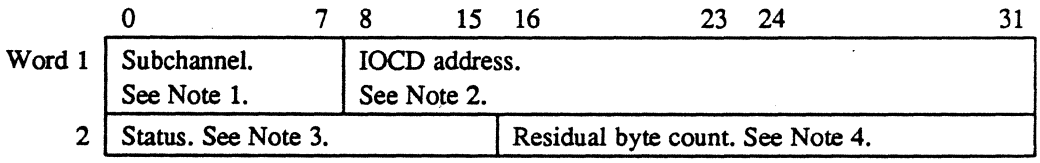
The HAT contains the addresses and the names of the entry points to the H.DCSCI I/O processing routines. It is used by modules such as SYSINIT and the I/O control system (IOCS) to access H.DCSCI.

2.2.4 Status Doubleword

A status doubleword contains the result of the last executed I/O command doubleword (IOCD) when an I/O termination occurs. The MFP generates a status doubleword when an interrupt occurs or when it receives a status stored response from a start I/O (SIO) or halt I/O (HIO) instruction.

When an I/O operation completes, H.DCSCI checks the 16 status bits in the status doubleword for error conditions. H.DCSCI considers the I/O operation complete if the status doubleword has the channel end and device end bits set. If other bits are set, H.DCSCI issues a sense IOCD for additional information about the error.

H.DCSCI stores the sense information in the DCA and maps the status bits, sense bits, and drive status bits to the user's file control block (FCB). For more information about the FCB, refer to the System Tables and Variables chapter in the MPX-32 Technical Manual, Volume I. A status doubleword has the following format:



Notes:

1. This field contains the subchannel address of the interrupting device.
2. The IOCD address points 8 bytes past the last executed IOCD.
3. This field contains the following status bits:

Bit	Definition
0	reserved
1	post program-controlled interrupt
2	incorrect length
3	channel program check
4	channel data check
5	reserved
6	interface control check
7	reserved
8	device busy
9	status modifier
10	controller end
11	attention
12	channel end
13	device end
14	unit check
15	unit exception

4. This field contains the number of bytes not transferred for the last IOCD processed.

2.3 Entry Points

Entry points are H.DCSCI routines that perform specific I/O processing. The HAT contains the addresses and names of these routines. This section describes the processing performed by H.DCSCI in the following entry points:

- opcode processor (OP.)
- IOQ driver
- service interrupt processor
- lost interrupt processor (LI.XIO)
- post-transfer processor (PX.)
- SYSGEN initialization (SG.)
- execute channel program opcode processor (XCHANP). For more information about this entry point, refer to the H.XIOS section in this manual.

2.3.1 Entry Point OP. – Opcode Processor

Entry point OP. processes the opcode placed in the FCB by the I/O service originally called by the user. (OP. is a subroutine extention of H.IOCS,29.) It then indicates the appropriate action for H.IOCS,29 by taking one of the following returns to H.IOCS,29:

BU SERVCOMP	service complete, no device access required
BU IOLINK	link request to IOQ

If OP. takes return IOLINK, it must first call IOCS subroutine S.IOCS13 to allocate and initialize an IOQ. OP. then builds an IOCL into the IOQ using IOCS subroutines S.IOCS12 and IOCS entry point S.IOCS40. If necessary, OP. also obtains space to build the SCSI CDB.

Entry Conditions

Calling Sequence

BL	*1W,X2	X2 contains the HAT address. The 1W offset from this address is the address of OP.

Registers

R1	FCB address
R2	HAT address
R3	UDT address

Exit Conditions

Return Sequence

See descriptions of SERVCOMP and IOLINK.

Registers

R1	FCB address

2.3.2 IOQ Driver

The IOQ driver issues SIO instructions for the IOCLs. H.DCSCI can queue a maximum of 32 outstanding requests per channel. The IOQ driver has two entry points: IQ.XIO and IQ.XIO.1. These entry points are identical except that IQ.XIO activates the interrupt level upon entry and deactivates it before exiting. IQ.XIO and IQ.XIO.1 issue an SIO instruction for the first request in the IOQ.

2.3.3 Entry Point IQ.XIO

H.IOCS,29 calls entry point IQ.XIO when H.IOCS,29 queues an I/O request. It blocks external interrupts and enters IQ.XIO with the following calling sequence.

Entry Conditions

Calling Sequence

BL	*2W,X2	X2 contains the HAT address. The 2W offset from this address is the address of IQ.XIO.

Registers

R0	return address
R3	UDT address of the device to start
R7	IOQ address

Exit Conditions

Return Sequence

DACI		deactivate interrupt level
TRSW	R0	return to calling routine

Registers

R7	IOQ address

2.3.4 Entry Point IQ.XIO.1

The service interrupt processor calls IQ.XIO.1 to drive the IOQ when an I/O request completes. LI.XIO calls IQ.XIO.1 to clear the IOQ when a HIO instruction times out.

Entry Conditions

Calling Sequence

```
BL      IQ.XIO.1        return to call
(or)
BU      IQ.XIO.1        return is set up
```

Registers

R0	return address SI
R3	UDT address of device to start

Exit Conditions

Return Sequence

```
TRSW R0  R0 set up prior to return if call is from LI.XIO
```

Registers

R1	CHT address
R2	DCA address

2.3.5 Service Interrupt Processor

The service interrupt processor performs postaccess processing associated with the device access which just completed. It has one entry point, SI, and two routines, SI.UNLNK and SI.EXIT that perform the logic sequence described in the following section.

2.3.6 Entry Point SI

SI services interrupts and performs device-dependent logic. It is entered directly from the extended I/O interrupt fielder program (H.IFXIO) with the interrupt level active when one of the following conditions occur:

- an I/O request completes normally
- status checking is inhibited
- status contains a channel end with no device end
- a sense IOCD, unexpected interrupt, or device time out occurs

The following text describes the SI routines that H.DCSCI enters when one of these conditions occur.

Normal Completion or Status Checking Inhibited Routine – H.DCSCI enters this routine when an I/O request completes with no errors or when status checking is inhibited. It performs any required device-specific processing. An example is the collection of sense information about an I/O operation that just completed.

The extended I/O common routines collect sense information only when an I/O request produces an error or when the I/O request was for an execute channel program and sense information was requested.

This routine:

- checks for a reserve request. If the request exists, the routine increments the reserve count.
- checks for an advance or backspace file request. If the request exists, the routine sets EOF and EOM or BOM and continues processing at SI.EXIT. If no request exists, the routine updates the FAT.
- computes the transfer count
- continues processing at SI.UNLNK

Channel End with No Device End Routine – H.DCSCI enters this routine when an I/O request produces an interrupt whose status contains channel end and no device end. This is not a normal case except when a reserve request is issued to a dual-ported disk that is reserved to the opposing CPU. Any other condition is treated as an unexpected interrupt.

This routine checks for a channel end from a reserve request. If there is no channel end, the routine continues processing as an unexpected interrupt. It then updates the time out in the UDT, shows the I/O as active, and continues processing at SI.EXIT.

Normal Sense Command with IOQ Routine – H.DCSCI enters this routine following an interrupt caused by issuing a sense IOCD for an I/O request that completed with an error indication. This routine examines the status and sense information, initiates error recovery if applicable, and sets the appropriate indicators based on the sense data. It then computes the actual transfer count (if an error condition is indicated), updates the IOQ transfer count, and continues processing at SI.UNLNK.

Unexpected Interrupt Routine – H.DCSCI enters this routine when an interrupt occurs that was not expected. This routine increments the spurious interrupt count for the device and the channel and continues processing at SI.EXIT.

Device Time Out Routine – H.DCSCI enters this routine when an HIO instruction generates an interrupt. (The HIO instruction was issued for a timed-out device.) This routine sets the error condition and the time out flag in the IOQ. It then continues driving the IOQ and processing at SI.EXIT.

Table 2-2 lists the cause of interrupts and the response by SI in performing conditional service interrupt processing.

Table 2-2
Interrupts and Responses by SI

Cause of Interrupt	Response by SI
Channel end with no device end	checks for a reserve request (if dual port), continues processing at SI.EXIT
Device time out	marks unrecoverable error condition and actual transfer count in the IOQ, H.DCSCI continues to drive the queue
Execute channel program	sets error condition in the IOQ if an error is indicated. If sense information is required, issues a sense IOCD and continues processing at SI.EXIT. If no error is found and no sense information is required, continues processing at SI.UNLNK.
I/O request completes with error	issues sense IOCD and continues processing at SI.EXIT
Normal sense command with IOQ	if execute channel program was requested, continues processing at SI.UNLNK, otherwise, completes the actual transfer count computed and updates the IOQ
Rewind or seek complete	clears device rewinding or seeking bit, continues processing at SI.EXIT
Spurious interrupt when device is not configured	increments spurious interrupt count, and exits the interrupt level. Continues processing at SI.EXIT
Spurious interrupt when device is configured but interrupt is not expected	increments spurious count for the device and the channel, continues processing at SI.EXIT

Entry Conditions

Calling Sequence

BU	*3W,X2	X2 contains the HAT address. The 3W offset from this address is the address of SI.

Registers

R1	CHT address
R3	UDT address

Exit Conditions

Return Sequence

BU	SI.UNLNK	SI unlinks the I/O request, reports the I/O request as complete and IOQ processing continues
(or)		
BU	SI.EXIT	SI exits the interrupt level without unlinking the I/O request and reporting the I/O as complete. It then initiates error retry or collects sense data.

Registers

R1	IOQ address for SI.UNLNK
R2	DCA address

2.3.7 SI.UNLNK Routine

The SI.UNLNK routine unlinks the IOQ entry from the UDT and reports I/O complete to MPX-32. SI.UNLNK is entered with the interrupt level active. SI.UNLNK is also entered when a device times out due to a kill request or if a device malfunctions.

Entry Conditions

Calling Sequence

BU	SI.UNLNK

Registers

R1	IOQ address
R2	DCA address
R7	IQ. return address

Exit Conditions

Return Sequence

Continues with remainder of SI logic.

Registers

Registers are not changed.

2.4 SI.Exit Routine

The SI.EXIT routine continues driving the IOQ and exits the interrupt level. SI.EXIT is entered with the interrupt level active.

Entry Conditions

Calling Sequence

```
BU      SI.EXIT
```

Registers

R2 DCA address

Exit Conditions

Return Sequence

Continues with remainder of SI logic.

Registers

Registers remain unchanged.

2.4.1 Entry Point LI.XIO – Lost Interrupt Processor

S.IOCS5 calls LI.XIO to take corrective measures when an expected interrupt fails to occur. It is also called from H.IOCS,38 when a kill request is issued to a task and the task has an I/O operation in progress. In both cases, the I/O request terminates with an HIO instruction. If the controller responds to the HIO instruction, SI.A performs the required interrupt handling. LI.XIO performs the following:

- activates the interrupt level
- increments the lost interrupt count (if LI.XIO is entered due to a lost interrupt)
- issues the HIO if it has not already been issued
- LI.XIO blocks external interrupts, deactivates the interrupt level, and returns to the calling routine.
- clears outstanding I/O queue entries

If the HIO instruction has already been issued but fails to generate an interrupt, LI.XIO is entered again and takes the following actions:

- activates the interrupt level
- increments lost interrupt count (if LI.XIO is entered due to a lost interrupt)
- marks the device as offline and malfunctioning
- unlinks the I/O request from the IOQ
- reports the I/O request as complete with errors (if the HIO instruction was issued because of a lost interrupt)
- branches and links to IQ.XIO.1 to clear any pending I/O requests to the failing device
- blocks external interrupts, deactivates the interrupt level, and returns to calling routine

Entry Conditions

Calling Sequence

BL	*4W,X1	X1 contains the HAT address. The 4W offset from this address is the address of LI.XIO.

Registers

R0	return address
R3	UDT address of device to halt

Exit Conditions

Return Sequence

TRSW	R0

Registers

Registers remain unchanged.

2.4.2 Entry Point PX. – Post-Transfer Processor

S.IOCS1 calls entry point PX. to perform processing after completion of the I/O request and before returning to the requesting task. PX. executes at the task priority and with low system overhead.

Entry Conditions

Calling Sequence

BL	*5W,X2	X2 contains the HAT address. The 5W offset from this address is the address of PX.

Registers

R1	FCB address
R2	HAT address
R3	UDT address

Exit Conditions

Return Sequence

TRSW	R0

Registers

R1	FCB address

Notes:

If an advance file or a backspace file request is issued, PX. updates the current position in the FAT.

If a read track label zero request is issued, H.DCSCI builds a dummy record that specifies the number of logical blocks available on the disk. This record also specifies that the highest available disk block is defective. This is necessary because SCSI disks do not support the read track label zero request.

If an open request is performed, H.DCSCI fills in the sectors per track field (UDT.SPT) in the UDT.

2.4.3 Entry Point SG. – SYSGEN Initialization

SYSGEN calls entry point SG. to initialize certain handler parameters and data structures during the construction of an MPX-32 image. One DCA is initialized for each UDT entry containing the name of the handler. SYSGEN overlays any remaining DCA's and the remainder of the code in H.DCSCI. SG. updates the DCA with the default time out for I/O. SG. also updates the CHT, CDT, and the UDT to identify the device as a SCSI disk.

Entry Conditions

Calling Sequence

BL	*6W,X1	X1 is the HAT address. The 6W offset from this address is the address of SG.

Registers

Not applicable.

Exit Conditions

Return Sequence

M.XIR	standard handler SYSGEN exit macro

Registers

Registers are not changed.

3 H.DCSCI Issuing I/O Operations

3.1 Overview

There are two ways to issue I/O operations to a SCSI disk drive via H.DCSCI: device-dependent I/O and device-independent I/O. Both methods use the same data structures, channel commands, and CPU instructions.

With device-dependent I/O, the user performs direct channel I/O to the SCSI disk drive via H.DCSCI. The user builds a channel program that H.DCSCI accesses with a physical or logical address. The user also builds an accompanying SCSI command data block (CDB) if there is a transfer command packet (TCP) channel command in the channel program. To initiate the I/O operation, the user issues an execute channel program (EXCPM) request.

With device-independent I/O, the user initiates I/O operations by issuing service (SVC) calls to the I/O control system (IOCS). IOCS verifies the logical address that the user places in the task's file control block (FCB) and links the I/O request to H.DCSCI. H.DCSCI then constructs the necessary channel program and CDB and issues the appropriate CPU instruction to initiate the I/O operation.

This section discusses CPU instructions, channel commands, IOCS service calls, and error processing. For more information about issuing I/O operations, refer to the Resource Assignment/Allocation and I/O chapter in Volume I of the MPX-32 Reference Manual.

3.2 CPU Instructions

H.DCSCI uses CPU instructions to perform I/O to an MFP. The CPU instructions have the following format:

0 5	6 8	9 12	13 15	16 31
Opcode. See Note 1.	Register. See Note 2.	Instruction.	Augument code. See Note 3.	Constant. See Note 4.

Notes:

1. Bits 0-5 specify the hexadecimal operation code 0-FC.
2. Bits 6-8 specify the general purpose register. When these bits are nonzero, the register contents are added to the constant specified in bits 16-31 to form the logical channel and subaddress.
3. Bits 13-15 specify the augment code up to hexadecimal 7.
4. Bits 16-31 specify a constant that is added to the contents of the register specified by bits 6-8. This forms the logical channel and subaddress. If bits 6-8 are zero, this field specifies the logical channel and subaddress.

The MFP generates a condition code to report the result of a CPU instruction. Following are the possible condition codes:

Condition Code	Meaning
0001	Channel busy — request denied because channel internal process queue is full.
0010	Channel inoperative or undefined — request denied because channel is not present, functional, or defined.
0011	Subchannel busy — request denied because of outstanding halt I/O (HIO), stop I/O (STPIO), or clear queue instruction.
1000	Request accepted and queued — the CPU instruction was accepted by the channel for execution.

Table 3-1 lists and the following text describes the CPU instructions.

Table 3-1
CPU Instructions

Instruction	Hex Opcode
activate channel interrupt (ACI)	E
clear queue	7
deactivate channel interrupt (DACI)	F
disable channel interrupt (DCI)	D
enable channel interrupt (ECI)	C
halt I/O (HIO)	6
reset channel (RSCHNL)	5
reset controller (RSCTL)	8
start I/O (SIO)	2
stop I/O (STPIO)	4
test I/O (TIO)	3

3.2.1 Activate Channel Interrupt (ACI)

The ACI instruction causes the MFP to assert its interrupt priority level and actively contend for recognition from the CPU. While contending for recognition, the channel cannot request another interrupt.

3.2.2 Clear Queue

The clear queue instruction terminates all outstanding I/O operations for the specified subaddress. The current operation terminates with channel end and device end status. Queued operations terminate with channel end, device end, and unit exception status to indicate that they were not initiated.

3.2.3 Deactivate Channel Interrupt (DACI)

The DACI instruction causes the MFP to suspend contention for interrupt priority. If an interrupt request is queued and enabled, the channel may now issue an interrupt.

3.2.4 Disable Channel Interrupt (DCI)

The DCI instruction prevents the MFP from requesting interrupts from the CPU.

3.2.5 Enable Channel Interrupt (ECI)

The ECI instruction allows the MFP to request interrupts from the CPU.

3.2.6 Halt I/O (HIO)

The HIO instruction causes the MFP to terminate an I/O operation and post a status doubleword that contains device end (DE) and channel end (CE) status.

3.2.7 Reset Channel (RSCHNL)

The RSCHNL instruction causes the MFP to stop operation, reset all activity, and become idle. RSCHNL resets all subchannels and any requesting or active interrupt levels.

3.2.8 Reset Controller (RSCTL)

The RSCTL instruction causes the addressed subchannel to terminate an I/O operation. If the subchannel is hung, RSCTL resets the device so that I/O operation may resume. If the addressed subchannel is not currently being serviced, RSCTL clears the I/O request. The MFP presents any pending status for the addressed subchannel to the CPU. RSCTL does not generate final status.

3.2.9 Start I/O (SIO)

The SIO instruction initiates an I/O operation to the specified channel or subchannel. If this channel or subchannel is present and available, it accepts the SIO instruction. The channel or subchannel then performs the I/O operation specified by the SIO instruction. If the I/O operation cannot be started, the MFP returns the appropriate condition codes and status.

3.2.10 Stop I/O (STPIO)

The STPIO instruction terminates the I/O operation at the addressed subchannel when the current channel command has been executed. STPIO resets the data chain and command chain flags, and the MFP returns the appropriate condition codes and status.

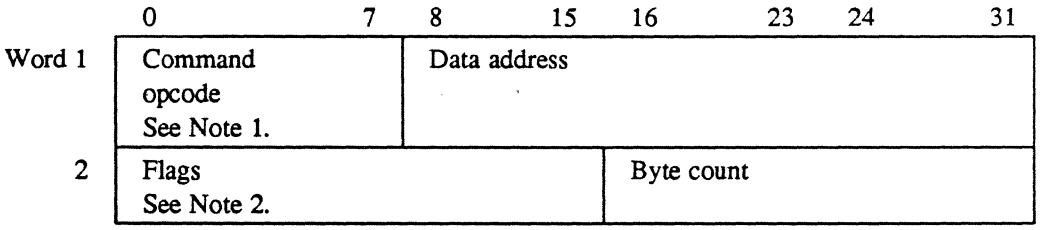
3.2.11 Test I/O (TIO)

The TIO instruction verifies the current state of the channel or subchannel and clears pending interrupt conditions. The MFP returns condition codes that reflect the status of the channel and addressed subchannel.

3.3 Channel Commands

Channel commands contain the information required for data transfers. This information includes the type of operation, the address in CPU memory where data is to be moved to or from, flags that indicate to the MFP what to do after completing execution of the channel command, and the amount of data (in bytes) to transfer.

Channel commands are specified in I/O command doubleword (IOCD) format as follows:



Notes:

1. The command opcode field specifies the command to be executed.
2. The flags are defined as follows:

Bit	Meaning when set
0	data chain
1	command chain
2	suppress incorrect length
3	skip read data
4	post program-controlled interrupt
5-15	not used (must be zero)

Table 3-2 lists and the following text describes the channel commands that can be issued for a SCSI disk.

Table 3-2
H.DCSCI Channel Commands

Command	Opcode
channel control	x'80'
initialize channel	x'00'
initialize subaddress	x'F0'
inquiry	x'B3'
no operation	x'03'
read capacity	x'53'
read	x'02'
reassign block	x'13'
release	x'C3'
reserve	x'A3'
rezero	x'37'
seek	x'07'
sense	x'04'
transfer command packet	x'D3'
transfer in channel	x'08'
write	x'01'

3.3.1 Channel Control

The channel control command returns three words of channel information. Word 1 is the board model number, word 2 is the firmware model number, and word 3 is the firmware revision level.

3.3.2 Initialize Channel

The initialize channel command transfers disk drive information to the MFP and sets the status buffer address for the MFP. The status buffer address must be doubleword bounded and specified in the data address field. The byte count field is ignored. The initialize channel command, performed by H.DCSCI, must be the first channel command to any channel that has an MFP configured.

3.3.3 Initialize Subaddress

The initialize subaddress command initializes several channel operations with the information from the byte in memory specified by the data address field. The byte count must be one. Following is the format of the byte:

Bit	Meaning when set
0	clear SIO queue on error. As each queued SIO is cleared, termination status is returned indicating this action. If reset, when a queued SIO command terminates in error, continue processing SIO queue.
1	reserved
2	use drive-specified block size. If reset, convert logical block size of 256 bytes to simulate block size of 768.
3-7	reserved

3.3.4 Inquiry

The inquiry command returns information about the device such as the peripheral device type, vender identification, and device-type qualifier. Refer to ANSI SCSI Committee Work Document X3T9.2/82.8 for more information about the information returned for this command.

3.3.5 No Operation

The no operation command is a non-data transfer command that executes without selecting an associated disk drive. The byte count and data address must be zero.

3.3.6 Read Capacity

The read capacity command returns two words of information. The first word is the total number of logical blocks on the disk. The second word contains the byte count of each logical block. The byte count specifies the maximum number of bytes returned. This command cannot be data chained.

3.3.7 Read

The read command transfers data from the disk to the address specified in the data address field. The byte count must be greater than zero.

3.3.8 Reassign Block

The reassign block command reassigns a logical block on the specified disk. The data address specifies the address of the defect list. The byte count is the length of the defect list and must be greater than zero. For more information about defect lists, refer to the ANSI SCSI Committee Work Document X3T9.2/82.8.

3.3.9 Release

The release command releases a reserved device by the reserving CPU. The release is not issued if more than one task has the device reserved. The data address and byte count fields are ignored.

3.3.10 Reserve

The reserve command reserves a device to the requesting CPU until a release command is issued. The data address and byte count fields are ignored.

3.3.11 Rezero

The rezero command loads zero into an internal register. The MFP uses this register to calculate the next logical block address for a read or write command. The data address and byte count fields are ignored.

3.3.12 Seek

The seek command loads the specified logical block address into an internal register. The MFP uses this register to calculate the logical block address for the next read or write operation. The data address field points to a buffer that contains a 1-, 2-, or 4-byte logical disk address. This address may specify a byte boundary in memory. The byte count field indicates that a 1-, 2-, or 4-byte seek address was specified. If one or two bytes are specified, the remaining bytes of the seek register are filled with zeros. This command cannot be data chained.

Note: If the logical block size of the media is 256 and a 768 block size emulation is enabled, the logical block address is multiplied by three before it is saved in the register.

3.3.13 Sense

The sense command causes H.DCSCI to issue an extended sense command to the specified device for the number of bytes requested in the byte count field. For more information about the returned information, refer to ANSI SCSI Committee Working Document X3T9.2/82.2.

3.3.14 Transfer Command Packet

The transfer command packet command sends a SCSI CDB to a device. The data address specifies the address of the CDB built by the user. The byte count specifies the length of the CDB. If a data transfer is required with the CDB, the command chain flag must be set. This command must be command chained to a write or a read command that specifies the data address and byte count.

3.3.15 Transfer in Channel

The transfer in channel command causes IOCD execution to continue at the address specified in the data address field. A transfer in channel command cannot point to another transfer in channel command, and it cannot be the first command in a channel program. The handler uses a transfer in channel command to link channel commands in the device context area (DCA) to channel commands in the I/O queue (IOQ).

3.3.16 Write

The write command transfers data to the disk from the address specified in the data address field. The byte count must be greater than zero.

3.4 IOCS Service (SVC) Calls

Table 3-3 lists the SVC calls, their applicable functions, and the bit setting for the FCB field FCB.SCFG.

Table 3-3
IOCS SVC Calls

Function	SVC	FCB.SCFG	Notes
advance file	1,x'34'	0000	
advance record	1,x'33'	0000	
backspace file	1,x'36'	0000	
backspace record	1,x'35'	0000	
execute channel program	1,x'25'	n/a	1
format unit	1,x'10'	n/a	3
get/remove bad block	1,x'0D'	0100	4
initialize subaddress	1,x'3E'	0001	5
inquiry	1,x'0D'	0101	6
mode select	1,x'38'	0101	7
mode sense	1,x'38'	n/a	2, 8
open	n/a	n/a	
read capacity	1,x'0D'	0110	2
read data	1,x'31'	0000	2
read defect data	1,x'3E'	0010	2, 9
read diagnostic data	1,x'3E'	0011	2, 13
read extended data	1,x'31'	0000	2
read MFP	1,x'0D'	0001	2, 10
reassign block(s)	1,x'3E'	0000	11
release unit	1,x'27'	n/a	
reserve unit	1,x'26'	0010	
request sense	1,x'35'	0010	2
rewind	1,x'37'	0010	
rezero	1,x'0D'	0011	
seek	1,x'0D'	0010	12
send diagnostics	1,x'0D'	1000	2
test unit ready	1,x'0D'	0000	2
write data	1,x'32'	0000	2
write diagnostic data	1,x'3E'	0100	2, 13
write EOF	1,x'38'	0000	

Notes:

1. FCB.IOQA must contain the logical address of the channel program to execute.
2. FCB.ERWA specifies the buffer address. FCB.EQTY specifies the byte count.
3. FCB.ERWA specifies the address of the buffer that contains the SCSI command packet to be sent to the unit. FCB.EQTY specifies the byte count.
4. FCB.ERWA specifies the buffer address. To get and remove a bad block, the buffer must contain the bad block number.

5. FCB.ERWA and FCB.EQTY specify the address of the buffer that contains the initialization parameters and the these parameters, respectively. The byte count must be one.
6. The byte count specified in FCB.EQTY must be larger than the expected data. H.DCSCI sets the suppress incorrect length bit.
7. FCB.ERWA specifies the address of the buffer that contains the mode select parameters. The requestor specifies these parameters. FCB.EQTY specifies the byte count for the mode select parameter data.
8. The first byte specified by the buffer address in FCB.ERWA is the page code for the mode sense function. For more information about page codes, refer to ANSI Committee Working Document X3T9.2/82.2. The byte count specified in FCB.EQTY must be large enough for the returned data. H.DCSCI sets the suppress incorrect length bit in the read command.
9. The first byte in the buffer specified by the address in FCB.ERWA is the format of the requested data as follows: x'00' block format, x'04' bytes from index format, x'05' physical format. The byte count specified in FCB.EQTY must be large enough for the amount of data to be returned. H.DCSCI sets the suppress incorrect length bit in the read command.
10. The byte count must be from 1 to 12 bytes.
11. FCB.ERWA specifies the address of the buffer that contains the block address(es). Each block is four bytes. FCB.EQTY specifies the number of blocks to reassign.
12. FCB.ERWA specifies the address of the buffer that contains the seek argument. FCB.EQTY specifies the length of the seek argument.
13. FCB.BBA must contain the logical block address.

3.5 Error Processing

If an I/O operation terminates abnormally, H.DCSCI posts information about the I/O operation in the FCB. This information is device status in FCB word 3, bits 16 through 31, and sense data in FCB words 11 and 12. For more information about the status, refer to the status doubleword section in Chapter 2 of this document.

The following is a list of the sense data returned for H.DCSCI:

FCB Word	Byte	Sense Data	Notes
11	0	device subaddress	
	1	sense key	1
	2	error class/code	2
	3	not used	
12	0-3	channel status	3

Notes:

1. The values for the sense key byte are defined as follows:

Value	Meaning
0	no sense
1	recovered error
2	not ready
3	medium error
4	hardware error
5	illegal request
6	unit attention
7	data protect
8	blank check
9	vendor unique
A	copy aborted
B	aborted command
C	equal comparison satisfied
D	volume overflow
E	miscompare
F	reserved

2. This specifies extended sense data format. For more information, refer to ANSI Committee Working Document X3T9.2/82.2.
3. This word contains channel status in bits 0 through 16 and the residual byte count in bits 17 through 31.

Extended I/O Disk Handler (H.DCXIO)

MPX-32 Technical Manual

Volume II

Contents

Contents

Page

List of Figures

List of Tables

1 H.DCXIO Overview

1.1 General Information

The Extended I/O Disk Handler (H.DCXIO) is a software component of MPX-32 intended to provide support for Extended I/O Disk Processors (XIO), Input/Output Disk Processors (IOP), and Floppy Disk Controllers (IOP) connected to an MPX-based CONCEPT/32 computer.

The product is designed to support any number and mix of extended I/O disk drives listed below. These include fixed-head disks (FHD), moving-head disks (MHD), cartridge module drives (CMD), fixed module drives (FMD), minimodule drives (Winchester) (MMD) and floppy disks.

The design supports IOCS callable I/O service requests as described in the MPX-32 Reference Manual, Volume I.

An execute channel program capability has been incorporated to allow the user to execute his own IOCD list. Error conditions are detected and noted in the FCB, however, error correction and error retry are the responsibility of the user. Reserve and release IOCDs should never be included within an execute channel program IOCD list. See sections 2.18 and 2.19.

1.2 Disks Supported

The IOP disk controller and XIO disk processor Phase I support the single-ported versions of the following disks. The XIO disk processor Phase II supports both single and multiported versions of the following disks:

Manufacturer	Encore ID #	Type	Heads	Cylinders	Maximum Formatted Data Byte Capacity	SYSGEN Device Code
CDC	9320	MHD	5	823	63.20MB	MH080
CDC	9323	MHD	19	823	240.15MB	MH300
CDC	8104	FHD	4	64	3.93MB	FH005
CDC	8117	CMD	1+1	823	25.28MB	CD032
CDC	8122	MMD	5	823	63.20MB	MH080
CDC	8127	MMD	10	823	126.41MB	MH160
CDC	8858	MHD	24	711	262.10MB	MH340
CDC	8155	FMD	40	843	517.94MB	MH600
CDC	8172	Floppy	2	77	1.02MB	FL001

CDC – Control Data Corporation
MHD – removable media, moving-head disk
FHD – captive media, moving-head disk
CMD – cartridge module drive, removable and captive media
FMD – fixed module drive
MMD – minimodule drive (Winchester)

Moving head disks (MHD) and the 600MB fixed module drive are available in both single and multiport versions.

The 5-megabyte fixed-head disk (FHD) is by definition a fixed-head device with 256 fixed heads. However, software must treat this unit as a moving-head disk with 64 cylinders and 4 heads. The fixed-head disk can be single or multiported.

A cartridge module drive (CMD) is two devices in one package. The first is the removable media and the second is the captive media. MPX-32 software dedicates the even subchannel to the removable media and the odd subchannel to the captive media. Cartridge module drives only operate in the single-port mode.

1.3 Track Format

The disk processor and IOP disk support two track formats. One format, designated F16, provides 16 data sectors where each data sector contains storage for 1024 data bytes. The second format, designated F20, provides 20 data sectors where each data sector contains 768 data bytes. While the disk processor and IOP disk are capable of supporting both track formats, only the F20 format is supported by MPX-32.

1.4 Dual Subchannel I/O

The disk processor and IOP disk firmware allow two communication paths to each device. These paths are called subchannels and occur in sequential even and odd pairs. This is to say that a device with a unit address plug of 1 has software subaddress assignments of 02 and 03. For further information, concerning XIO disk processor subaddressing refer to section 3.11.

Under MPX-32, dual-subchannel I/O is applicable only to cartridge module drives where the even subchannel is dedicated to the removable media and the odd subchannel is dedicated to the captive media. For SYSGEN purposes, this device must be assigned an even subaddress on the device directive. The odd subaddress is configured automatically by SYSGEN.

For devices other than cartridge module drives, the odd subchannel address is unusable and should never be assigned on the SYSGEN DEVICE directive.

1.5 Multiport Support for XIO Disk Processor Phase II

Multiporting allows CPUs to share a single disk drive. In order to maintain disk and system integrity, mechanisms must exist to prevent the CPUs from accessing the device at the same time. This is accomplished through device reservation and makes the device inaccessible to the nonreserving CPU. Device reservation can be implicit or explicit.

1.5.1 Implicit Device Reservation

Implicit device reservation is a disk processor function and is transparent to the operating system. If a drive is multiported, the disk processor automatically issues a reserve command to the device before initiating an I/O request. Once the I/O is complete, the disk processor issues a release command. An I/O request from the opposing CPU is postponed from the time the reserve is issued until the release is performed.

1.5.2 Explicit Device Reservation

Explicit device reservation makes a device inaccessible to the opposing CPU for a user requested period of time. The explicit device reservation is user invoked through the M.RESP service request. The device remains unavailable to the opposing CPU until the user releases the device through the M.RELP service request. When performing explicit device reservation, the release timer switch located on the disk drive must be set to the off position to disable the drive from performing its own release. This is a drive performed release which is different from the implicit disk processor release mentioned above. Also, the channel 1 and channel 2 inhibit switches located on the disk drive must be in the off position. If more than one user on the same CPU has a device explicitly reserved at the same time, the drive is not released until the last such user explicitly releases it.

1.6 System Failure in Multiport Environment

In a multiport environment, one of the systems may fail while the shared disk is reserved. If this happens, the shared disk can be accessed by an opposing processor through the J.UNLOCK system task. See Chapter 3 of the Technical Manual, Volume I.

1.7 Maximum Byte Transfer and IOCD Generation

The MPX-32 services available for user read and write requests allow for a maximum transfer of 65K bytes per request. Any larger requests are truncated to this amount.

S.IOCS40 processes read and write requests by building the necessary data chained IOCDs to span map blocks. The number of IOCDs generated for any transfer request depends on the size of the transfer and the placement buffer within the MAP block.

1.8 Hardware/Software Relationship

The disk handler consists of four parts: H.IFXIO, H.DCXIO, XIO.SUB, and the DCAs. H.IFXIO is the interrupt fielder and corresponds one for one with the channel. H.DCXIO is a system reentrant handler that processes device-dependent functions. XIO.SUB is the XIO common subroutine package the disk handler calls to perform all common XIO functions. The common XIO subroutine package is described in detail in the XIO Common Subroutines and Device Handlers chapter. Device context areas (DCAs) are areas of storage and record keeping and correspond one for one with the number of subchannels configured. They are physically located at the end of H.DCXIO.

Up to four disk drives can be connected to any IOP disk controller. The number of disk controllers configured per system is limited by the number of IOP channels and mechanical restrictions.

Up to eight disk drives can be connected to any disk processor. The number of disk processors configured per system is limited by the number of channels and cabinet space available.

2 H.DCXIO Commands

2.1 Extended I/O Commands

Extended I/O (XIO) provides channel commands for completing I/O requests. All channel commands have the following IOCD format:

	0 7	8 15	16 23	24 31
Word 1	Command code. See Note 1.	Absolute data address or TIC branch address. See Note 2.		
2	Flags. See Note 3.		Byte count.	

Notes:

1. The command code field defines the operation that is performed during command execution.
2. The absolute data address must be a 24-bit absolute address. The TIC branch address must be a 24-bit word-bounded absolute address.
3. Flag bits have the following significance:

Bit	Description
0	data chain (DC)
1	command chain (CC)
2	suppress incorrect length indication (SLI)
3	skip read data (SKIP)
4	postprogram controlled interrupt (PPCI)
5-15	zero

XIO channel commands are:

Channel Command	Hexadecimal Command Code	MPX-32 Service Call	Used by MPX Software
initiate channel (INCH)	00	none	yes
initialize controller (INC)	FF	none	yes
sense (SENSE)	04	none	yes
transfer in channel (TIC)	08	none	yes
write data (WD)	01	M.WRIT	yes
write sector label (WSL)*	31	none	no
write track label (WTL)*	51	none	no
read data (RD)	02	M.READ	yes
read sector label (RSL)*	32	none	no
read track label (RTL)	52	none	no
read angular position (RAP)*	A2	none	no
no operation (NOP)	03	none	yes
seek cylinder (SKC)	07	none	yes
format for no skip (FNSK)	0B	none	no
lock protected labels (LPL)*	13	none	no
load mode register (LMR)	1F	none	yes
reserve (RES)*	23	M.RESP	yes
release (REL)*	33	M.RELP	yes
rezero (XEZ)	37	none	yes
test star (TESS)	AB	none	no
increment head address (IHA)	47	none	no
priority override (POR)*	43	none	no
set reserve track mode (SRM)*	4F	none	no
reset reserve track mode (XRM)*	5F	none	no
read ECC (REC)	B2	none	yes

*These commands are supported only by the XIO disk processor.

2.2 Initialize Channel (INCH)

The Initialize Channel (INCH) command relays disk drive information to the disk processor, declares a buffer area, and makes the declared buffer area available to the disk processor. INCH must be the first I/O command to any channel that has a disk processor configured. INCH is performed automatically by the disk handler.

The data address specified in the INCH IOCD points to a 9-word buffer that must begin on a word boundary. The first word of this 9-word buffer must contain a 24-bit address that points to a file-bounded 224 word buffer. This buffer is used by the disk processor for record keeping. The remaining 8 words contain disk drive information for each configured drive. See Figure 2-1.

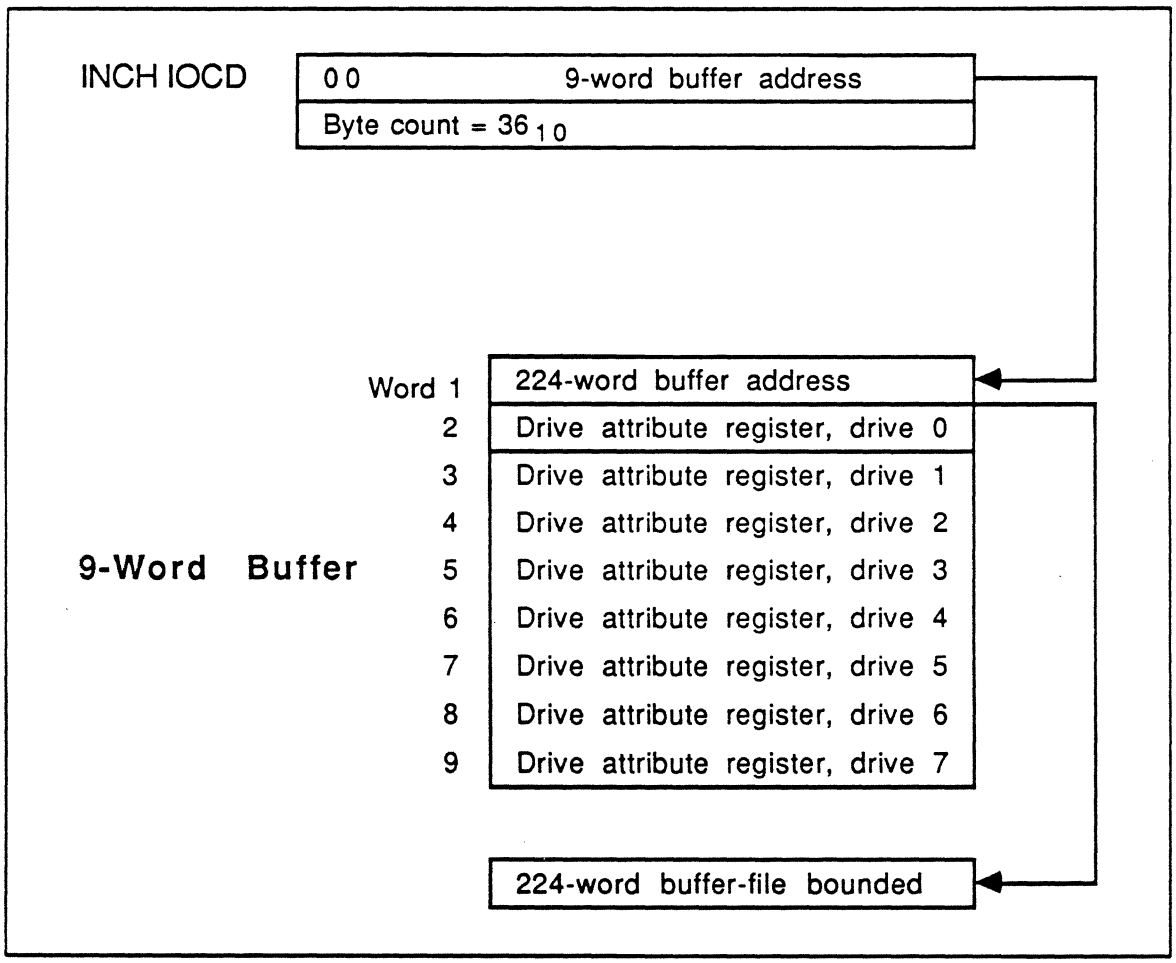


Figure 2-1
Initialize Channel I/O Command Doubleword and Buffer

Note: The above description applies only to the XIO disk processor. The IOP disk processor and floppy disk are handled differently.

2.2.1 Drive Attribute Register Layout

0 7	8 15	16 23	24 31
Flags. See Note 1	Sector count. See Note 2.	MHD count. See Note 3.	FHD count. See Note 4.

Notes:

1.

Bit	Meaning
0-1	10 = FHD 01 = MHD 11 = MHD with FHD option 00 = reserved
2	1 = cartridge module drive
3	reserved
4	1 = drive not present
5	1 = drive is dual-ported (XIO disk processor only)
6-7	zero (reserved for future use)

2. Sector count is the number of sectors per track (20 decimal).
3. MHD count is the number of heads on the MHD or the number of heads for the removable media portion of the cartridge module drive.
4. FHD count is the number of heads for the captive media portion of the cartridge module drive.

2.2.2 Generation of Drive Attribute Registers

80 MB single-port moving head disk:

DATAW X'40140500'

Value	Meaning
40	moving head disk
14	20 sectors per track
05	5 MHD count
00	zero FHD count

80 MB dual-port moving head disk:

DATAW X'44140500'

Value	Meaning
44	dual-port indication moving head disk
14	20 sectors per track
05	5 MHD count
00	zero FHD count

5 MB single-port fixed head disk:

DATAW X'40140400'

Value	Meaning
40	moving head disk
14	20 sectors per track
04	4 MHD count
00	zero FHD count

32 MB cartridge module drive:

DATAW X'20140101'

Value	Meaning
20	cartridge module drive
14	20 sectors per track
01	1 removable media count
01	1 captive media count

300 MB single port moving head disk:

DATAW X'40141300'

Value	Meaning
40	moving head disk
14	20 sectors per track
13	19 MHD count
00	zero FHD count

Drive that is not present:

DATAW X'0800000'

Value	Meaning
08	drive not present

2.3 Initialize Controller (INC)

The INC command allows the controller initialization information to be passed to the IOP disk controller. This information is four words of drive configuration data and is loaded into the drive attribute registers. The format for the drive configuration data is the same as format for the drive attribute registers.

2.4 Sense (SENSE)

The SENSE command retrieves the result of the last SIO processed by a subchannel. SENSE can also determine certain retry requirements. The results of SENSE are stored in the DCA structure associated with the device. See section 3.4. Some of the sense information generated is passed to the user. See section 3.4.5.

The following diagram shows the information returned from a sense command. This does not apply to the floppy disk. See the Line Printer/Floppy Disk Controller Technical Manual for sense information.

	0 7	8 15	16 23	24 31
Word 1	Cylinder		Track	Sector
2	Mode byte. See Note 1.	Contents of SENSE buffer register. See Note 2.		
3	Drive attribute register. See Note 3.			
4	Drive status. See Note 4.		Not used	

Notes:

1. Mode byte bit assignments are:

Bit	Function
0	one implies the drive carriage will be offset
1	effective only when bit 0 is set to one; zero implies a positive track offset and one implies a negative track offset; a positive offset is an offset toward the next higher cylinder number
2	one implies a read timing offset
3	effective only when bit 2 is set to one; zero implies that a positive read strobe timing adjustment will be used; one implies that a negative read strobe timing adjustment will be used
4	one implies diagnostic mode for error correction code (ECC) generation and checking.
5	one implies that reserved tracks can be accessed without causing an error; a zero implies that reserved track data cannot be written. IOP disks should be zero.
6	one implies the associated subchannel (SSC) will access the captive media portion of a cartridge module drive (CMD).
7	one implies the channel functions will use the RAM buffer for data operations, i.e., buffer mode is invoked. IOP disks are zero.

When all mode bits are set to the zero state, data operations occur between main memory and a moving head disk (MHD); this setting is the normal mode. A halt channel directive (HCHNL) places all channels in this mode. A halt I/O (HIO) does not change the selected subchannel's mode.

2. Sense buffer register bit assignments are:

Bit	Meaning
8	command rejected
9	intervention requested
10	spare
11	equipment check
12	data check
13	data over or under run
14	disk format error
15	defective track encountered
16	last track flag encountered
17	alternate track
18	write protection error
19	write lock error
20	mode check
21	invalid memory address
22	release fault
23	chaining error
24	lost revolution
25	disk addressing or seek error
26	buffer check
27	ECC error in sector label
28	ECC error in data
29	ECC error in track label
30	reserve track access error
31	uncorrectable ECC

Note: Bits 18-19, 21-23, 26, and 30 are not applicable for IOP disks.

3. Drive attribute register. See section 2.2.1 in this chapter for further details.

4. Drive status bit assignments are:

Bit	Meaning
0	seek end
1	unit selected
2	sector pulse counter bit 0
3	sector pulse counter bit 1
4	sector pulse counter bit 2
5	sector pulse counter bit 3
6	sector pulse counter bit 4
7	sector pulse counter bit 5
8	disk drive detected a fault
9	seek error
10	on cylinder
11	unit ready
12	write protected
13	drive is busy
14	spare
15	spare

Note: Bits 2-7 and 13 are not applicable for IOP disks.

2.5 Transfer in Channel (TIC)

The TIC command causes input/output command doubleword (IOCD) execution to continue at the address specified in the TIC command. TIC serves as a branch for IOCD execution. A TIC command cannot point to another TIC command and TIC cannot be the first command in an IOCD list. TIC is used by the handler to link IOCDs in the device context area to IOCDs in the I/O queue (IOQ). See the DCA information in section 3.4.

2.6 Write Data (WD)

The WD command is a user write request that transfers data to the disk from the address specified in the IOCD.

2.7 Write Sector Label (WSL)

The WSL command writes sector labels to the disk. WSL is not currently used by the disk handler.

2.8 Write Track Label (WTL)

The WTL command writes track labels to the disk. WTL is not currently used by the disk handler.

2.9 Read Data (RD)

The RD command transfers data from the disk to the address specified in the IOCD. RD is used in the user read request.

2.10 Read Sector Label (RSL)

The RSL command reads sector labels from the disk. RSL is not currently used by the disk handler.

2.11 Read Track Label (RTL)

The RTL command reads track labels from the disk. RTL is not currently used by the disk handler.

2.12 Read Angular Position (RAP)

The RAP command reads the sector pulse counter from the disk. RAP is not currently used by the disk handler.

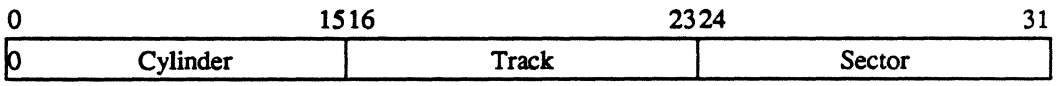
2.13 No Operation (NOP)

The NOP command is a nondata transfer command that executes without an associated disk drive. A nonzero transfer count gives incorrect length status on completion of the command.

Completed read data IOCDs within the I/O queue (IOQ) IOCD list are changed to NOP commands at entry point SI. of H.DCXIO when performing error correction code (ECC) logic.

2.14 Seek Cylinder (SKC)

The SKC command causes a disk head seek or select to the specified cylinder, track, and sector. The address specified in SKC points to a memory word which contains the following:



Entry point OP. of H.DCXIO computes the cylinder, track, and sector address for user requested reads and writes, and stores this information into the IOQ. S.IOCS12 is then called to build the seek IOCD and store it into the IOQ.

2.15 Format for No Skip (FNSK)

The FNSK command is used to format a disk. FNSK is not currently used by the disk handler.

2.16 Lock Protect Label (LPL)

The LPL command involves write lock. LPL is not currently used by the disk handler.

2.17 Load Mode Register (LMR)

The LMR command identifies a byte of information that specifies the manner in which I/O is to take place with the disk. The address specified in the input/output control doubleword (IOCD) points to this byte of information which is physically located in the I/O queue. See section 2.4 in this chapter, word 2, byte 0 of the sense information, for interpretation of the mode bits. The disk handler automatically generates LMR as the first IOCD presented for disk access user requests. LMR physically resides in the IOQ.

2.18 Reserve (RES)

The RES command causes a device to be reserved to the requesting CPU until such time as a release (REL) is issued. RES is user callable through the M.RESP service routine and is associated with dual-port operations. Execute channel programs must never include a reserve command and should use the M.RESP service routine when device reservation is desired. RES is a functional NOP for the IOP disk controller.

2.19 Release (REL)

The REL command causes a reserved device to be released by the reserving CPU. The release is not issued if more than one task has the device reserved. REL is user callable through the M.RELP service routine and is associated with dual-port operations. Execute channel programs must never include a release command and should use the M.RELP service routine when device release is desired. REL is a functional NOP for the IOP disk controller.

2.20 Rezero (XEZ)

The XEZ command is a recalibration request to the disk which resets the drive's seek logic and causes the drive to locate cylinder and track zero. Entry point SI. of H.DCXIO uses XEZ to recover from seek and drive fault errors. XEZ is in the device context area (DCA).

2.21 Test Star (TESS)

The TESS command compares the currently addressed cylinder, track, and sector to that specified by the Test Star IOCD. TESS can skip the next sequential IOCD. TESS is not currently used by the disk handler.

2.22 Increment Head Address (IHA)

The IHA command selects sector zero of the next sequential track in the associated disk drive. IHA is not currently used by the disk handler.

2.23 Priority Override (POR)

The POR command provides a mechanism for overriding and disabling dual-ported disk drive reserve functions. The drive specified in POR is reserved for the requesting channel until the channel releases the drive. POR is not currently used by the disk handler.

2.24 Set Reserve Track Mode (SRM)

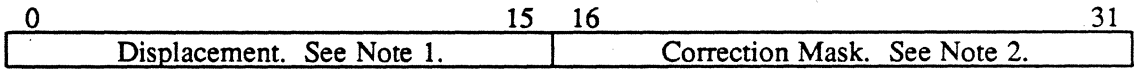
The SRM command allows all data areas designated as reserve tracks to be read or written. The reserve track mode jumper must be set on the Device Interface Adapter (DIA) board. SRM is not currently used by the disk handler.

2.25 Reset Reserve Track Mode (XRM)

The XRM command makes all data areas designated as reserve tracks unavialable for write operations. XRM is not currently used by the disk handler.

2.26 Read ECC (REC)

The REC command causes the disk processor/IOP disk to compute and present error correction information needed to recover from a disk read error. The information returned to the address specified in the REC IOCD contains:



Notes:

1. Displacement is the number of bits from the end of the last sector transferred to the last bit in the field found to contain the error.
2. Correction mask is a 9-bit mask that corrects of inaccurate memory data.

REC recovers from data errors at entry point SI. of H.DCXIO.

3 H.DCXIO Usage

3.1 CPU Instructions

The extended I/O (XIO) philosophy provides CPU instructions for accomplishing I/O requests. All CPU instructions have the following format:

0 5	6 8	9 12	13 15	16 31
Opcode. See Note 1.	Register. See Note 2.	Instruction code.	Augment code. See Note 3.	Constant. See Note 4.

Notes:

1. Bits 0-5 specify the hexadecimal operation code 0-FC.
2. Bits 6-8 specify the general register. When these bits are nonzero, the register contents are added to a constant to form the logical channel and subaddress.
3. Bits 13-15 specify the augment code up to a hexadecimal 7.
4. Bits 16-31 specify a constant that is added to the contents of the register to form the logical channel and subaddress. If the register is zero, only constant is used to specify the logical channel and subaddress.

3.2 Condition Codes

Condition codes are generated for all extended I/O instructions and indicate if the initiation of an I/O instruction was successful. For extended I/O, the four normal condition code bits are interpreted as a four bit hexadecimal number ranging from 0 to F. The following are the 16 possible condition code responses to an extended I/O instruction.

Condition code				Hexadecimal value	Meaning
CC1	CC2	CC3	CC4		
0	0	0	0	0	accepted will echo
0	0	0	1	1	channel busy
0	0	1	0	2	channel inoperable or undefined
0	0	1	1	3	subchannel busy
0	1	0	0	4	status stored
0	1	0	1	5	unsupported transaction
0	1	1	0	6	unassigned
0	1	1	1	7	unassigned
1	0	0	0	8	request accepted
1	0	0	1	9	unassigned
1	0	1	0	A	unassigned
1	0	1	1	B	unassigned
1	1	0	0	C	unassigned
1	1	0	1	D	unassigned
1	1	1	0	E	unassigned
1	1	1	1	F	unassigned

Condition code checking within the disk handler varies depending on the instruction issued.

3.2.1 SIO Instructions

The following are condition code checking and disk handler actions performed by the common XIO subroutines.

Hexadecimal instruction Codes	Meaning	Action
0	request accepted	continue normal processing
1	channel busy	delay, then reissue request
2	channel inoperative or undefined	set operator intervention bit, show error condition for FCB, abort I/O request
3	subchannel busy	exit handler, wait for interrupt
4	status stored	branch to XIO.SUB and process as if an interrupt occurred
5	unsupported transaction	show error condition for FCB, abort the I/O request
6-7	unassigned	show error condition for FCB, abort the I/O request
8	request accepted	continue normal processing
9-F	unassigned	show error condition for FCB, abort the I/O request

3.2.2 HIO Instructions

Hexadecimal instruction codes	Meaning	Action
4	status stored	branch to XIO.SUB and process as though an interrupt had occurred

No other condition codes are checked.

3.2.3 ACI, DACI, RSCTL, RSCHNL, DCI, and ECI Instructions

No condition codes are checked.

3.3 XIO CPU Instructions

Hexadecimal Instruction Codes	Description
2	start I/O (SIO)
3	test I/O (TIO)
6	halt I/O (HIO)
5	halt channel (HCHNL)
5	reset channel (RSCHNL)
4	stop I/O (STPIO)
8	reset controller (RSCTL)
C	enable channel interrupt (ECI)
D	disable channel interrupt (DCI)
E	activate channel interrupt (ACI)
F	deactivate channel interrupt (DACI)

3.3.1 Start I/O (SIO)

The SIO instruction begins I/O execution if the subchannel number is valid and the channel has no pending final status. If the channel has pending final status, the SIO instruction is rejected with a status stored condition code response. Because there is no indicator of I/O completion, the status stored response is the same as an interrupt status presentation. SIO is used by entry point IQ. of XIO.SUB.

3.3.2 Test I/O (TIO)

Test I/O (TIO) instruction tests controller status and returns appropriate condition codes and status reflecting the state of the channel and addressed subchannel. TIO is used by entry point SI. of XIO.SUB before exiting the interrupt level.

3.3.3 Halt I/O (HIO)

The HIO instruction terminates all activities in a specific subchannel at the end of its current sector. HIO does not halt I/O on a malfunctioning device. HIO does not affect subchannels other than the subchannel addressed; however, HIO generates a status stored response if status is pending in any of the channel's subchannels and it rejects the HIO instruction. Because there is no indicator of I/O completion, the status stored response is the same as an interrupt status presentation. HIO is used by entry point LI. of XIO.SUB to recover from I/O requests that time out.

3.3.4 Halt Channel (HCHNL) and Reset Channel (RSCHNL)

The HCHNL and reset channel (RSCHNL) instructions are the same and terminate all activity in the channel. Before issuing HCHNL or RSCHL, an INCH command must be performed. See section 2.2 in this chapter for further details. The RSCHNL instruction is used by the initialization entry point of H.IFXIO.

3.3.5 Stop I/O (STPIO)

The STPIO instruction performs a termination of an IOCD list by stopping IOCD execution at the completion of the current IOCD. STPIO applies only to the addressed subchannel; however, if there is pending status for any subchannel associated with the addressed channel, STPIO is not executed and a status stored condition code response is returned. Because there is no indicator of I/O completion, the status stored response is the same as an interrupt status presentation. STPIO is not used by the disk handler.

3.3.6 Reset Controller (RSCTL)

The RSCTL instruction causes the addressed subchannel to immediately terminate its I/O operation. If the subchannel is in a hung condition, the device is reset so that I/O operations can resume. RSCTL is always accepted, never generates a status stored response, and never generates an interrupt. RSCTL is used at initialization entry point of H.IFXIO before issuing the INCH command.

3.3.7 Enable Channel Interrupt (ECI)

The ECI instruction causes the addressed channel to enable request interrupts from the CPU. ECI is used at initialization entry point of H.IFXIO after issuing the INCH command.

3.3.8 Disable Channel Interrupt (DCI)

The DCI instruction causes the addressed channel to disable requesting interrupts from the CPU. DCI is used at initialization entry point of H.IFXIO before issuing the INCH command.

3.3.9 Activate Channel Interrupt (ACI)

The ACI instruction causes the addressed channel to begin actively contending with other interrupt levels. This prevents the addressed channel level and all lower priority levels from requesting an interrupt. ACI is used by XIO.SUB to protect certain sensitive code paths.

3.3.10 Deactivate Channel Interrupt (DACI)

The DACI instruction causes the addressed channel to remove its interrupt level from contention. DACI is used by XIO.SUB before entry points SI. and IQ. exiting.

3.4 Related Data Structures

This section outlines the data structures used by the disk handler. Information on the following data structures is located in the MPX-32 Technical Manual Volume I, Chapter 2:

- I/O queue (IOQ)
- unit definition table (UDT)
- controller definition table (CDT)
- file control block (FCB)
- file assignment table (FAT)

3.4.1 Device Context Area (DCA)

A DCA is a data structure that exists for each subchannel and serves as a storage area for subchannel and subchannel operation information. The DCA contains a common section and a device-dependent section. See the MPX-32 Technical Manual, Volume I, Chapter 2 for a description of the common section.

Table 3-1
XIO Device-Dependent Disk Information

Word	Hex Byte	0 15 16 31
36-37	90	Rezero IOCD used for error retry (DCA.REZO)
38-39	98	TIC IOCD used with rezero (DCA.TIC)
40-41	A0	Load mode IOCD prototype (DCA.LMOD)
42-43	A8	Read ECC IOCD (DCA.RECC)
44	B0	ECC data buffer (DCA.ECC)
45	B4	Number of ECC corrections this device (DCA.ECNT)
46	B8	Bit displacement remainder (DCA.BITD) for ECC | Byte displacement for ECC (DCA.BYTD)
47	BC	Current IOCD address (DCA.A1)
48	C0	Last buffer address for previous IOCD (DCA.A2)
49	C4	Start buffer address for current IOCD (DCA.A3)
50	C8	Address of end of erring sector (DCA.A4)
51	CC	Address of erring halfword (DCA.A5)
52	D0	Number of bytes transferred for current IOCD (DCA.B1)
53	D4	End of current IOCD buffer (DCA.B2)
54	D8	Status save cell for ECC logic (DCA.WST3)
55	DC	Status save cell for ECC logic (DCA.WST4)
56	E0	Sector/cylinder for disk (DCA.SCYL)
57	E4	EOF buffer for nondata transfer command (DCA.EOFB)
58	E8	Address of initialize controller routine (DCA.INCA)
59	EC	Not used (DCA.UNU1/DCA.UNU2)
60		
61	F4	Number of uncorrectable I/O errors this device (DCA.UREC)
62	F8	NOP IOCD for error retry (DCA.NOP)
63		
64	100	NOP TIC IOCD for error retry (DCA.NOPT)

3.4.2 Status Doubleword

A status doubleword is used each time an interrupt is generated or as a result of the status stored response to a SIO or HIO instruction. It has the following format:

	0 — 8	9 — 15	16 — 31
Word 1	Subchannel. See Note 1.	IOCD Address. See Note 2.	
2	Status. See Note 3.		Residual Byte Count. See Note 4.

Notes:

1. Subchannel is the subchannel address of interrupting device.
2. IOCD address points 8 bytes past the last IOCD processed.
3. Status bits are defined as follows:

Bit	Definition
0	ECHO (ECHO)
1	program controlled interrupt (PCI)
2	incorrect length (IL)
3	program check (PCK)
4	channel data check (CDC)
5	channel control check (CCC)
6	interface control check (ICC)
7	chaining check (CC)
8	device busy (DB)
9	status modified (SM)
10	controller end (CNE)
11	attention (ATT)
12	channel end (CE)
13	device end (DE)
14	unit check (UC)
15	unit exception (UE)

4. Residual byte count is the number of bytes not transferred for the last IOCD processed.

3.4.3 Input/Output Control Doubleword (IOCD)

See section 2.1 in this chapter for information on the input/output control doubleword.

3.4.4 Sense Buffer

See section 2.4 in this chapter for information on the SENSE buffer.

3.4.5 INCH Buffer

See section 2.2 in this chapter for information on the INCH buffer.

3.4.6 Status Returned to User's FCB

The handler places the following information into the IOQ. This information is relayed to the user's file control block (FCB) by IOCS. Not all sense information is returned to the user by the handler.

3.5 Handler Entry Points

See XIO Common Subroutine Package and Handlers (H.XIOS) in the MPX-32 Technical Manual, Volume II, the XIO Common Subroutines and Device Handlers Chapter for a description of XIO device-dependent entry points.

3.6 Error Processing for Conventional I/O Requests

When an I/O operation completes, the 16 status bits presented in the status doubleword are checked for error conditions. If only channel end (CE) and device end (DE) are presented, the I/O operation is considered complete with no errors and normal post-access processing is continued. If other bits are found to be set, the common routine issues a sense command for additional information about the error. The sense information is stored in the device context area (DCA). See section 2.4. The status bits, sense bits, and drive status bits are then interrogated to determine the appropriate action as shown in Figure 3-1.

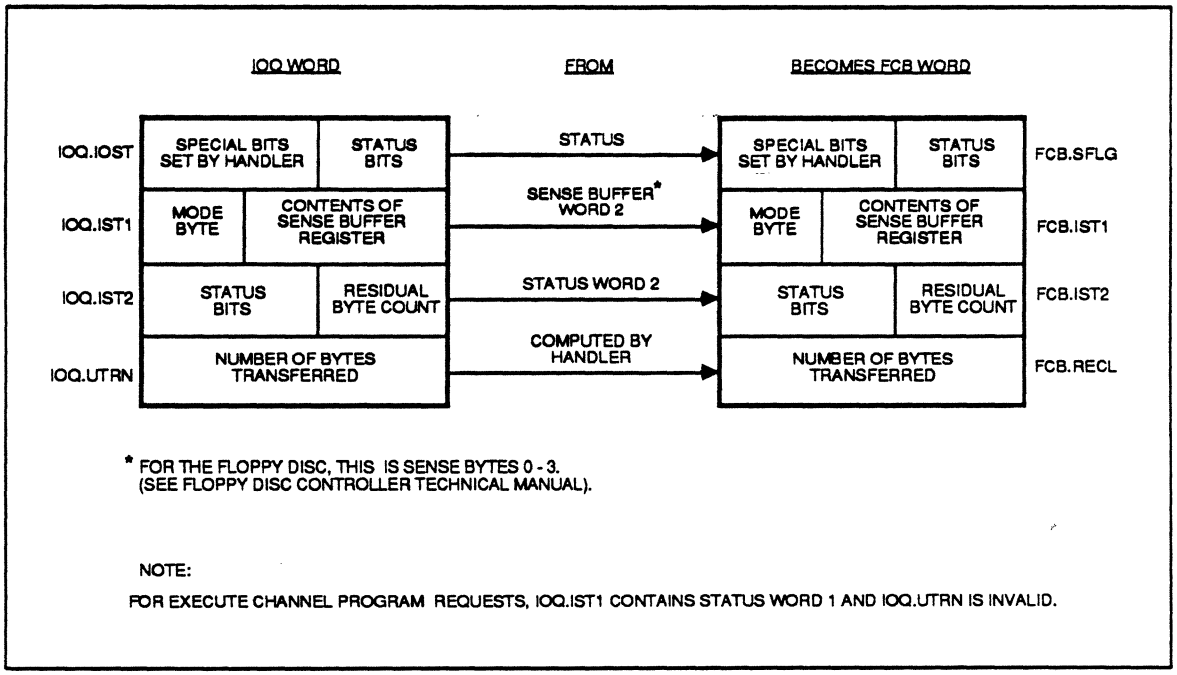


Figure 3-1
Status Returned to User's FCB

3.7 XIO/IOP Disk Error Processing

3.7.1 Abort the I/O Request

The following status bits, sense bits, and drive status bits abort the I/O request:

Status Bit	Meaning
3	program check
4	channel data check
5	channel control check
6	interface control check
7	chaining check

Sense Bit	Meaning
8	command reject
9	operator intervention required*
10	spare
11	equipment check
12	data check
14	disk format error
18	write protect error**
19	write lock error
20	mode check
21	invalid memory address
23	chaining error
30	reserve track access error

Drive Status Bit	Meaning
12	write protected**

* This condition also causes the device inoperable bit (4) in word 3 of the FCB to be set.

** If I/O request is a write, this condition also causes the write protect violation bit (3) in word 3 of the FCB to be set.

The above errors set the error condition found bit (1) in word 3 of the FCB. This indicates to the user that the I/O operation completed abnormally. I/O requests that time-out are aborted with the time-out bit set in the FCB.

3.7.2 Retry the I/O Request

The following sense bits cause five retries of the entire IOCD list. If this fails, rezero and retry are performed.

Sense Bit	Meaning
13	data over or under run
14	disk format error
27	ECC error in sector label
29	ECC error in track label
31	uncorrectable ECC

3.7.3 Perform Read ECC Correction Logic

The following sense bit causes ECC logic to be performed:

Sense Bit	Meaning
28	ECC error in data

When an ECC error in data is detected, the read ECC command is issued to obtain correction information. The information is invalid if the status returned from the read ECC command contains other than channel end (CE) and device end (DE) or if the bit displacement exceeds the sector size. If the information is invalid, the error correction logic is bypassed and error retry is initiated as per section 3.7.2.

When the correction information is valid, the bit displacement locates the address of the erring bits. If the address is outside the user's buffer, no error correcting takes place and the I/O request is considered complete without error. If the address is within the user's buffer, the data is corrected. The IOCD list is then modified to begin data transfer from the point of interruption and the I/O transfer is continued.

3.7.4 Rezero and Retry

The following sense bit and drive status bits cause drive recalibration and retry of the whole IOCD list (maximum of 5 times):

Sense Bit	Meaning
25	disk addressing or seek error

Drive Status Bit	Meaning
8	disk drive detected a fault
9	seek error

If error retry is unsuccessful, the error condition found bit (1) in word 3 of the FCB is set. This indicates that the I/O operation completed abnormally. The sense bits and drive status bits not listed do not affect normal postaccess processing.

3.8 Floppy Disk Error Processing

3.8.1 Abort the I/O Request

The following status bits and sense bits abort the I/O request:

Status Bit	Meaning
3	program check
4	channel data check
5	channel control check
6	interface control check
7	chaining check

Sense Bit	Meaning
0	command reject
1	intervention requested
3	equipment check
6	illegal address
9	write protected

The above errors set the error condition found bit (1) in word 3 of the FCB indicating that the I/O operation completed abnormally. I/O requests which time-out are aborted with the time-out bit set in the FCB.

3.8.2 Retry the I/O Request

The following sense bits cause five retries of the whole IOCD list. If this fails, rezero and retry is performed.

Sense Bit	Meaning
2	bus out check
4	data check
7	ID address mark
8	illegal STAR/format register
13	overrun/underrun
15	ID address mark

3.8.3 Rezero and Retry

The following sense bit causes drive recalibration and retry of the whole IOCD list, maximum of five times:

Sense Bit	Meaning
5	overrun

If error retry is unsuccessful, the error condition found bit 1 in word 3 of the FCB is set indicating that the I/O operation completed abnormally.

3.9 Error Processing for Execute Channel Program Requests

It is not possible to perform error processing for execute channel programs at the handler level. Information returned consists of status words 1 and 2 passed to FCB words 11 and 12. If bits other than channel end (CE) and device end (DE) are present, bit 1, error condition found, of word 3 in the FCB is set. Bits 16-31 of word 3 are valid. Sense information is returned as described in the MPX-32 Reference Manual Volume I, Chapter 5. Error correction and error retry are the responsibility of the user.

3.10 SYSGEN Considerations

SYSGEN CONTROLLER and DEVICE directives are used to define XIO disks configured in an MPX-32 system. See Volume III of the MPX-32 Reference Manual for details.

3.11 XIO Disk Processor/IOP Disk Processor Subaddressing

Each disk drive attached to an XIO disk processor or IOP disk processor is assigned a unique device subaddress. This device subaddress is determined by a plug installed in the drive or by switches contained in the drive. Plug or switch values range from 0 to 7. The device subaddress specified in the SYSGEN DEVICE directive is the plug or switch value (0 to 7) multiplied by two and converted to its hexadecimal equivalent. For example, plug or switch value 7 is specified as 'E'. Therefore, only even subaddresses may be specified in SYSGEN DEVICE directives. If more than one device is specified in the directive, the increment field (INC) must be specified and must be an even number.

Cartridge module drives also conform to the above rules. Because a cartridge module drive is two drives in one cabinet, the captive media portion of the drive is automatically configured by SYSGEN to be the odd subaddress following the even subaddress (removable medium) specified in the DEVICE directive. MPX-32 treats the captive media as a fixed head disk (DF). All direct references to the captive media portion of a cartridge module drive must specify fixed head disk (DF) as the device type code.

3.12 Floppy Disk Subaddressing

Valid floppy disk subaddresses are limited to 0 and 1. Either one or two SYSGEN DEVICE directives can be used when specifying floppy disks. If one DEVICE directive is used to configure two devices, the count field (n) of the DEVICE directive must be two and the increment field (INC) must be one or omitted.

3.13 Sample XIO Disk Processor SYSGEN Directives

```
1. CONTROLLER=DM08,PRIORITY=09,CLASS=F,MUX=XIO,-
   HANDLER=(H.IFXIO,I)
2. DEVICE=00,DTC=DM,HANDLER=(H.DCXIO,S),DISC=MH080
3. DEVICE=(02,2,2),DTC=DM,DISC=MH300
4. DEVICE=06,DTC=DM,DISC=CD032
5. DEVICE=(08,2,2),DTC=DM,DISC=CD032
6. DEVICE=0C,DTC=DM,DISC=(MH080,D)
```

1. The `CONTROLLER` directive specifies an `F`-class `XIO` disk processor on channel `08` at priority level `09`. The handler name (interrupt fielder) is `H.IFXIO` and is channel reentrant, one copy per channel.
2. This `DEVICE` directive specifies an `80MB` moving head disk assigned to subaddress `00`. The handler is `H.DCXIO` and is system reentrant (one copy per system).
3. This `DEVICE` directive specifies two `300MB` moving head disks assigned to subaddresses `02` and `04`.
4. This `DEVICE` directive specifies a `32MB` cartridge module drive whose removable media is assigned to subaddress `06`. The captive media is assigned to subaddress `07` automatically by SYSGEN.
5. This `DEVICE` directive specifies two `32MB` cartridge module drives whose removable media are assigned subaddresses `08` and `0A`. Their captive media are assigned subaddresses `09` and `0B` automatically by SYSGEN.
6. This `DEVICE` directive specifies an `80MB` dual ported moving head disk assigned to subaddress `0C`.

3.14 Sample IOP Disk Processor SYSGEN Directives

```
1. CONTROLLER=DM7E,PRIORITY=13,CLASS=F,MUX=IOP,SUBCH=E,-
   HANDLER=(H.IFXIO,I)
2. DEVICE=E0,DTC=DM,HANDLER=(H.DCXIO,S),DISC=MH080
3. DEVICE=E2,DTC=DM,DISC=CD032
```

1. The CONTROLLER directive specifies an F-class IOP disk processor on channel 7E at priority level 13. The handler name, interrupt fielder, is H.IFXIO and is channel reentrant, one copy per channel.
2. This DEVICE directive specifies an 80MB moving head disk assigned to controller address E and device subaddress 0. The handler is H.DCXIO and is system reentrant, one copy per system.
3. This DEVICE directive specifies a 32MB cartridge module drive assigned to controller address E. The removable media is assigned to device subaddress 2 and the captive media is assigned to device subaddress 3 automatically by SYSGEN.

3.15 Sample Floppy Disk SYSGEN Directives

```
1. CONTROLLER=LF7E,PRIORITY=13,CLASS=F,MUX=IOP,SUBCH=F,-
   HANDLER=(H.IFXIO,I)
2. DEVICE=F0,DTC=FL,HANDLER=(H.DCXIO,S),DISC=FL001
3. DEVICE=F1,DTC=FL,DISC=FL001
                or
4. DEVICE=(F0,2,1),DTC=FL,HANDLER=(H.DCXIO,S),DISC=FL001
```

1. The CONTROLLER directive specifies an F-class IOP line printer/floppy disk on channel 7E at priority 13. The handler name, interrupt fielder, is H.IFXIO and is channel reentrant, one copy per channel.
2. This DEVICE directive specifies a floppy disk assigned to controller address F and device subaddress 0. The handler is H.DCXIO and is system reentrant, one copy per system.
3. This DEVICE directive specifies a floppy disk assigned to controller address F and device subaddress 1.
4. This DEVICE directive specifies two floppy disks assigned to controller address F. The floppy disks are assigned device subaddresses of 0 and 1. The handler name is H.DCXIO and is channel reentrant, one copy per system.

High Speed Disk Handler (H.DPXIO)

MPX-32 Technical Manual

Volume II

Contents

Page

Page

List of Figures

List of Tables

1 H.DPXIO Overview

1.1 General Information

The high speed disk handler (H.DPXIO) provides support for high speed disk processors (HSDPs) connected to an MPX-32 based CONCEPT/32 computer.

H.DPXIO supports:

- any number and mix of extended I/O disk drives
- I/O control system (IOCS) callable I/O service requests as described in Volume I of the MPX-32 Reference Manual
- direct channel I/O with the execute channel program (EXCPM) request as described in Volume I of the MPX-32 Reference Manual

The handler consists of the common XIO subroutine package (XIO.SUB), the device handler logic (H.DPXIO), the interrupto fielder (H.IFXIO), and the device context area (DCA). Figure 1-1 presents a simplified overview of the relationship between these components and the operating system.

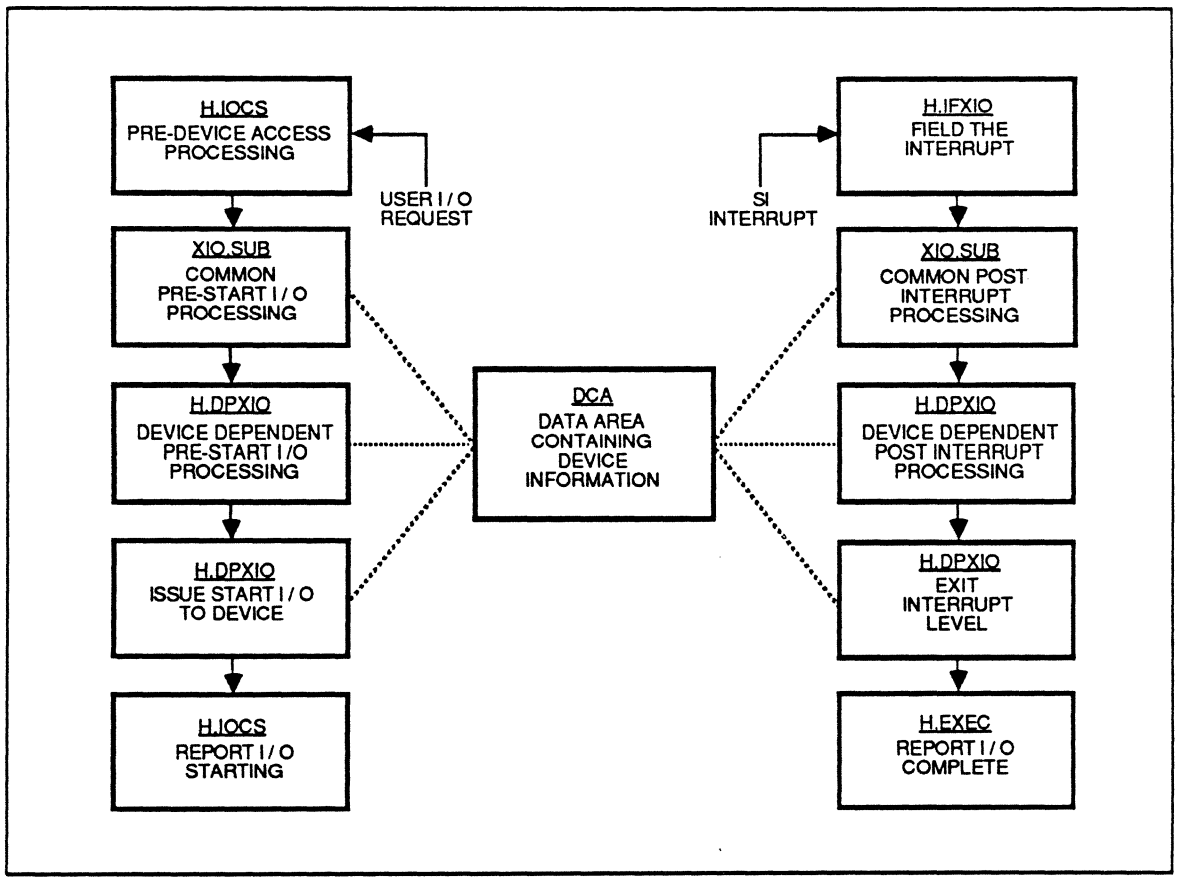


Figure 1-1
Simplified Software Block Diagram

1.2 Disks Supported

The HSDP supports single and dual ported versions of the following disks:

Manufacturer	Encore ID #	Type	Sectors	Heads	Data Cylinders	Maximum Formatted Byte Capacity	SYSGEN Device Code
CDC	8138	RSD	22	5	823	69,527,040	any
Fujitsu	8884	FMD	30	20	842	400,926,720	any
Fujitsu	8887	FMD	45	10	823	284,428,800	any
CDC	8888	XMD	54	16	1064	732,168,192	any
Fujitsu	8889	FMD	45	20	842	581,990,400	any

CDC — Control Data Corporation
RSD — removable storage drives
XMD — extended module drives
FMD — fixed module drives

1.3 Track Format

The HSDP supports variable track formats, where each data sector contains 768 data bytes. Allocation units are a factor of the track sectoring.

1.4 Dual Subchannel I/O

The HSDP firmware allows two communication paths to each device. These paths are called subchannels and occur in sequential even and odd pairs. For example, a device with a unit address plug of 1 has software subaddress assignments of 02 and 03.

1.5 Dual Port Support for HSDPs

Dual porting allows CPUs to share a disk drive. To maintain disk and system integrity, the CPUs cannot access the device at the same time. This is accomplished through device reservation, which makes the device inaccessible to the nonreserving CPU. Device reservation can be implicit or explicit.

1.5.1 Implicit Device Reservation

Implicit device reservation is a HSDP function that is transparent to the operating system. If a drive is dual ported, the HSDP automatically issues a reserve command to the device before initiating an I/O request. Once the I/O is complete, the HSDP issues a release command. An I/O request from the opposing CPU is postponed from the time the HSDP issues the reserve until the release is performed. For more information about the reserve and release commands, refer to section 2.19 and 2.20.

1.5.2 Explicit Device Reservation

Explicit device reservation makes a device inaccessible to the opposing CPU for the requested time period. The explicit device reservation is invoked through the M.RESP service request. The device remains unavailable to the opposing CPU until the device is released with the M.RELP service request. When performing explicit device reservation, the release timer switch on the disk drive must be off to disable the drive from performing its own release. This is a drive-performed release, which is different from the implicit HSDP release. Also, the channel 1 and channel 2 inhibit switches on the disk drive must be off.

If more than one user on the same CPU has a device explicitly reserved at the same time, the drive is not released until the last such user explicitly releases it.

1.6 System Failure in Dual Port Environment

In a dual-port environment, a system may fail while the shared disk is reserved. If this happens, an opposing processor can access the shared disk through the J.UNLOCK system task. For more information, refer to Chapter 3 of the Technical Manual, Volume I.

1.7 Maximum Byte Transfer and I/O Command Doubleword Generation

The MPX-32 services available for user read and write requests allow for a maximum transfer of 65K bytes per request. Any larger requests are truncated to 65K bytes.

S.IOCS40 processes read and write requests by building the necessary data-chained I/O command doublewords (IOCDs) to span map blocks. The number of IOCDs generated for a transfer request depends on the size of the transfer and the placement buffer within the MAP block.

1.8 Hardware/Software Relationship

The handler consists of H.IFXIO, H.DPXIO, XIO.SUB, and the device context areas (DCAs). H.IFXIO is the interrupt fielder and corresponds one-to-one with the channel. H.DPXIO is a system reentrant handler that processes device-dependent functions. XIO.SUB is the extended I/O common subroutine package the handler calls to perform all extended I/O functions. DCAs are areas of storage and record keeping and correspond one-to-one with the number of subchannels configured. They are physically located at the end of H.DPXIO.

Up to eight disk drives can be connected to any HSDP. (Four disk drives are normally supported; however, there is an eight drive option available.) The number of HSDPs configured per system is limited by the number of channels and cabinet space available.

2 H.DPXIO Commands

2.1 Extended I/O Commands

Extended I/O provides channel commands for completing I/O requests. All channel commands have the following I/O command doubleword (IOCD) format:

	0 7	8 15	16 23	24 31
Word 1	Command code. See Note 1.	Absolute data address or transfer in channel (TIC) branch address. See Note 2.		
2	Flags. See Note 3.		Byte count. See Note 4.	

Notes:

1. The command code defines the operation that is performed during command execution.
2. The absolute data address must be a 24-bit absolute address. The TIC branch address must be a 24-bit, word-bounded, absolute address.
3. The flags are defined as follows:

Bit	Description
0	data chain (DC)
1	command chain (CC)
2	suppress incorrect length (SIL)
3	skip read data (SKIP)
4	post program-controlled interrupt (PPCI)
5-15	zero

4. The byte count specifies the amount of data (in bytes) to transfer.

Following are the extended I/O channel commands:

Channel Command	Hexadecimal Command Code	MPX-32 Service Call	Used by MPX Software
Initiate channel (INCH)	00	none	yes
Initialize controller (INC)	FF	none	yes
Sense	04	none	yes
Transfer in channel (TIC)	08	none	yes
Write data (WD)	01	M.WRIT	yes
Write sector label (WSL)	31	none	no
Write track format (WTF)	41	none	no
Write track label (WTL)	51	none	no
Read data (RD)	02	M.READ	yes
Read express bus with ECC (RE)	12	none	no
Read express bus with no ECC (RENO)	22	none	no
Read sector label (RSL)	32	none	no
Read vendor label (RVL)	42	none	no
Read track label (RTL)	52	none	no
Read angular position (RAP)	A2	none	no
No operation (NOP)	03	none	yes
Seek cylinder, track, and sector (SKC)	07	none	yes
Format track (FMT)	0B	none	no
Load mode register (LMR)	1F	none	yes
Reserve (RES)	23	M.RESP	yes
Release (REL)	33	M.RELP	yes
Rezero (XEZ)	37	none	yes
Increment head address (IHA)	47	none	no
Priority override (POR)	43	none	no
Read ECC (REC)	B2	none	yes

2.2 Initialize Channel (INCH)

The INCH command transfers disk drive information to the HSDP, declares a buffer area, and makes the declared buffer area available to the HSDP. INCH must be the first IOCD to any channel that has an HSDP configured. INCH is performed automatically by the handler.

The data address specified in the INCH IOCD points to a 9-word buffer that must begin on a word boundary. The first word of this 9-word buffer must contain a 24-bit address that points to a file-bounded, 224-word buffer. The HSDP uses this buffer for record keeping. The remaining 8 words contain disk drive information for each configured drive. See Figure 2-1.

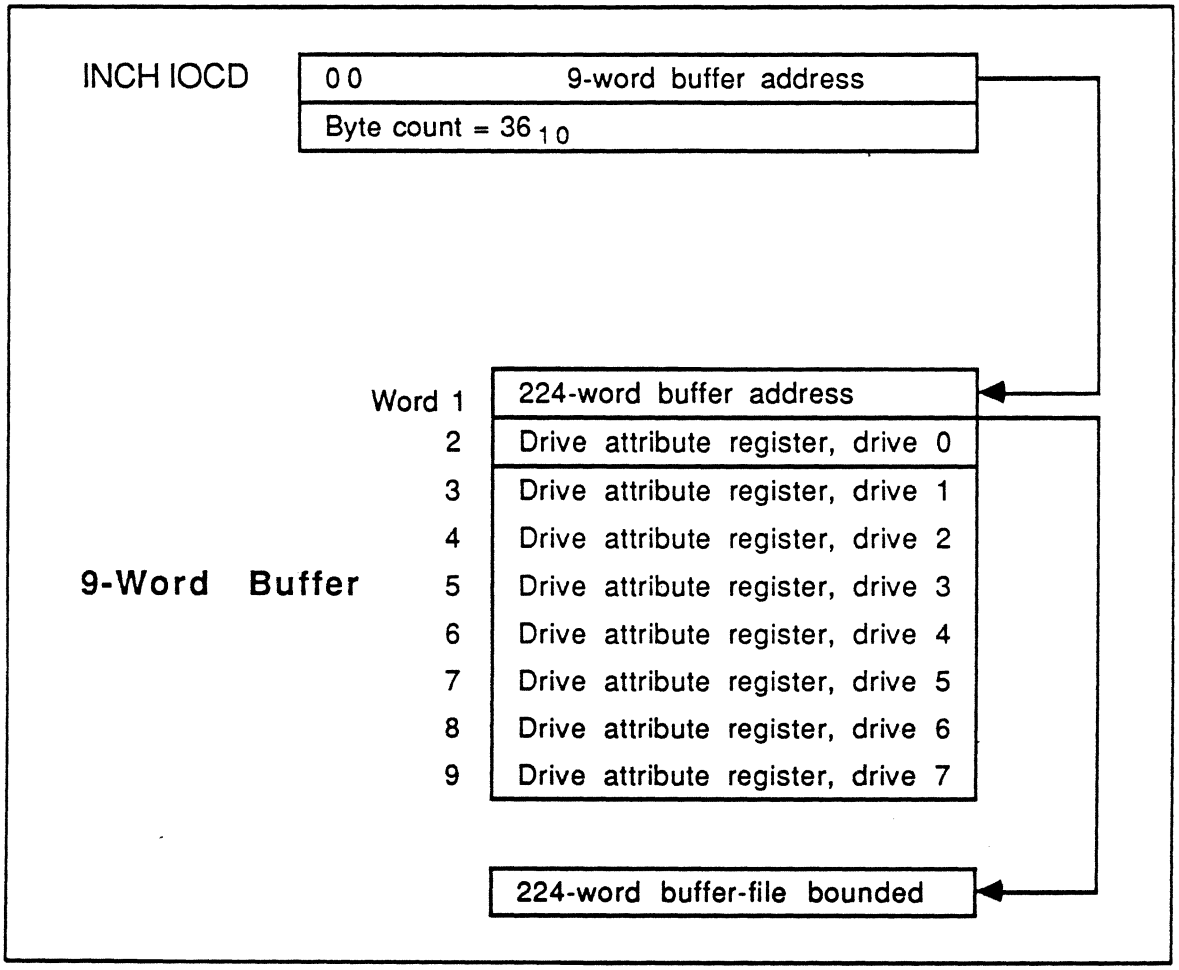


Figure 2-1
INCH IOCD and Buffer

Following is the format of the drive attribute register:

0 7	8 15	16 23	24 31
Flags. See Note 1.	Sector count. See Note 2.	MHD count. See Note 3.	Reserved (zero)

Notes:

1. The flags are assigned as follows:

Bit	Meaning
0-1	drive type: 00 = reserved 01 = MHD 10 = reserved 11 = reserved
2-3	optimize seeks: 00 = optimize seeks and post IOCLs out of order 01 = optimize seeks and post IOCLs in order 10 = do not optimize seeks 11 = do not optimize seeks
4	drive present: 0 = yes 1 = no
5	dual port: 0 = no 1 = yes
6-7	sector size: 00 = 768 bytes 01 = 1024 bytes 10 = 2048 bytes 11 = reserved

2. Sector count is the number of sectors per track. This count is variable and drive-dependent.
3. MHD count is the number of heads on the drive.

The following examples show the declaration of drive attribute registers.

80 MB single-port moving head disk:

DATAW X'40160500'

Value	Meaning
40	moving head disk, optimize seeks and post out of order, drive present, not dual ported, 768 byte sector size
16	22 sectors per track
05	5 MHD count
00	reserved, zero

80 MB dual-port moving head disk:

DATAW X'64160500'

Value	Meaning
64	moving head disk, do not optimize seeks, dual ported, 768 byte sector size
16	22 sectors per track
05	5 MHD count
00	reserved, zero

Drive that is not present:

DATAW X'0800000'

Value	Meaning
08	drive not present

2.3 Initialize Controller (INC)

The INC command allows the controller initialization information to be passed to the HSDP. This information is 8 words of drive configuration data and is loaded into the drive attribute registers. The format for the drive configuration data is the same as format for the drive attribute registers.

2.4 Sense

The sense command retrieves the results of the last SIO processed by a subchannel. The sense command also determines retry requirements. The results of a sense command are stored in the DCA structure associated with the device. Some of the sense information is passed to the user. The following shows the format of the information returned from a sense command.

	0 7	8 15	16 23	24 31
Word 1	Cylinder		Track	Sector
2	Mode byte. See Note 1.	Contents of sense buffer register. See Note 2.		
3	Drive attribute register. See Note 3.			
4	Drive status. See Note 4.		Not used	

Notes:

1. Mode byte bit assignments are:

Bit	Function
0	one implies the drive carriage is offset
1	effective only when bit 0 is set to one, zero implies a positive track offset, and one implies a negative track offset a positive offset is an offset toward the next higher cylinder number
2	one implies a read timing offset
3	effective only when bit 2 is set to one, zero implies that a positive read strobe timing adjustment one implies that a negative read strobe timing adjustment
4	one implies diagnostic mode for error correction code (ECC) generation and checking
5	controls the transfer of an ID during express bus read commands, zero implies this function is enabled
6	enables auto retry according to the firmware auto retry algorithms, zero implies enabled
7	disables the subchannel from interacting with the disk drive, (for diagnostics use only) zero implies disabled

When all mode bits are zero, data operations occur between main memory and a moving head disk (MHD); this setting is the normal mode. A halt channel (HCHNL) instruction places all channels in this mode. A halt I/O (HIO) instruction does not change the selected subchannel's mode. For more information about the HIO and HCHNL instructions, refer to section 3.3.3 and 3.3.4 respectively.

2. Sense buffer register bit assignments are:

Bit	Meaning
8	command rejected
9	intervention requested
10	unit select error
11	equipment check
12	reserved (zero)
13	reserved (zero)
14	disk format error
15	defective track encountered
16	reserved (zero)
17	at alternate track
18	write protection error
19	write lock error
20	mode check
21	invalid memory address
22	release fault
23	chaining error
24	lost revolution
25	disk addressing or seek error
26	reserved (zero)
27	reserved (zero)
28	ECC error in data
29	reserved (zero)
30	reserved (zero)
31	uncorrectable ECC

3. Drive attribute register. For more information, refer to section 2.2 in this chapter.
4. Drive status bit assignments are:

Bit	Meaning
0	seek end
1	unit selected
2	sector pulse counter bit 0
3	sector pulse counter bit 1
4	sector pulse counter bit 2
5	sector pulse counter bit 3
6	sector pulse counter bit 4
7	sector pulse counter bit 5
8	disk drive detected a fault
9	seek error
10	on cylinder
11	unit ready
12	write protected
13	drive is busy
14	reserved (zero)
15	reserved (zero)

2.5 Transfer in Channel (TIC)

The TIC command causes IOCD execution to continue at the address specified in the TIC command. TIC serves as a branch for IOCD execution. A TIC command cannot point to another TIC command and cannot be the first command in an IOCD list (IOCL). The handler uses a TIC command to link IOCDs in the device context area (DCA) to IOCDs in the I/O queue (IOQ). See the DCA information in Section 3.4.

2.6 Write Data (WD)

The WD command transfers data to the disk from the address specified in the IOCD.

2.7 Write Sector Label (WSL)

The WSL command writes sector labels to the disk. WSL is not currently used by the handler.

2.8 Write Track Label (WTL)

The WTL command writes track labels to the disk. WTL is not currently used by the handler.

2.9 Read Data (RD)

The RD command transfers data from the disk to the address specified in the IOCD. RD is used in a user read request.

2.10 Read Express Bus With ECC (RE)

The RE command reads data from the disk to the express bus. If an ECC error occurs, the sector in error is not transferred to the express bus.

2.11 Read Express Bus With No ECC (RENO)

The RENO command reads data from the disk to the express bus. Any ECC errors are ignored.

2.12 Read Sector Label (RSL)

The RSL command reads sector labels from the disk. RSL is not currently used by the disk handler.

2.13 Read Vendor Label (RVL)

The RVL command reads a vendor label from the disk to memory.

2.14 Read Track Label (RTL)

The RTL command reads track labels from the disk. RTL is not currently used by the handler.

2.15 Read Angular Position (RAP)

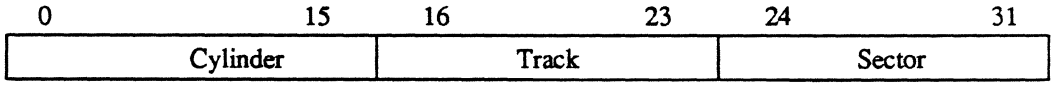
The RAP command reads the sector pulse counter from the disk. RAP is not currently used by the handler.

2.16 No Operation (NOP)

The NOP command is a non-data transfer command that executes without selecting an associated disk drive. A nonzero transfer count gives incorrect length status on completion of the command.

2.17 Seek Cylinder, Track, and Sector (SKC)

The SKC command causes a disk head seek or select to the specified cylinder, track, and sector. The address in the SKC points to a memory word which contains the following:



Entry point OP. of H.DPXIO computes the cylinder, track, and sector address for user requested reads and writes, and stores this information into the IOQ. S.IOCS12 is then called to build the SKC IOCD and store it into the IOQ. For more information about OP., refer to section 3.5.1.

2.18 Format Track (FMT)

The FMT command formats a track on the disk. FMT is not currently used by the handler.

2.19 Load Mode Register (LMR)

The LMR command points to a byte of information that specifies the mode in which I/O is to take place with the disk. The address specified in the LMR IOCD points to this byte of information which is physically located in the IOQ. For a description of the mode bits, refer to section 2.4. The handler automatically generates LMR as the first IOCD presented for disk access user requests. LMR physically resides in the IOQ.

2.20 Reserve (RES)

The RES command reserves a device to the requesting CPU until a release (REL) command is issued. RES can be called with the M.RESP service routine and is associated with dual-port operations. Execute channel program requests must never include a RES command; instead use the M.RESP service routine to reserve a device.

2.21 Release (REL)

The REL command releases a reserved device by the reserving CPU. The release is not issued if more than one task has the device reserved. REL can be called with the M.RELP service routine and is associated with dual-port operations. Execute channel program requests must never include a REL command; instead use the M.RELP service routine to release a device.

2.22 Increment Head Address (IHA)

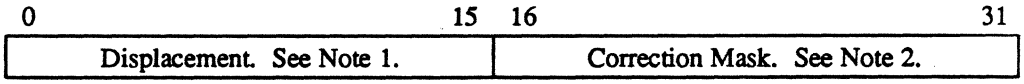
The IHA command selects sector zero of the next sequential track in the associated disk drive. IHA is not currently used by the handler.

2.23 Priority Override (POR)

The POR command overrides and disables dual-ported disk drive reserve functions. The drive specified in POR is reserved for the requesting channel until the channel releases the drive. POR is not currently used by the handler.

2.24 Read ECC (REC)

The REC command causes the HSDP to compute and present error correction information needed to recover from a disk read error. The information returned to the address specified in the REC IOCD contains:



Notes:

1. Displacement is the number of bits from the end of the last sector transferred to the last bit in the field found to contain the error.
2. Correction mask is an 11-bit mask that corrects inaccurate memory data.

REC recovers from data errors at entry point SI.A of H.DPXIO. For more information about SI.A, refer to section 3.5.6.

3 H.DPXIO Usage

3.1 CPU Instructions

Extended I/O provides CPU instructions for accomplishing I/O requests. All CPU instructions have the following format:

0 5	6 8	9 12	13 15	16 31
Opcode. See Note 1.	Register. See Note 2.	Instruction code	Augument code. See Note 3.	Constant. See Note 4.

Notes:

1. Bits 0-5 specify the hexadecimal operation code 0-FC.
2. Bits 6-8 specify the general purpose register. When these bits are nonzero, the register contents are added to a constant to form the logical channel and subaddress.
3. Bits 13-15 specify the augment code up to hexadecimal 7.
4. Bits 16-31 specify a constant that is added to the contents of R to form the logical channel and subaddress. If R is zero, this field specifies the logical channel and subaddress.

3.2 Condition Codes

Condition codes indicate if the initiation of an I/O instruction was successful. For extended I/O, the condition code bits are interpreted as a four bit hexadecimal number from 0 to F. Condition code checking within the handler varies depending on the instruction issued. The following are the 16 possible condition code responses to an extended I/O instruction.

Condition code				Hexadecimal value	Meaning
CC1	CC2	CC3	CC4		
0	0	0	0	0	unassigned
0	0	0	1	1	unassigned
0	0	1	0	2	channel inoperable or undefined
0	0	1	1	3	subchannel busy
0	1	0	0	4	unassigned
0	1	0	1	5	unsupported transaction
0	1	1	0	6	unassigned
0	1	1	1	7	unassigned
1	0	0	0	8	command accepted
1	0	0	1	9	unassigned
1	0	1	0	A	unassigned
1	0	1	1	B	unassigned
1	1	0	0	C	unassigned
1	1	0	1	D	unassigned
1	1	1	0	E	unassigned
1	1	1	1	F	unassigned

3.3 Extended I/O CPU Instructions

Table 3-1
CPU Instructions

Instruction	Hexadecimal Instruction Code
Start I/O (SIO)	2
Test I/O (TIO)	3
Halt I/O (HIO)	6
Halt channel (HCHNL)	5
Reset channel (RSCHNL)	5
Stop I/O (STPIO)	4
Reset controller (RSCTL)	8
Enable channel interrupt (ECI)	C
Disable channel interrupt (DCI)	D
Activate channel interrupt (ACI)	E
Deactivate channel interrupt (DACI)	F

3.3.1 Start I/O (SIO)

The SIO instruction begins I/O execution if the subchannel number is valid. Entry point IQ.XIO of H.DPXIO issues an SIO.

The following are the condition codes and handler actions performed in response to a SIO instruction.

Condition code (hex value)	Meaning	Action
0-1	unassigned	show error condition for FCB, abort the I/O request
2	channel inoperative or undefined	set operator intervention bit, show error condition for FCB, abort I/O request
3	subchannel busy	exit handler, wait for interrupt
4	unassigned	show error condition for FCB, abort the I/O request
5	unsupported transaction	show error condition for FCB, abort the I/O request
6-7	unassigned	show error condition for FCB, abort the I/O request
8	request accepted	continue normal processing
9-F	unassigned	show error condition for FCB, abort the I/O request

3.3.2 Test I/O (TIO)

The TIO instruction tests controller status and returns appropriate condition codes and status reflecting the state of the channel and addressed subchannel. Entry point SI.A of H.DPXIO issues a TIO instruction before exiting the interrupt level.

3.3.3 Halt I/O (HIO)

The HIO instruction terminates all activities for the specified subchannel at the end of its current sector. HIO does not halt I/O on a malfunctioning device. HIO does not affect subchannels other than the subchannel addressed; however, HIO generates a status stored response if status is pending in any of the channel's subchannels and it rejects the HIO instruction. Because there is no indicator of I/O completion, the status stored response is the same as an interrupt status presentation. Entry point LI.XIO of H.DPXIO uses a HIO instruction to recover from I/O requests that time out.

The following is the condition code and handler action performed in response to a HIO instruction.

Condition code (hex value)	Meaning	Action
4	status stored	branch to SI. processing in the handler and process as though an interrupt had occurred

3.3.4 Halt Channel (HCHNL) and Reset Channel (RSCHNL)

The HCHNL and RSCHNL instructions are the same and terminate all activity in the channel. Before issuing HCHNL or RSCHL, an INCH command must be performed. See section 2.2 in this chapter. No condition codes are checked with these instructions. The initialization entry point of H.IFXIO uses a RSCHNL instruction.

3.3.5 Stop I/O (STPIO)

The STPIO instruction performs an orderly termination of the current IOCL by stopping IOCD execution at the completion of the current IOCD. STPIO applies only to the addressed subchannel. The current IOCL operation terminates, status is posted, and all queued IOCL entries are deleted. STPIO is not used by the handler.

3.3.6 Reset Controller (RSCTL)

The RSCTL instruction causes the addressed subchannel to do an orderly termination of its I/O operation. If the subchannel is in a hung condition, the device is reset so that I/O operations can resume. RSCTL is always accepted, it never generates a status stored response, or an interrupt. No condition codes are checked with this instruction. RSCTL is used at initialization entry point of H.IFXIO before issuing the INCH command.

3.3.7 Enable Channel Interrupt (ECI)

The ECI instruction causes the addressed channel to enable request interrupts from the CPU. No condition codes are checked with this instruction. ECI is used at initialization entry point of H.IFXIO after issuing the INCH command.

3.3.8 Disable Channel Interrupt (DCI)

The DCI instruction causes the addressed channel to disable requesting interrupts from the CPU. This instruction does not clear any pending HSDP status. No condition codes are checked with this instruction. DCI is used at initialization entry point of H.IFXIO before issuing the INCH command.

3.3.9 Activate Channel Interrupt (ACI)

The ACI instruction causes the addressed channel to begin actively contending with other interrupt levels. This prevents the addressed channel level and all lower priority levels from requesting an interrupt. No condition codes are checked with this instruction. H.DPXIO uses an ACI instruction to protect certain sensitive code paths.

3.3.10 Deactivate Channel Interrupt (DACI)

The DACI instruction causes the addressed channel to remove its interrupt level from contention. No condition codes are checked with this instruction. H.DPXIO entry points SI.A and IQ.XIO use a DACI instruction before exiting.

3.4 Related Data Structures

This section outlines the data structures used by the handler. For more information about the following data structures, refer to the MPX-32 Technical Manual, Volume I, Chapter 2:

- I/O queue (IOQ)
- unit definition table (UDT)
- controller definition table (CDT)
- file control block (FCB)
- file assignment table (FAT)

3.4.1 Device Context Area (DCA)

The DCA is a data structure that stores subchannel operation information. A DCA exists for each subchannel. The DCA contains a common section and a device-dependent section. Table 3-2 lists the HSDP device-dependent disk information. Refer to the MPX-32 Technical Manual, Volume I, Chapter 2 for a description of the common section.

Table 3-2
HSDP Device-Dependent Disk Information

Word	Hex Byte	0 15 16 31
36	90	Time out (DCA.MAX)
37	94	Mode byte for drive (DCA.MODE)
38/39	98/9C	Queue associated with error (DCA.ERRQ)
40/41	A0/A4	Load mode IOCD prototype (DCA.LMOD)
42/43	A8/AC	Read ECC IOCD (DCA.RECC)
44	B0	ECC data buffer (DCA.ECC)
45	B4	Number of ECC corrections this device (DCA.ECNT)
46	B8	Bit displacement remainder for ECC (DCA.BITD) | Byte displacement for ECC (DCA.BYTD)
47	BC	Current IOCD address (DCA.A1)
48	C0	Buffer address for current IOCD (DCA.A2)
49	C4	Buffer address for last IOCD (DCA.A3)
50	C8	Buffer address for previous to last IOCD (DCA.A4)
51	CC	Unused sector area last buffer address (DCA.G1)
52	D0	Unused sector area buffer count (DCA.C1)
53	D4	Buffer count for current IOCL (DCA.C2)
54	D8	Status save cell for ECC logic (DCA.WST3)
55	DC	Status save cell for ECC logic (DCA.WST4)
56	E0	Sector/cylinder for disk (DCA.SCYL)
57	E4	EOF buffer for nondata transfer command (DCA.EOFB)
58	E8	Address of initialize controller routine (DCA.INCA)
59	EC	Buffer count for last IOCL (DCA.C3)
60	F0	Save area for subroutine return (DCA.SAVR)
61	F4	Number of uncorrectable I/O errors this device (DCA.UREC)
62	F8	Buffer count for previous to last IOCL (DCA.C4)
63-65	FC-104	3 word buffer for ECC (DCA.EBUF)

3.4.2 Status Doubleword

A status doubleword reports the result of the last executed IOCD when an I/O termination occurs. It is generated as a result of an interrupt or a status stored response to a SIO or HIO instruction. The following shows the status doubleword format:

Word	0 — 8	9 — 15	16 — 31
1	Subchannel. See Note 1.	IOCD address. See Note 2.	
2	Status. See Note 3.		Residual byte count. See Note 4.

Notes:

1. Subchannel is the subchannel address of interrupting device.
2. IOCD address points 8 bytes past the last executed IOCD.
3. Status bits are defined as follows:

Bit	Definition
0	reserved (zero)
1	reserved (zero)
2	incorrect length (IL)
3	program check (PCK)
4	channel data check (CDC)
5	reserved (zero)
6	reserved (zero)
7	chaining check (CC)
8	device busy (DB)
9	reserved (zero)
10	reserved (zero)
11	reserved (zero)
12	channel end (CE)
13	device end (DE)
14	unit check (UC)
15	unit exception (UE)

4. Residual byte count is the number of bytes not transferred for the last IOCD processed.

3.4.3 I/O Command Doubleword (IOCD)

See section 2.1 in this chapter for information about IOCDs.

3.4.4 Sense Buffer

See section 2.4 in this chapter for information about the sense buffer.

3.4.5 INCH Buffer

See section 2.2 in this chapter for information about the INCH buffer.

3.5 Handler Entry Points

The handler services I/O requests for the calling routine. (Only one copy of the handler is configured regardless of the number of devices or channels specified with the specific device type.) The handler is made part of the resident operating system and proper linkages are established by naming the handler in the DEVICE statement within the SYSGEN directive file. The handler consists of the following routines:

- opcode processor (OP.),
- I/O queue driver (IQ.XIO),
- service input processor (SI.),
- lost interrupt processor (LI.XIO),
- execute channel program opcode processor (XCHANP). For more information about this processor, refer to the XIO.SUB chapter in the MPX-32 Technical Manual, Volume II.

The HAT table is the means by which linkages are established between the I/O control system (H.IOCS) and the I/O processing routines. The handler has six entry points which are defined in the following HAT table and described in the following sections.

```
HAT     DATAW     6          number of entries in table
        ACW       OP.        opcode processor address
        ACW       IQ.XIO     I/O queue driver address
        ACW       SI.A       service interrupt processor address
        ACW       LI.XIO     lost interrupt processor address
        ACW       PX.        post transfer processor address
        ACW       SG.        SYSGEN initialization address
```

3.5.1 Entry Point OP. - Opcode Processor

Entry point OP. processes the opcode placed in the file control block (FCB) by the I/O service originally called by the user. It then indicates to H.IOCS,29 what action is to be taken. (Entry point OP. is a subroutine extension of H.IOCS,29, a portion of IOCS logic that is common to all I/O services that can initiate a physical device access.) To indicate what action is to be taken, OP. takes one of the following returns to H.IOCS,29:

```
BU        ILOPCODE    opcode is illegal for this device

BU        SERVCOMP    service complete, no device access required

BU        IOLINK      link request to IOQ
```

If return 3 (IOLINK) is taken, OP. must first:

1. call IOCS subroutine S.IOCS13 to allocate and initialize an IOQ
2. build into the IOQ entry an IOCL with the proper command codes and flags using IOCS subroutines S.IOCS12 and IOCS entry point R.IOCS,40

Entry Conditions

Calling Sequence

BL	*1W,X2	register X2 contains the address of the device-dependent handler HAT table. The one word offset from this address contains the address of OP.

Registers

R1	FCB address
R2	device-dependent handler HAT address
R3	UDT address

Exit Conditions

Return Sequence

See descriptions of ILOPCODE, SERVCOMP, and IOLINK.

Registers

R1	FCB address

3.5.2 I/O Queue (IOQ) Driver

The IOQ driver issues SIO instructions for the built IOCDs. The HSDP can queue a maximum of 32 outstanding requests per channel. It has two entry points: IQ.XIO and IQXIO.1. IQ.XIO and IQ.XIO.1 are identical except that IQ.XIO activates the interrupt level upon entry and deactivates it before exiting. IQ.XIO and IQ.XIO.1 issue an SIO instruction for the first request in the IOQ.

The following is the basic logic sequence within the IOQ queue driver with the entry points noted:

- IQ.XIO activates the interrupt level.
- IQ.XIO.1 determines which I/O request to process.
- IQ.XIO.2 performs the following:
  - checks if the device is online and functioning
  - checks for a release request, if there is a request, decrements the reserve count
  - gets various I/O request parameters from the IOQ and sets up for an SIO. These include the time-out value, the IOCL address, and the channel and subchannel.
  - issues an SIO
  - examines condition codes presented by the SIO and takes appropriate action
  - deactivates the interrupt level if entered at IQ.XIO
  - returns to the calling routine

3.5.3 Entry Point IQ.XIO

H.IOCS,29 calls entry point IQ.XIO each time H.IOCS,29 queues an I/O request and, depending on the queueing scheme, when the channel or device is not busy. It blocks external interrupts and enters IQ.XIO by the calling sequence.

Entry Conditions

Calling Sequence

BL	*2W,X2	register X2 contains the device dependent handler HAT address. The address in the HAT table points to the subroutine IQ.XIO.

Registers

R0	return address
R3	UDT address of the device to start
R7	IOQ address

Exit Conditions

Return Sequence

DACI		deactivate interrupt level
TRSW	R0	returns to calling routine

Registers

R7	IOQ address

3.5.4 Entry Point IQ.XIO.1

The SI.A entry point calls IQ.XIO.1 to drive the IOQ following completion of an I/O request. The lost interrupt processor (LI.XIO) calls IQ.XIO.1 to flush the IOQ following a HIO instruction that timed out. For more information about LI.XIO, refer to section 3.5.4. IQ.XIO.1 is entered with the interrupt level active by the calling sequence.

Entry Conditions

Calling Sequence

BL	IQ.XIO.1	if return to call is desired
(or)		
BU	IQ.XIO.1	if return is set-up

Registers

R0	return address (SI.A)
R3	UDT address of device to start

Exit Conditions

Return Sequence

TRSW	R0	R0 is properly set up prior to return if call is from LI.XIO

Registers

R1	CHT address
R2	DCA address

3.5.5 Service Interrupt Processor

The service interrupt processor performs postaccess processing associated with the device access which just completed. It has one entry point, SI.A, and two routines, SI.UNLNK and SI.EXIT, that perform the following basic logic sequence within the service interrupt processor:

- SI.A entry point:
  - determines which device caused the interrupt (status presentation)
  - determines the cause of the interrupt and branches to the appropriate action
  - performs normal device-specific postaccess processing
  - updates the actual transfer quantity (if required)
- SI.UNLNK routine:
  - unlinks the IOQ entry from the IOQ
  - deletes the IOQ if a kill request was issued

- reports I/O complete
- continues processing at SI.EXIT

- SI.EXIT routine:
  - branches to IQ.XIO.1 to continue driving the IOQ (this may or may not return)
  - tests for any more status pending and if so, branches back to SI.; otherwise, deactivates the interrupt level
  - exits from the interrupt via S.EXEC5

3.5.6 Entry Point SI.A

Entry point SI.A is entered directly from the XIO interrupt fielder program (H.IFXIO) to service the interrupts and to perform device-dependent logic. The calling sequence enters SI.A with the interrupt level active. Device-dependent service interrupt calls occur:

- when I/O completes normally or when status checking is inhibited
- when status contains channel end with no device end
- following a normal sense command (with IOQ)
- following an unexpected interrupt
- following a device time out

Entry Conditions

Calling Sequence

BU	*3W,X2	X2 is the address of the handler's HAT

Registers

R1	CHT address
R3	UDT address

Exit Conditions

Return Sequence

BU	SI.UNLNK	the I/O request is to be unlinked, reported complete, and IOQ processing continued
(or)		
BU	SI.EXIT	exit the interrupt level without unlinking the I/O request and reporting the I/O complete to initiate error retry or collect sense information.

Registers

R1	IOQ address (required for SI.UNLNK only)
R2	DCA address

3.5.7 SI.UNLNK Routine

The SI.UNLNK routine unlinks the IOQ entry from the UDT and reports I/O complete to the executive. The calling sequence enters SI.UNLNK with the interrupt level active. SI.UNLNK is also entered if a device timed out due to a kill command or if a device malfunctioned.

Entry Conditions

Calling Sequence

BU	SI.UNLNK

Registers

R1	IOQ address
R2	DCA address
R7	IQ. return address

Exit Conditions

Return Sequence

Continues with remainder of SI. logic.

3.5.8 SI.EXIT Routine

The SI.EXIT routine continues driving the IOQ and exits the interrupt level. The calling sequence enters SI.EXIT with the interrupt level active.

Entry Conditions

Calling Sequence

```
BU      SI.EXIT
```

Registers

R2 DCA address

Exit Conditions

Return Sequence

Continues with remainder of SI. logic.

3.5.9 Conditional Service Interrupt Processing

Table 3-3 lists the cause of interrupts and the response by SI.A in performing conditional service interrupt processing.

Table 3-3
Interrupts and Responses by SI.A

Cause of Interrupt	Response by SI.A
Channel end with no device end	checks for a reserve request (if a dual port disk) continues processing at SI.EXIT
Device time out	marks unrecoverable error condition in the IOQ; actual transfer count in IOQ handler continues to drive the queue
Execute channel program	sets error condition in IOQ if an error is indicated. If sense information is required, issues a sense command and continues processing at SI.EXIT; otherwise, if no error is found and no sense information is required, continues processing at SI.UNLNK.
I/O request complete with error	issues sense command and continues processing at SI.EXIT
Normal sense command with IOQ	if execute channel program was requested, continues processing at SI.UNLNK; otherwise, completes the actual transfer count computed and updates the IOQ
Rewind or seek complete	clears device rewinding or seeking bit; continues processing at SI.EXIT
Spurious interrupt when device is not configured	increments spurious interrupt count, and exits the interrupt level; continues processing at SI.EXIT
Spurious interrupt when device is configured but interrupt is not expected	increments spurious count for the device and the channel

3.5.10 Normal Completion or Status Checking Inhibited

The handler enters this routine when an I/O request completes with no errors or when status checking is inhibited. It performs any device-specific processing necessary under these conditions. An example is the collection of sense information about the I/O operation just completed. The XIO common routines collect sense information only when an I/O request produces an error or the I/O request was for an execute channel program and sense information was requested.

This routine performs the following:

- checks for a reserve request. If the request exists, the routine increments the reserve count.
- checks for an advance or backspace file request. If the request exists, the routine sets EOF and EOM or BOM and continues processing at SI.EXIT. If no request exists, the routine updates the FAT.
- computes the transfer count
- continues processing at SI.UNLNK

3.5.11 Channel End with No Device End

The handler enters this routine when an I/O request produces an interrupt whose status contains channel end and no device end. This is not a normal case except when a reserve is issued to a dual ported disk that is reserved to the opposing CPU. Any other condition is treated as an unexpected interrupt.

This routine performs the following:

- checks for a channel end from a reserve request. If there is no channel end, the routine continues processing as an unexpected interrupt.
- updates the time out in the UDT and shows the I/O as active
- continues processing at SI.EXIT

3.5.12 Normal Sense Command with IOQ

The handler enters this routine following an interrupt caused by issuing a sense command on behalf of an I/O request that completed with an error indication. This routine examines the status and sense information, initiates error recovery if applicable, and performs the following:

- sets the appropriate indicators based on the sense data
- computes the actual transfer count if an error condition is indicated and updates the IOQ transfer count
- continues processing at SI.UNLNK

3.5.13 Unexpected Interrupt

The handler enters the unexpected interrupt routine when an interrupt occurs that was not expected. This routine performs the following:

- increments spurious interrupt count for the device
- increments spurious interrupt count for the channel
- continues processing at SI.EXIT

3.5.14 Device Time Out

The device time out routine is entered by an interrupt that was generated by issuing a HIO instruction for a device that timed out. This routine performs the following:

- sets the error condition in the IOQ
- sets the time out flag in the IOQ
- continues driving the IOQ
- continues processing at SI.EXIT

3.5.15 Entry Point LI.XIO - Lost Interrupt Processor

S.IOCS5 calls LI.XIO to take corrective measures when an expected interrupt fails to occur. It is also called from H.IOCS,38 when a kill command is issued to a task and the task has I/O in progress. In both cases, the I/O request terminates with a HIO instruction. If the controller responds to the HIO, SI.A performs the required interrupt handling.

The following is the basic logic sequence within LI.XIO:

- activate the interrupt level
- increment the lost interrupt count if not a kill request
- issue the HIO if it has not already been issued
- block external interrupts
- deactivate the interrupt level
- return to calling routine
- clears outstanding I/O queue entries

If the HIO has already been issued but fails to generate an interrupt, LI.XIO is entered again and takes the following actions:

- activates the interrupt level
- increments lost interrupt count if not a kill request
- marks the device as off-line
- marks the device as malfunctioning
- unlinks the I/O request from the IOQ
- reports the I/O request as complete with errors if the HIO was issued because of an interrupt which failed to occur
- branches and links to IQ.XIO.1 to flush any pending I/O requests to the failing device
- blocks external interrupts
- deactivates the interrupt level
- returns to calling routine

Entry Conditions

Calling Sequence

S.IOCS5 and H.IOCS,38 call LI.XIO with interrupts blocked by:

BL	*4W,X1	register X1 contains the device-dependent handler HAT address. The address in the HAT points to LI.XIO.

Registers

R0	return address
R3	UDT address of device to halt

Exit Conditions

Return Sequence

TRSW	R0

Registers

None

3.5.16 Entry Point PX. - Post-Transfer Processing

S.IOCS1 calls the entry point PX. to perform processing after completion of the I/O request and before returning to the requesting task. This entry point executes at the task priority and at a low level of system overhead.

Entry Conditions

Calling Sequence

BL	*5W,X2

Registers

R1	FCB address
R2	device-dependent handler HAT address
R3	UDT address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1 FCB address

If an advance file or a backspace file was performed, update the current position in the FAT.

3.5.17 Entry Point SG. - SYSGEN Initialization

SYSGEN calls the entry point SG. to initialize certain handler parameters, device context areas (DCAs), and data structure elements during the construction of an MPX-32 image. The repeated assembly of the macro DCA.DATA creates a maximum number of DCA's. During the execution of this entry point:

- one DCA is initialized for each UDT entry containing the name of the handler. SYSGEN overlays any remaining DCA's and the remainder of the code in the handler
- the DCA is updated with the number of sectors per cylinder and the default time out for I/O
- the IOCDs for the LOAD MODE and read ECC are built and stored in the DCA

SYSGEN overlays any remaining DCAs and the remainder of the code in the handler.

Entry Conditions

Calling Sequence

The last entry point, and it is computed from information in the HAT table.

Registers

None

Exit Conditions

Return Sequence

M.XIR this is the standard handler SYSGEN exit macro

Registers

None

3.6 Error Processing for Conventional I/O Requests

When an I/O operation completes, the 16 status bits presented in a status doubleword are checked for error conditions. If only channel end and device end are presented, the I/O operation is considered complete with no errors and normal post-access processing is continued. If other bits are set, the handler issues a sense command for additional information about the error. The sense information is stored in the DCA. See Section 2.4 in this chapter for further details. The status bits, sense bits, and drive status bits are mapped to the user's FCB as shown in Figure 3-1.

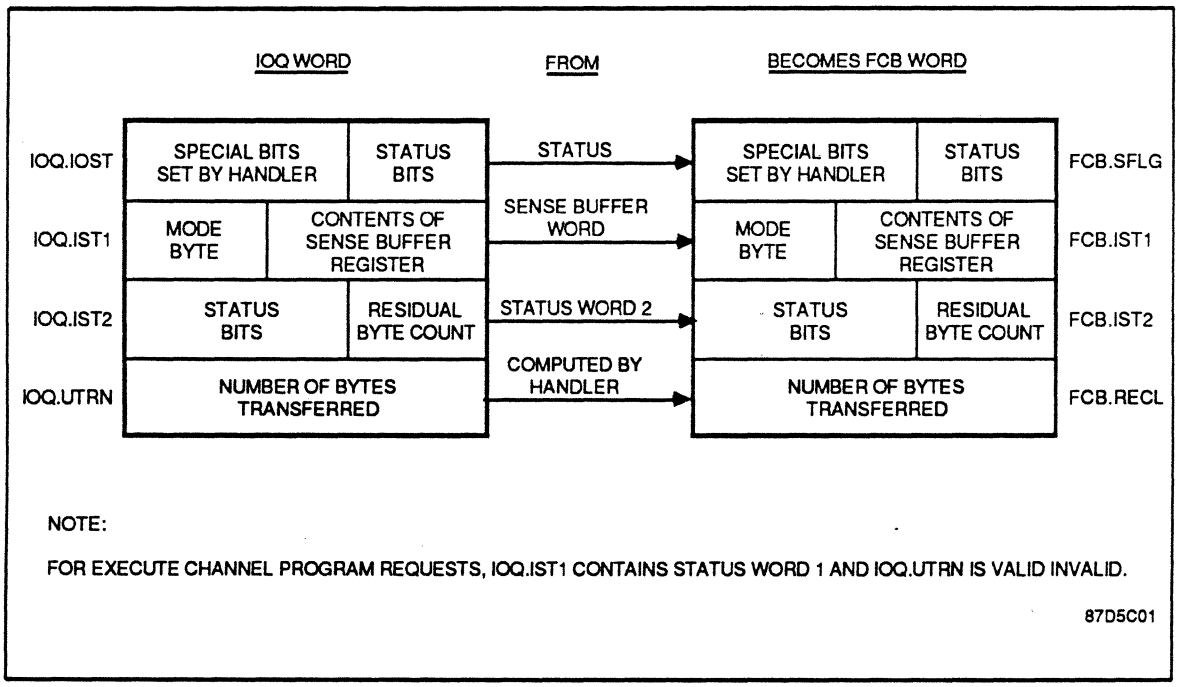


Figure 3-1
Status Returned to User's FCB

3.7 HSDP Error Processing

3.7.1 Abort the I/O Request

The status bits, sense bits, and drive status bits listed below abort the I/O request. These errors set the error condition found bit (1) in word 3 of the FCB. This indicates that the I/O operation completed abnormally. I/O requests that time-out are aborted with bit 10 in word 3 of the FCB set.

Status	Bit Meaning
3	program check
4	channel data check
5	channel control check
6	interface control check
7	chaining check

Sense	Bit Meaning
8	command reject
9	operator intervention required*
10	spare
11	equipment check
12	data check
14	disk format error
18	write protect error**
19	write lock error
20	mode check
21	invalid memory address
23	chaining error
30	reserve track access error

Drive Status Bit	Meaning
12	write protected**

* This condition also causes the device inoperable bit (4) in word 3 of the FCB to be set.

** If I/O request is a write, this condition also causes the write protect violation bit (3) in word 3 of the FCB to be set.

3.7.2 Retry the I/O Request

The following sense bits cause five retries of the IOCL. If this fails, rezero and retry are performed.

Sense Bit	Meaning
13	data overrun or underrun
14	disk format error
27	ECC error in sector label
29	ECC error in track label
31	uncorrectable ECC

3.7.3 Perform Read ECC Correction Logic

The following sense bit causes ECC logic to be performed:

Sense Bit	Meaning
28	ECC error in data

When an ECC error in data is detected, the REC command is issued to obtain correction information. The information is invalid if the status returned from the REC command contains any bits set other than channel end and device end or if the bit displacement exceeds the sector size. If the information is invalid, the error correction logic is bypassed and error retry is initiated.

When the correction information is valid, the bit displacement locates the address of the erring bits. If the address is outside the user's buffer, no error correcting takes place and the I/O request is considered complete without error. If the address is within the user's buffer, the data is corrected. The IOCL is modified to begin data transfer from the point of interruption and the I/O transfer is continued.

3.7.4 Rezero and Retry

The following sense bit and drive status bits cause drive recalibration and retry of the IOCL (maximum of 5 times):

Sense Bit	Meaning
25	disk addressing or seek error

Drive Status Bit	Meaning
8	disk drive detected a fault
9	seek error

If error retry is unsuccessful, the error condition found bit (1) in word 3 of the FCB is set. This indicates that the I/O operation completed abnormally. The sense bits and drive status bits not listed do not affect normal post-access processing.

3.8 Error Processing for Execute Channel Program Requests

It is not possible to perform error processing for execute channel programs at the handler level. Information returned consists of status words 1 and 2 passed to FCB words 11 and 12. If bits other than channel end (CE) and device end (DE) are present, bit 1, error condition found, of word 3 in the FCB is set. Bits 16-31 of word 3 are valid. Sense information is returned as described in the MPX-32 Reference Manual Volume I, Chapter 5. Error correction and error retry are the responsibility of the user.

3.9 SYSGEN Considerations

SYSGEN CONTROLLER and DEVICE directives define the HSDP disks configured in an MPX-32 system. For more information, refer to Volume III of the MPX-32 Reference Manual.

3.10 HSDP Subaddressing

Each disk drive attached to an HSDP is assigned a unique device subaddress. This device subaddress is determined by a plug installed in the drive or by switches contained in the drive. Plug or switch values range from 0 to 7. The device subaddress specified in the SYSGEN DEVICE directive is the plug or switch value (0 to 7) multiplied by two and converted to its hexadecimal equivalent. For example, plug or switch value 7 is specified as 'E'. Therefore, only even subaddresses may be specified in SYSGEN DEVICE directives. If more than one device is specified in the directive, the increment field (INC) must be specified and must be an even number.

3.11 Sample HSDP SYSGEN Directives

```
1.  CONTROLLER=DM04,PRIORITY=05,CLASS=F,MUX=XIO,HANDLER=(H.IFXIO,I)
2.  DEVICE=00,DTC=DM,HANDLER=(H.DPXIO,S),DISC=ANY
```

1. The `CONTROLLER` directive specifies an `F`-class `XIO` disk processor on channel `4` at priority level `5`. The handler name (interrupt fielder) is `H.IFXIO` and is channel reentrant, one copy per channel.
2. This DEVICE directive specifies any disk assigned to subaddress `00`. The handler is `H.DPXIO` and is system reentrant (one copy per system).

IOP Eight-Line Full Duplex Handler (H.F8XIO)

MPX-32 Technical Manual

Volume II

Contents

1 H.F8XIO Overview

1.1 General Information

The IOP Eight-Line Full Duplex Handler (H.F8XIO) is system level reentrant, i.e., only one copy is required regardless of the number of eight-line communication multiplexers configured in a system. Reentrancy is accomplished by device context areas (DCAs). The appropriate number of DCAs are initialized in the SYSGEN initialization entry point. As with all XIO device handlers, interrupts are fielded by the XIO channel executive program (H.IFXIO) and then turned over to the service interrupt entry point of H.F8XIO for processing.

2 H.F8XIO Usage

2.1 Entry Points

2.1.1 Opcode Processor (OP.)

This entry point provides the interface for processing an opcode stored in the user's file control block (FCB). Depending on the particular opcode, this processing may or may not involve device access.

The eight-line handler supports 14 IOCS opcodes enabling support of all command codes recognized by the eight-line multiplexer plus necessary standard functions, such as open. The opcodes and their functions are as follows:

OP	Call	Handler function	8 Line command
00	M.FILE	open, perform inch if necessary	--
01	M.RWND	sense status	04
02	M.READ	read. See the Options section	8E,8A,0A,02, or 06
03	M.WRIT	write. See Note 1.	0D,01,05, or FF
04	M.WEOF	NOP	03
05	SVC 1,X'25'	EXCPGM (execute channel program)	--
06	M.FWRD,R	set data terminal ready	17
07	M.FWRD	reset data terminal ready	13
08	M.BACK,R	define ACE parameters (INIT)	FF
09	M.BACK	reset request to send	1B
0A	M.UPSP	set request to send	1F
0B	SVC 1,X'3E'	set or reset break. See Note 2.	37 or 33
0C	SVC 1,X'0D'	define special character	0B
0D	M.CLSE	close	--
0E	SVC1,X'26'	set half duplex. See Note 3.	--
0F	SVC1,X'27'	set full duplex. See Note 3.	--

For an explanation of the individual command codes, see the Eight-Line Asynchronous Communications Multiplexer MPX-32 Technical Manual Volume I.

Notes:

1. If bit 1 is set in FCB.SCFG, the write is interpreted as an init and a define ACE parameters command (FF) is issued. If bit 1 in FCB.SCFG is reset and bit 3 in FCB.SCFG is set, a write with input subchannel monitoring command (05) is issued. If both bits 1 and 3 in FCB.SCFG are reset, a write command (01) is issued.
2. If bit 0 in FCB.SCFG is set, a set break command (37) is issued. If bit 0 in FCB.SCFG is reset, a reset break command (33) is issued. It is not necessary to issue a reset break after a break interrupt is received.
3. The following status is returned in the condition codes:

CC1	0 = successful
CC2	0
CC3	0
CC4	1 = previously full duplex
	or
CC1	0 = unsuccessful
CC2,CC3	00 = not SYSGENed for full duplex 01 = service not legal using the write subaddress 10 = not initialized for FULL by J.TINIT 11 = not initialized for NOECHO by J.TINIT
CC4	0

2.1.2 I/O Queue Processor (IQ.XIO)

This entry point performs standard I/O queue scheduling.

2.1.3 Service Interrupt Processor (SI.)

This entry point consists of the six standard subroutines described in the Subroutines section.

2.1.4 Lost Interrupt Processor (LI.XIO)

This entry point performs standard lost interrupt and kill I/O processing.

2.1.5 Posttransfer Processing (PX.)

This entry point performs conversion of lower case ASCII to upper case on formatted read operations. This processing can be inhibited by setting bit 10 in FCB.GCFG.

In addition, reads performed for TSM tasks must have the data copied from the operating system buffer to the user's buffer.

2.1.6 Pre-SIO Processor (PRE.SIO)

This entry point is called by XIO.SUB prior to issuing an SIO. This entry point checks for asynchronous attention interrupts on the read subaddress while a write is in progress for half duplex devices. If this is true, a break is issued to the task and the interrupt routine is exited.

2.1.7 SYSGEN Initialization Processor (SG.)

This entry point creates DCAs at assembly time and initializes them at SYSGEN time. In order to implement the full duplex capabilities of H.F8XIO, it is necessary to have two unit definition table (UDT) entries per port. For example, one UDT for read, one UDT for write. To accomplish this, SYSGEN directives must specify both the read subaddresses configured (0 through 7) and the write subaddresses configured (8 through F).

If two corresponding UDT entries are SYSGENed, all operations are requested using the read subaddress' UDT. Write operations are queued to the read subaddress' UDT for half duplex, and to the write subaddress' UDT for full duplex. If only half duplex operation is desired, only the read subaddresses need to be specified in the SYSGEN directives.

Note:

Devices SYSGENed as half duplex can still be initialized as full duplex in order to make use of full duplex commands such as read echoplex.

2.2 Options

2.2.1 Read Echoplex

This option causes a read command (02) to be issued if bit one is set in FCB.SCFG. If the bit is not set, a read echoplex command (06) is issued.

2.2.2 ASCII Control Character Detect

This option allows input to terminate whenever a control character (X'00' through X'1F') or a delete (X'7F') is detected. This option is selected by setting bit zero in FCB.SCFG before issuing the read. This option is implied in the read echoplex command.

2.2.3 Special Character Detect

This option allows input to terminate whenever a predefined 8-bit character is detected. This option is selected by setting bit three in FCB.SCFG before issuing the read.

2.2.4 Purge Input Buffer

This option specifies any input data held in the type ahead buffer is to be purged before any new incoming data. This option is selected by setting bit 4 in FCB.SCFG before issuing the read. This option is forced following a ring or break interrupt.

2.2.5 Read with Software Flow Control

This option enables XON/XOFF protocol if the UDT.RXON bit is set in UDT.BIT2, or bit 13 (FCB.RXON) of FCB.GCFG is set. The DTR line is normally used for control (command 8A), but if UDT.RDTR is not set, the RTS line is used (command 0A). As this operation uses the write subchannel for XON/XOFF transmission, Echoplex must be performed locally by the terminal device, not the controller (command 06). When used in the RTS mode, this command is the functional complement of the write with input subchannel monitor (command 05).

2.2.6 Read with Hardware Flow Control

This option enables hardware flow control using the DTR line (command 8E). It is used only when the UDT.RHWF bit is set in UDT.BIT2.

2.2.7 Write with Software Flow Control

This option enables XON/XOFF protocol if the UDT.WXON bit is set in UDT.BIT2, or bit 11 of FCB.GCFG is set. WXON (command 05), monitors the CTS line and the input subchannel for the XON/XOFF (OC1/OC3) control codes. This option was previously referred to as write with input subchannel monitoring and is a functional equivalent.

2.2.8 Write with Hardware Flow Control

This option enables hardware flow control by monitoring the CTS line (command 0D). It is used only when the UDT.WHWF bit is set in UDT.BIT2.

2.3 Subroutines

2.3.1 Normal

This subroutine is called by XIO.SUB when an I/O operation completes normally or when error processing is inhibited. Special processing is not performed unless the operation is a formatted read.

For formatted reads that are not on a TSM device, a check is made for an ETX character. If one is found, the end-of-file indicator is set.

For formatted reads that are on a TSM device, in addition to ETX, a check is made for a carriage return, backspace, tab, or delete. Appropriate actions are taken if any of these are found.

2.3.2 Unexpt

This subroutine is called by XIO.SUB when an unexpected interrupt occurs. A sense status command is issued in order to determine the reason for the interrupt and the interrupt routine is exited.

2.3.3 Snsnoioq

This subroutine is called by XIO.SUB in response to an interrupt generated by a sense command issued by the UNEXPT subroutine. The sense data is examined to determine whether the unexpected interrupt was due to a ring, break, or DSR or RLSD failure. If none of these is true, the routine is exited.

If the interrupt was due to a ring, bit UDT.LOGO is set in the UDT. If the interrupt was due to DSR or RLSD failure, bit UDT.DEAD is set in the UDT. A break is then issued to the task which has the device allocated. If another ring is expected, the task must reset UDT.LOGO.

2.3.4 Sense

This subroutine is called by XIO.SUB when an I/O operation completes in error. A sense status command is executed prior to calling this subroutine. This subroutine examines the status to determine if the error was due to DSR or RLSD failure. If so, bit UDT.DEAD is set in the UDT, and an error flag is set in the IOQ.

Next, the status is tested for attention, unit check, or unit exception. If any of these occurred, a break is issued to the task and the error bit is set. Status is then tested for channel errors. If errors are found, the error bit is set in the IOQ.

2.3.5 Cenode

This subroutine is called by XIO.SUB when channel end, but no device end, is found in the status. This condition does not occur with the 8-line asynch.

2.3.6 Timeo

This subroutine is called by XIO.SUB when an I/O operation does not complete and times out. An HIO (halt I/O) is issued prior to calling this subroutine. TIMEO processing consists of setting the error processing inhibit bit in the IOQ, returning to XIO.SUB, and performing all security checks for special processing.

2.4 Device Context Area

The following DCA fields contain information specific to H.F8XIO.

2.4.1 DCA Word 9

Equate Name

DCA.FLAG (bytes 0-2 only)

Definition

When set, bits are defined as follows:

Bit	Equate Name	Definition
0-5		defined in DCA
6-15		reserved for common subroutine usage
16	W.R	read and write configured
17	PRG	force purge type ahead buffer
18	DUAL.CH	dual channel mode
19	DCA.SMEN	a security timeout violation has occurred on this device. Set by H.F8XIO.
20	DCA.LGIN	the unit is currently being logged in on. Set by J.TSM.

2.4.2 DCA Word 28

Equate Name

DCA.STMP

Definition

Time stamp for current read operation to the appropriate terminal. If a timeout occurs before the system time is greater than or equal to this value, it is not treated as a security violation. This allows forced I/O timeouts without causing the terminal to be logged off.

2.4.3 DCA Word 29

Equate Name

DCA.SEC

Definition

Bytes defined as follows:

Byte	Equate Name	Definition
0	DCA.LOGC	Maximum number of failed logon attempts allowed for this device. Range 0 through 255. Set by J.TINIT.
1	DCA.LOGT	Timeout value in seconds for a read during login. Range 0 through 255. Set by J.TINIT.
2,3	DCA.TRMT	Timeout value in seconds for a read during a normal terminal session. Range 0 through 59940. Set by J.TINIT.

2.4.4 DCA Word 34

Equate Name

DCA.RA

Definition

Save return address across S.EXEC13.

2.4.5 DCA Word 35

Equate Name

DCA.WOSB

Definition

Save operating system buffer word count.

2.4.6 DCA Word 36

Equate Name

DCA.TSMP

Definition

Save TSM SI process address.

General Purpose Multiplexer Support (H.GPMCS)

MPX-32 Technical Manual

Volume II

Contents

1 H.GPMCS Overview

1.1 General Information

The General Purpose Multiplexer Controller (GPMC) is structured as follows:

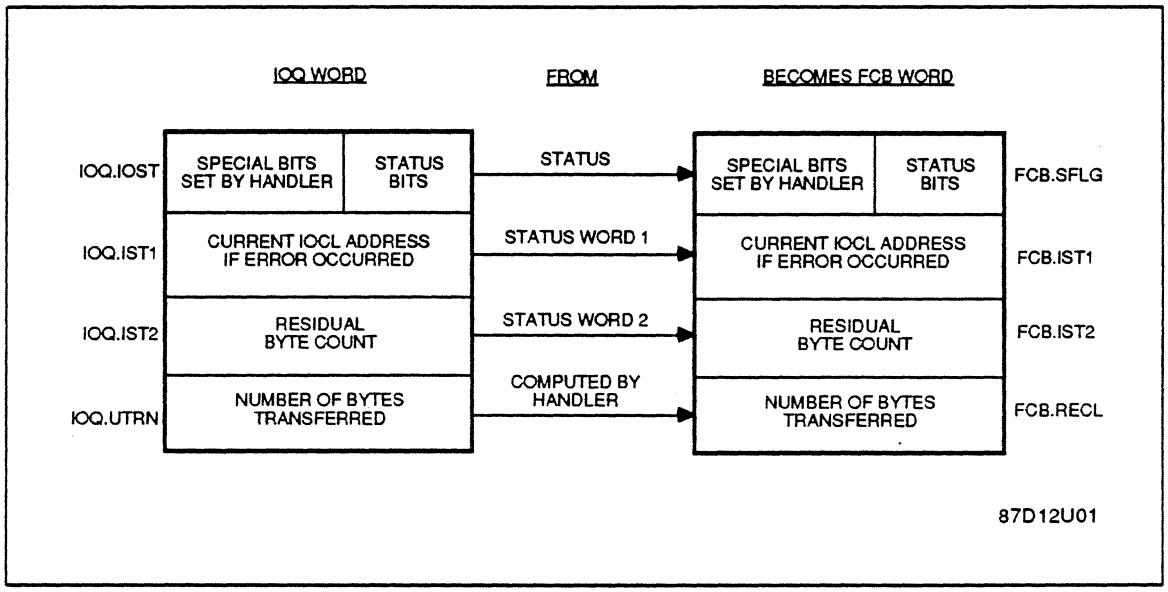
- All General Purpose Device Controller (GPDC) handlers supplied are system re-entrant, therefore, only one copy of a handler is needed per system. Customers writing their own handlers do not need to make them reentrant; however, if they are not, interrupt reentrancy must be specified in the SYSGEN directives.
- All handlers and the interrupt executive use common logic contained within the GPMC.SUB module. GPMC.SUB is loaded only once when a GPMC is configured. Customers writing their own handlers do not need to use the common logic within GPMC.SUB, although in most cases it can simplify design and development.
- SYSGEN creates only one controller definition table (CDT) entry per GPMC (regardless of the number of device addresses). The CDT fields utilized are as follows:

CDT.CLAS	hexadecimal 0D for Model 9103
CDT.FLGS	bit 2 (CDT.GPMC) set to indicate a GPMC controller
CDT.IOST	bit 0 (CDT.NIOQ) set if IOQ is linked to UDT bit 1 (CDT.MUX) set to indicate a multiplexing controller
CDT.SIHA	points to the H.MUX0 HAT
CDT.UT*n*	where *n* is a hexadecimal digit from 0 to F. The UT*n* is a table of sixteen entries, each of which corresponds to a channel. If a device is configured, the corresponding entry contains the address of the UDT, or zero if no UDT corresponds.

- SYSGEN creates one unit definition table (UDT) entry for each GPMC device address configured. The UDT fields utilized are as follows:

UDT.DCAA	points to the DCA which corresponds to the device
UDT.SIHA	points to the appropriate device handler HAT
UDT.STA2	bit 0 set to indicate IOQs are linked from UDT
UDT.TIAD	filled with the TI location address (for debug purposes only)
UDT head cell	IOQ entries are queued here

An execute channel program capability is incorporated to allow users to execute their own IOCD list. Error conditions are detected and noted in the file control block (FCB), however, error correction is the responsibility of the user. The information contained within the FCB is as follows:

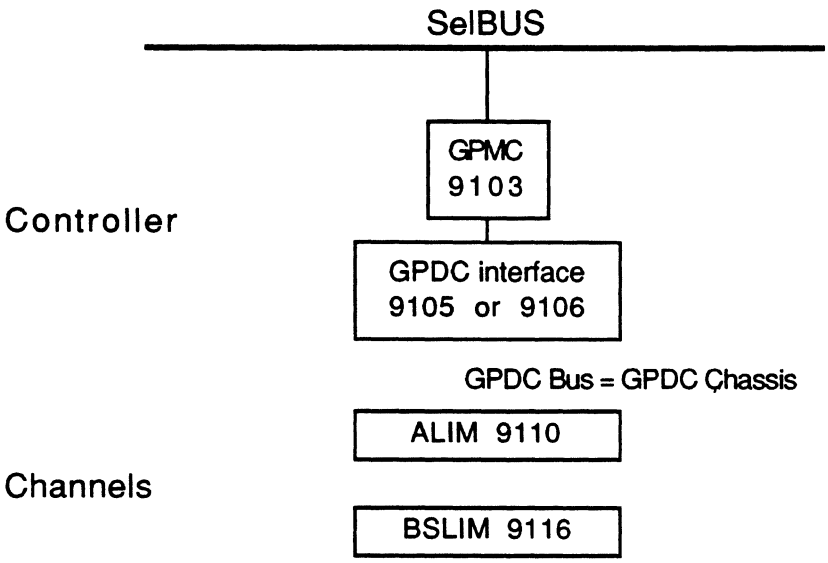


Customer written handlers must use the device context area (DCA) because H.MUX0 and GPMC.SUB reference certain locations within the DCA.

GPMC supports the following devices:

9103	extended GPMC, Class D (l6MB)
9109	synchronous line interface module (SLIM)
9110	asynchronous line interface module (ALIM)
9112	paper pape reader/punch controller
9116	binary synchronous line interface module (BLIM)

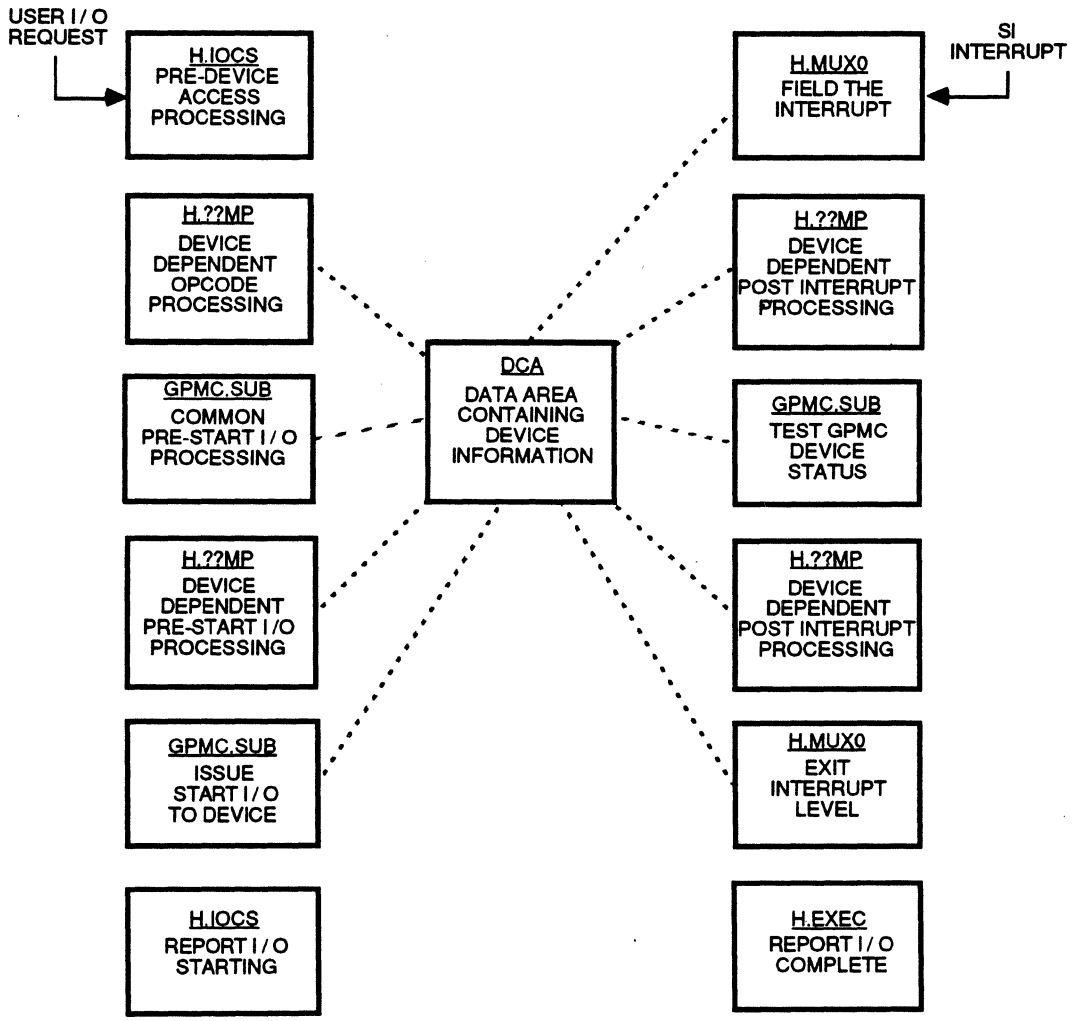
1.2 Hardware Structure



Refer to the GPMC Technical Manual, 325-329104/9103 for details of the GPMC operation. Refer to the appropriate technical manual part number 325-32*xxxx*, where *xxxx* is the model number, for details of a particular GPDC channels operation.

1.3 Software Block Diagram

A simplified overview of the relationship between the GPMC and the operating system is presented below.



87D12T11

2 H.GPMCS Usage

2.1 GPDC Device Handlers (H.??MP)

By specifying a handler name using the SYSGEN DEVICE directive, the handler is made part of the resident operating system and proper linkages are established. The device dependent handler HAT table is the means by which linkages are established between the Input/Output Control System (H.IOCS) and the I/O processing routines.

When an interrupt occurs, the device handlers are not entered directly. The interrupt is handled by the GPMC Interrupt Fielder (H.MUX0) which passes the interrupt to the appropriate handler. For a description of H.MUX0, see section 2.2.

GPDC device handlers have eight entry points.

2.1.1 Entry Point OP. - Opcode Processing

This entry point is a subroutine extension of H.IOCS,29. This entry processes the opcode placed in the file control block (FCB) by the I/O service originally called by the user.

OP. examines the opcode and other pertinent FCB control specifications, and indicates to H.IOCS,29 what action is to be taken. To indicate the action to be taken, OP. takes one of three possible returns to H.IOCS,29 as follows:

BU	ILOPCODE	1. opcode is illegal for this device
BU	SERVCOMP	2. service complete, no device access required
BU	IOLINK	3. link request to I/O queue

If return 3 (IOLINK) is taken, OP. must first call subroutine S.IOCS13 to allocate and initialize an IOQ; then, build an I/O command list (IOCL) with the proper command codes and flags into the the IOQ entry using subroutines S.IOCS12 and S.GPMC5.

Entry Conditions

Calling Sequence

BL	*1W,X2	register X2 contains the address of the device dependent handler HAT table. The one word offset from this address contains the address of OP.

Registers

R1	FCB address
R2	device dependent handler HAT address
R3	UDT address

Exit Conditions

Return Sequence

See descriptions of ILOPCODE, SERVCOMP and IOLINK in the previous text.

Registers

R1 FCB address

2.1.2 Entry Point IQ.- Queue Start Interrupt Service

This entry point is entered from H.IOCS,29 and issues a start I/O (SIO) for the first request in the I/O queue. Upon entry, this entry point sets the interrupt linkage such that subsequent interrupts at this level will cause execution of device handler entry point SI. (Queue Drive Interrupt Service Routine). Entry point IQ. merges with entry point SI. at the preaccess processing phase before calling subroutine S.GPMC1 to issue the start I/O.

Entry Conditions

Calling Sequence

M.IOFF		block interrupts
BL	*2W,X2	register X2 contains the address of the device dependent handler HAT table. The 2-word offset from this address contains the address of IQ.
M.IONN		unblock interrupts

Registers

R0 return address

R3 UDT address of the device to start

Exit Conditions

Return Sequence

Returns to H.IOCS.

2.1.3 Entry Point SI. - Queue Drive Interrupt Service

This entry point is entered from the GPMC Interrrupt Fielder (H.MUX0) and performs postaccess processing associated with the device access which has just completed. Typically, postaccess processing includes device testing, updating IOQ status information, unlinking the queue entry for which processing has been completed, and finding the next highest priority queue entry in the I/O queue for processing.

This entry point also performs preaccess processing associated with the next queued device access request. Typically, preaccess processing includes updating the IOQ entry and making minimal format conversions which might be necessary to service the request.

This entry point is also entered to process a lost interrupt.

When the queue has been emptied, processing is discontinued and the interrupt linkage is set to the spurious entry point (SP.).

Entry Conditions

Calling Sequence

Entered from H.MUX0 as the result of completion of a command issued to a device. Refer to the GPMC Interrupt Fielder (H.MUXO) section in this chapter.

Registers

R0	return address
R3	device context area (DCA) address

Exit Conditions

Return Sequence

After issuing the next command or after determining the I/O queue is empty, entry point SI. returns to H.MUX0 to perform the following:

- report I/O complete via the appropriate executive routine
- restore the state of the machine to mapped, block all interrupts, and issue a deactivate request on the interrupt level being serviced
- exit by S.EXEC5

2.1.4 Entry Point LI.- Lost (Timed-Out) Interrupt Processing

This entry point is entered from S.IOCS5 to perform appropriate measures for a device when an expected interrupt fails to occur. This entry point is also entered from H.IOCS,38 to kill an outstanding I/O request.

This entry point maintains a count of time outs. In addition, a CD terminate is executed to force entry into entry point SI. for processing the time out.

Lost interrupt processing is part of the GPMC.SUB module. Its entry point in the device handler is through BU S.GPMC2.

Entry Conditions

Calling Sequence

M.IOFF		block interrupts
BL	*4W,X2	from S.IOCS5 (or) H.IOCS,38
M.IONN		unblocked interrupts

Registers

R2	HAT address
R3	UDT address

Exit Conditions

Return Sequence

Registers

TRSW	R0

2.1.5 Entry Point PX. - Posttransfer

This entry point is called from S.IOCS1 if a device requires lengthy posttransfer processing.

Entry Conditions

Calling Sequence

BL	*5W,X2	from S.IOCS1

Registers

R0	return address
R1	FCB address
R2	HAT address
R3	UDT address

Exit Conditions

Return Sequence

TRSW	R0	to S.IOCS1

2.1.6 Entry Point SP.- Spurious Interrupt Processing

This entry point is entered from H.MUX0 to prevent a spurious interrupt from causing illegal execution of handler entry points. This entry point maintains a count of all spurious interrupts.

Entry Conditions

Calling Sequence

Entered from H.MUX0. Refer to the GPMC Interrupt Fielder (H.MUX0) section in this chapter.

Registers

None

Exit Conditions

Return Sequence

Returns to H.MUX0 (see entry point SI.).

2.1.7 Entry Point OI. - Error Processing

This entry point is entered to determine if operation intervention is applicable when IOCS detects an error or abnormal condition during device access.

Entry Conditions

Calling Sequence

```
BL      *7W,R2          from S.IOCS1
```

Registers

R1	FCB address
R2	HAT address
R3	IOQ address

Exit Conditions

Return Sequence

```
TRSW    R0
```

- Return 1 - Operator intervention not applicable. Selected when operator intervention is not applicable to the condition that occurred.
- Return 2 - Operator intervention applicable. Selected when an error message must be displayed on the operator's console. The operator is given the opportunity to specify retry attempt or abort of the I/O operation.

2.1.8 Entry Point SG.??? - SYSGEN Initialization

This entry point is called by SYSGEN to initialize certain handler parameters, device context areas (DCAs), and data structure elements during the construction of an MPX-32 image. A maximum of 64 DCAs are created by the repeated assembly of macro GPDC.IT. During execution of this entry point, one DCA is initialized for each UDT entry containing the name of the handler. Any remaining DCAs and the remainder of the code in the handler are overlayed by SYSGEN.

Entry Conditions

Calling Sequence

BL last entry point, it is computed from information in the HAT table

Registers

None

Exit Conditions

Return Sequence

M.XIR

Registers

None

2.2 GPMC Interrupt Fielder (H.MUX0)

The GPMC Interrupt Fielder (H.MUX0) is used to service all I/O interrupts which are generated by completing I/O requests. One copy of H.MUX0 is required for each GPMC. By using a SYSGEN CONTROLLER directive, H.MUX0 is made part of the resident operating system and proper linkages are established.

H.MUX0 is entered each time an interrupt occurs at the level the GPMC is configured. H.MUX0 queries the GPMC for the channel that caused the interrupt, and then vectors to the handler's service or spurious interrupt entries by the contents of IB.EP in the Device Context Area (DCA) which is built by SYSGEN and filled by the handler. Since H.MUX0 is entered only on an interrupt, it contains only one entry point that handles both service or spurious interrupts. IOCS goes directly to the device handler by using the contents of UDT.SIHA.

The handler entry point is found by:

1. Locating the device UDT through the list CDT.UTn.
2. Locating the DCA whose address is saved in UDT.CBLK.
3. Branching through IB.EP in the DCA.

H.MUX0 has two entry points.

2.2.1 Entry Point SI. - Interrupt Fielder

This entry point is entered each time an interrupt occurs at the level for which a GPMC is configured. Upon entry, the following is performed:

1. The new program status doubleword (PSD) is set so the machine state is unmapped, interrupts are unblocked, extended addressing option is set, and the interrupt level being serviced is active. The current map at the time of the interrupt is retained.
2. The global interrupt count (C.GINT) is incremented and all registers are saved.
3. The device that caused the interrupt is determined and the device context area (DCA) address associated withi the interrupt level is loaded into register three.
4. Control is transferred to the appropriate entry point within the device handler to process the interrupt.

When the device handler has determined the I/O queue is empty, control is returned to entry point SI.1.00 in H.MUX0 to perform the following:

Report I/O complete by the appropriate executive routine.

Restore the state of the machine to mapped, block all interrupts, and ensure a deactivate request is being serviced on the interrupt.

Entry Conditions

Calling Sequence

Entered as a result of an interrupt.

Registers

None

Registers

Through S.EXEC5

2.2.2 Entry Point SG. - SYSGEN Initialization

This entry point is called by SYSGEN to initialize certain interrupt fielder parameters and data structure elements during the construction of an MPX-32 image. After execution, the code associated with this entry point is overlayed by SYSGEN.

Entry Conditions

Calling Sequence

BL last entry point

Registers

R7 CHT address

Exit Conditions

Return Sequence

M.XIR

Registers

None

2.3 Common Logic

Module GPMC.SUB is loaded by SYSGEN if a GPMC is configured on the system. The subroutines contained within GPMC.SUB are reentrant; therefore, only one copy is needed per system.

2.3.1 Subroutine S.GPMC0 - Report GPMC Status

Entry Conditions

Calling Sequence

R0	return address
R2	IOQE address
R3	interrupt block address

Exit Conditions

Return Sequence

R0-R4	unchanged
R5,R6	destroyed
R7	device status in right halfword

2.3.2 Subroutine S.GPMC1 - I/O Initiation Logic

Entry Conditions

Calling Sequence

R2	IOQ address
R3	interrupt block address
R6	IOCL address

2.3.3 Subroutine S.GPMC2 - Lost Interrupt Logic

Entry Conditions

Calling Sequence

R0	IOCS return address
R2	handler address (unused)
R3	UDT address

Exit Conditions

Return Sequence

(Direct to IOCS.)

R0-R3	unchanged
R4-R7	destroyed

Refer to section 2.1.4.

2.3.4 Subroutine S.GPMC3 - Initiation and IOQ Entry Acquisition

If the opcode vector table entry bit 0 is set, an IOQ is built for the user by calling S.IOCS13. If bit 1 in the word is set, the IOQ is extended by enough space to hold the absolutized IOCL necessary to perform the requested I/O.

Entry Conditions

Calling Sequence

R0	H.??MP return address (not used)
R1	FCB address
R2	opcode vector table address
R3	UAT address

Exit Conditions

Return Sequence

(Through vector table.)

R0, R1	unchanged
R3	interrupt block address

Other Registers

Indeterminate.

2.3.5 Subroutine S.GPMC4 - Execute Channel Opcode Processor

The execute channel program opcode processor is called by the GPMC device handlers to process either physical or logical execute channel program requests. The basic logic sequence for processing the physical execute channel program request is:

- abort request if requesting task is not privileged
- build and initialize an IOQ
- abort request if the IOCD is not located in the first 128K words of memory
- store user specified time-out value into IOQ
- return to H.IOCS to link the I/O request

The basic logic sequence for processing the logical execute channel program request is:

1. validate legality of IOCD requests and transfer addresses
2. compute number of IOCDs required to satisfy the request
3. build and initialize an IOQ
4. build IOCD list within the IOQ
5. store user specified time-out value in IOQ
6. return to H.IOCS to link the I/O request

Entry Conditions

Calling Sequence

This entry point is called from the opcode processing entry point (OP.) of GPMC device dependent handlers (H.??MP) by branching indirect through the opcode processing table.

Registers

R1 FCB address

Exit Conditions

Return Sequence

BU IOLINK if IOCD list valid

(or)

BU CABORT if IOCD list invalid

Registers

R1 FCB address

R3 IOQ address

2.3.6 Subroutine S.GPMC5 - Build IOCDs for I/O Reads and Writes

This subroutine breaks down the IOCD into one or more transfers and sets the data chaining bit in IOCD if discontiguous, absolutes the IOCD address, and stores IOCDs into the IOCD buffer within the I/O queue and increments the IOCD buffer address as required.

Entry Conditions

Calling Sequence

```
BL      S.GPMC5
```

Registers

R3	I/O queue address
R6	IOCD word 1 with order byte and flags only (bits 16-31) is zero
R7	IOCD word 2 is zero

Exit Conditions

Return Sequence

```
TRSW    R0
```

Abort Cases

I038	dynamic storage space for IOCDs within IOQ exhausted

2.4 GPMC Support Macros

2.4.1 IB.DAT1, IB.DAT2

This macro defines the standard information for a device context area. Special handler information should be inserted between IB.DAT1 and IB.DAT2. The latter macro closes the device context area and computes its size. This macro must start on a doubleword boundary.

For use by unique handler logic, the SET label H.IOCL always points to the physical address of IB.IOCL in the current device context area.

Calling Sequence

```
IB.DAT1
.
.   (handler specific information)
.
IB.DAT2
```

Use the REPT directive to get multiple copies of the device context area.

2.4.2 M.IB

This macro establishes the device context area offset labels. It maps to the contents of IB.DAT1. Special handler information may be equated starting at IB.DFSIZ, which is doubleword bounded.

Calling Sequence

M.IB

2.4.3 GPDC.IT

This macro generates the SYSGEN initialization logic for a GPMC handler.

Calling Sequence

GPDC.IT *lab* [,*timout*], DGPMC

lab	starting label, SG. lab
timout	is a positive number indicating the number of seconds for device time-out. If not provided, a word variable "PA6"should contain the negative time-out count.
DGPMC	must be provided. The macro sets bit 3 of CDT.IOST to indicate a D-class GPMC.

2.5 M.DIB

This macro is called by GPDC.IT to initialize the device context area. This macro contains special case code for the ALIM (Model 9110) handler.

Calling Sequence

M.DIB *type*, DGPMC

type	is the information passed to GPDC.IT as lab
DGPMC	is the information passed to GPDC.IT as DGPMC

2.6 Device Context Area

				Decimal Word	Definition
IB.QEADR				0	current IOQ entry address
IB.CMPQE			*	1	address of IOQ entry just completed
IB.CDTD			*	2	prototype for device CDs & TDs
H.AUT	H.AUTCNT		*	3	GPMC status/residual byte count
IB.DQEAD				4	address of UDT location that points to DQE of allocating task
B.CDTA			*	5	address of CDT
IB.UDTA			*	6	address of UDT
IB. LICNT	IB. SPCNT	IB. OPKODE	IB. * DHBF	7	see Note 1
IB.EP			*	8	handler entry address to use (queue drive or spurious interrupt)
IB.EXIT				9	return address from handler entry
IB.LITIM			*	10	time-out value (in timer units)
H.CNT				11	I/O transfer count in bytes
H.BUF				12	I/O buffer address
H.NCT				13	negative to transfer count remaining
IB.IOCL				14	current IOCB address if transfer error occurred
Temporary storage				15	handler dependent
Optional device dependent information				16	handler dependent

* referenced by GPMC/SUB or H.MUX0

Note:

1. LINCT — lost interrupt count
   SPCNT — spurious interrupt count
   OPKODE — IOCS byte operation code
   DBHF — device handler bit flags as follows:

Bit	Definition
6	postprocessing needed (tells H.MUX0 to report I/O completion)
7	CD terminate issued by lost interrupt entry

High-Speed Data Handler (H.HSDG)

MPX-32 Technical Manual

Volume II

Contents

1 H.HSDG Overview

1.1 Introduction

The high-speed data (HSD) handler (H.HSDG) provides a software interface between user tasks and the HSD. The HSD is a D-class I/O controller that performs handshaking with the CPU. It also fetches and stores the data and status for I/O operations requested by the user task. H.HSDG, in turn, performs the following:

- issues channel programs
- collects status about the I/O operation from the HSD and reports the status to the user task and MPX-32

Note: If the HSD is jumpered for inter-bus link (IBL) mode, refer to the Inter-Bus Link Mode Handler (H.IBLG) behind the H.IBLG tab in this manual.

1.2 SYSGEN Considerations

To include H.HSDG as part of MPX-32, specify H.HSDG in the SYSGEN directive file. For more information about the SYSGEN directive file, refer to the System Generation (SYSGEN) Chapter in Volume III of the MPX-32 Reference Manual.

Following are examples of the CONTROLLER and DEVICE directives that configure MPX-32 for the HSD:

```
CONTROLLER=U040,PRIORITY=09,CLASS=D,HANDLER=(H.HSDG,I)
DEVICE=00,DTC=U0
```

2 H.HSDG - Data Structures

2.1 Introduction

This section describes the following data structures:

- I/O command block (IOCB)
- file control block (FCB)

Other data structures that H.HSDG uses to perform I/O are the controller definition table (CDT), I/O queue (IOQ), and file assignment table (FAT). For more information about these data structures, refer to the MPX-32 Technical Manual, Volume I, Chapter 2.

2.2 IOCB

The IOCB contains the information required for data transfers. This information includes the type of operation, the amount of data (in words) to transfer, the address in CPU memory where data is to be moved to or from, and status about the I/O operation or external device.

One or more IOCBs make up an I/O command list (IOCL). H.HSDG initiates I/O operations by loading the address of the IOCL into the transfer interrupt (TI) location assigned to the device. It then issues the start I/O command device instruction to begin IOCL execution.

Following is the IOCB format:

	0 7	8 15	16 23	24 31
Word 0	HSDCMD See Note 1.	UDDCMD See Note 2.	Transfer count. See Note 3.	
1	Data address or device command. See Note 4.			
2	Not used by hardware. See Note 5.			
3	HSD/device status word. See Note 6.			

Notes:

1. HSDCMD is the HSD command. It defines the operation of the controller as follows:

Bit	Meaning When Set
0	input transfer (read), reset indicates output transfer (write)
1	command transfer, word 1 of IOCB is sent to device
2	device status request, store in word 3 of IOCB
3	continue on error
4	interrupt when IOCB processing completes
5	transfer in channel, branch to specified IOCB
6	command chain, execute next IOCB
7	data chain, continue transfer with address and count specified in the next IOCB

2. UDDCMD is the user device-dependent command byte. It is passed by the HSD to the user device.

3. Transfer count is the amount of data (in 32-bit words) to transfer when the operation is not a command transfer or a transfer in channel.

4. Data address is the physical address of the data buffer to be read into or written from for data transfer operations, or the address of the next IOCB to process for a transfer in channel request. Device command is a 32-bit value sent to the device if HSDCMD bit 1 is set. For more information see the description for FCB word 2, bit 9 and FCB word 10 in the FCB section in this chapter.

5. This word is not used and is available for use by software. The exception is the subtract one and branch nonzero (SOBNZ) request. For more information about the SOBNZ request, refer to the Start I/O Request Format section in Chapter 3.

6. This word stores operation status when an interrupt on end-of-block is requested or stores external device status returned if bit 2 of word 1 is set.

2.3 FCB

The FCB contains information about a requested I/O operation. H.HSDG uses the FCB to report status about the I/O operation to the requesting task.

When a resource is opened, the I/O control system (IOCS) links the FCB associated with the task (located in the task's address space) to a FAT entry. This creates a logical connection between the task and the HSD that can be used for subsequent I/O requests. When a task requests an I/O operation for that logical connection, IOCS links the FCB to an I/O queue (IOQ) entry.

Note: To minimize I/O overhead, use the opened FCB for all I/O to a specific logical file code (LFC). Using alternate FCBs is possible, but not recommended because it changes the links among the FCB, file pointer table (FPT), and FAT that were established at open time. If alternate FCBs are used, an explicit close must be performed for each LFC used. Because MPX-32 relies on information in the FCB during I/O processing, do not modify the FCB from the time the I/O operation is issued until it completes end-action processing.

After the I/O operation completes, the HSD posts status in the FCB associated with the task.

This section details the 16-word FCB used by the HSD. In the following word format, the shaded regions designate areas which are reserved and used by the MPX-32 operating system.

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			

T2001

Word 0

Bit 0 is a system flag that is set by H.HSDG when an execute channel program has an asynchronous notification packet associated with it.

Bits 1-7 contain the operation code, which is a hexadecimal digit 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. For more information, refer to bit 6 of word 2.

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. This bit applies only to FCB format. For more information about FCB format, refer to Chapter 3.
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. This bit applies only to FCB format.
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. This bit applies to FCB and SIO format. For more information about SIO format, refer to chapter 3.
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. This bit applies only to FCB format.
12	specifies disable asynchronous status notification during no-wait I/O. This bit applies only to SIO format.
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 (refer to Chapter 3 for further information). This bit applies only to SIO format.
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). This bit applies only to SIO format.
15-23	reserved

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	EP5 error precluded request queuing. For a list of the EP5 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 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	reserved
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.

Note: Words 8 through 15, described in the following text, are valid only if bit 6 of word 2 is set.

Word 8

Bits 0-7 reserved

Bits 8-31 these bits are used as an expanded data address, a logical IOCL address, or a physical IOCL address as follows:

Expanded data address – This is the starting address of the data area for FCB format I/O operations. This address must be a word address. For more information about FCB format, refer to Chapter 3.

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 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

Bits	Definition
0-31	expanded quantity — number of bytes of data to be transferred (or) for XIO requests (execute channel program), bits 0-7 indicate the number of bytes of sense and bits 8-31 contain the user-specified sense buffer address.

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.

3 I/O Request Processing

3.1 Introduction

H.HSDG accepts requests for I/O operations in two formats: start I/O (SIO) and file control block (FCB). This chapter describes these formats and the I/O functions that can be requested with these formats.

3.2 SIO Format

In SIO format, the user constructs an I/O command list (IOCL) and places the address of the IOCL in FCB word 8. An IOCL is comprised of one or more I/O command blocks (IOCBs) and can be logical or physical.

To initiate IOCL execution, the user puts the address of the user-supplied FCB in register 1 and issues the following call

```
M.CALL H.IOCS,10
```

3.2.1 Physical IOCLs

A physical IOCL is ready to be processed by the HSD. All addresses are physical and all map block crossings are assumed to be resolved. Tasks must be privileged and resident to use a physical IOCL. To indicate that an IOCL is physical, the task must set bit 2 in FCB word 2.

3.2.2 Logical IOCLs

A logical IOCL represents the desired operation in terms of the logical address space of the task. It must be transformed into a physical IOCL before it can be processed by the HSD. H.HSDG converts logical IOCLs into physical IOCLs as follows:

- H.HSDG converts the logical addresses in data transfer IOCBs to physical addresses.
- H.HSDG breaks requests at a physical map block boundary (2KW) when a transfer crosses a boundary. This action resolves discontinuities in the logical-to-physical mapping by breaking the request into physically contiguous segments.
- H.HSDG saves the address of word 2 of the logical IOCB in word 2 of the physical IOCB. This allows control and status to pass back to the logical IOCL once the I/O operation specified by the physical IOCL executes. For more information, refer to the SIO Format Asynchronous Status Presentation and Notification section in this chapter.

After the conversion of the logical to physical IOCL is complete, H.HSDG updates the transfer in channel (TIC) request addresses to reflect the final address of the target IOCB.

3.2.3 Cyclic I/O (SIO Format)

This feature accelerates the cyclic execute channel program (EXCPM) I/O. Cyclic I/O means that the user repetitively issues the same I/O command list (IOCL), logical or physical, with the same buffers and transfer counts. The following two bits, bit 13 (INIT) and bit 14 (GO) in word 2 of the FCB regulate execution of this feature. To use this feature, the task must be resident and the IOQPOOL SYSGEN directive along with the parameter PERMIOQ (see MPX-32 Reference Manual Volume III) must be included in the system image SYSGEN directive file.

When the first IOCL for cyclic usage is issued, the INIT bit must be set. The INIT bit flags the handler which performs all pre-SIO processing for this EXCPM request up to but not including issuing the I/O. Subsequent EXCPM requests with the GO bit set and the INIT bit reset, bypass the pre-SIO processing and issue the I/O previously set up by the INIT request. This causes pre-SIO overhead to be processed only once for a cyclic I/O request.

A cyclic I/O request remains current until such time as a CLOSE or another INIT EXCPM request is issued. If another EXCPM request is issued with the INIT bit set, it becomes the new current cyclic I/O. Non-cyclic I/O requests may be interspersed between cyclic I/O requests without affecting the current cyclic request. The INIT and GO bits are ignored in the FCB format (non-EXCPM) I/O requests.

3.2.4 SIO Format-Specific IOCBs

This section describes the following SIO format IOCBs:

- transfer in channel (TIC)
- subtract one and branch nonzero (SOBNZ)

3.2.4.1 TIC

The TIC command instructs the HSD to continue processing the IOCL at the address specified in IOCB word 1. This is an unconditional branch in the IOCL. A TIC can be used to link IOCLs to form one logically contiguous command set or cause the device to re-execute an IOCL.

3.2.4.2 SOBNZ

The SOBNZ command is a special version of the TIC command. It permits the task to specify that a series of IOCBs may be executed a specific number of times. Following is the IOCB format of the SOBNZ command:

	0 7	8 15	16 23	24 31
Word 0	HSDCMD See Note 1.	UDDCMD See Note 2.	Not used	
1	Address of IOCB to branch to			
2	Initial count. See Note 3.		Current count. See Note 3.	
3	Address of IOCB to branch to			

Notes:

1. HSDCMD is the HSD command. It defines the operation of the controller as follows:

Bit	Meaning When Set
0	input transfer (read), reset for output (write)
1	command transfer, IOCB word 1 sent to device
2	device status request, store in IOCB word 3
3	continue on error
4	interrupt on end of block (when IOCB processing completes). For a TIC or SOBNZ command, this bit should be set in the IOCB that immediately precedes the TIC or SOBNZ command if an interrupt is required.
5	TIC, branch to specified IOCB
6	command chain, execute next IOCB
7	data chain, continue transfer with address and count specified in the next IOCB

   Bits 0 through 4 and 7 should be reset.

2. UDDCMD is the user device dependent command byte and is passed by the HSD to the user device.

3. This field should be set to the number of times the loop is to be executed plus one.

For physical IOCLs, the IOCB preceding the SOBNZ must have the following bits set: interrupt on end-of-block bit (bit 4 of word 0), and the most high order bits in word 2 (bits 0 and 1). If an interrupt on end-of-block is desired for the IOCB preceding the SOBNZ, bit 4 of word 0 and only the most high order bit in word 2 (bit 0) should be set in that IOCB. This is done by H.HSDG for logical IOCLs.

3.2.5 SIO Format Asynchronous Status Presentation and Notification

Asynchronous status presentation allows the user to generate an interrupt and receive status from H.HSDG upon completion of any IOCB within the IOCL. It is specified by setting bit 4 of IOCB word 0.

H.HSDG notifies software of asychronous status presentation via notification packets. These packets are a logical extension to the I/O end-action routines in MPX-32. They are delivered with the same priority as I/O end-action routines so that notification routines and end-action routines will not interrupt each other.

The prerequisites for asynchronous status notification are as follows:

- The I/O operation is in SIO format.
- No-wait I/O is specified. Asynchronous status notification is the default when performing no-wait I/O. To disable the notification at the completion of each IOCB that has the interrupt on end of block bit set (bit 4, word 0), set bit 12 of FCB word 2.
- Asynchronous status notification is enabled in the FCB (bit 12 of FCB word 2 is reset).
- FCB word 13 is supplied and is non-zero.
- Asynchronous status presentation is requested (bit 4 of IOCB word 0 is set).

The user can verify that status was posted by checking the right halfword of word 2 of the IOCB. If the value of word 2 in the IOCB has been incremented by one, status was posted in word 3 of the IOCB.

3.3 FCB Format

The FCB format provides an interface to the user device compatible with standard devices. The user issues requests with the FCB data structure. A device command and/or a data transfer can be initiated from an FCB request. For more information about the FCB, refer to Chapter 2.

For data transfer requests, the buffer address and byte count describe the user buffer for the operation. The user places the address of this buffer in the expanded random access address field (bytes 1, 2, and 3 in FCB word 8).

To initiate the I/O request, the user places the address of the user-supplied FCB in register 1 and issues a M.READ or a M.WRIT service call.

H.HSDG processes the FCB request in two phases. First, H.HSDG computes the required size of the IOCL. It then adds the size of the IOCL to the size of the I/O queue (IOQ) entry. IOCS allocates the IOQ from memory pool and the IOCS routine INIT.IOQ initializes the IOQ. H.HSDG constructs the IOCL according to the table based on the task's operation code. It then queues the request to the CDT and initiates the request. When the request completes, H.HSDG places the final status in the FCB.

3.4 IOCS Entry Points

Following is a list of the IOCS entry points and the applicable HSD I/O function. All of the entry points, with the exception of EXCPM, are used with FCB format.

IOCS Entry Point	Name	Use
1	OPEN	device/handler init (note 1)
2	RWND	rewind device (note 5)
3	READ	input data transfer (note 2)
4	WRITE	output data transfer (note 2)
5	WEOF	write end-of-file (note 5)
10	EXCPM	SIO request format (note 3)
7	ADVR	advance record (note 5)
8	ADVF	advance file (note 5)
9	BKSR	backspace record (note 5)
19	BKSF	backspace file (note 5)
20	UPSP	up space (note 5)
21	ERPT	erase or punch trailer (note 5)
22	EJCT	eject (note 5)
13	CLOSE	device/handler reset (note 4)
24	RSVP	reserve port (note 5)
27	RLSP	release port (note 5)

Notes:

1. Only called once until close. This function is intended for user device-specific processing as required. H.HSDG contains a test device instruction that ensures the presence of the HSD. This can be expanded by the user as needed.
2. May cause device command transfer and/or device status request.
3. Applicable only to SIO format.
4. This is the complementary function to device open.
5. This entry point can be expanded by the user to perform user-defined processing. The entry point dispatches to a common entry point in H.HSDG. H.HSDG uses an internal table, which can be specified by the user at open, to set the UDDCMD field of the IOCB. H.HSDG uses FCB word 10 as the device command word (IOCB word one) and sends it to the device.

3.5 Status

The HSD posts controller status about the transfer in the IOCB. To obtain specific information about the device, perform the following:

- SIO format – construct an IOCB and set bits 2 and 4 in word 0. This causes the HSD to request the device status, and store it in IOCB word 3. (Bit 0 of IOCB word 3 of the returned status is set to indicate device status as opposed to controller status.)

- FCB format – set bit 8 of FCB word 2. When the I/O operation completes, FCB word 12 contains the device status. The device status which is returned relates to a particular device. For more information about controller status returned refer to word 3 of the FCB format section in Chapter 2.

3.6 Device Considerations

Following is a list of device considerations:

- The device will not interrupt at end-of-list unless explicitly instructed. To ensure that the device will interrupt when its idle, software must set bit 4 of IOCB word 0 (if command or data chaining, bits 6 and 7 of word 0, respectively, are not specified).
- If there is a FIFO overflow or non-present memory and the IOCB has the continue on error bit (bit 3, word 0) set, the device posts status and interrupts only if the interrupt on end-of-block bit (bit 4, word 0) is set. The HSD should interrupt when it posts error status. When the HSD is operated in mode two, only nonpresent memory errors are affected when the continue on error bit is set. Therefore, it is recommended that mode two be used and continue on error bit be reset. For more information about HSD mode two, refer to the High-Speed Data Interface Technical Manual.
- If the device stops a transfer due to device end-of-block and is executing an IOCB in a data chain sequence that is not the last IOCB in that sequence, it stops processing the IOCL. The device will not interrupt unless bit 4 of word 0 was set in the IOCB currently being processed or it is operating in mode two. Interrupt on end-of-block for an IOCB of this nature is not the normal case. However, this varies depending on whether it is the last data-chained IOCB.

  If the IOCB where device end-of-block is posted also specifies command chain and no interrupt on end-of-block, the device does not post the residual byte count. No residual byte count is provided when there is no interrupt. To avoid this, operate the device in mode 2. For more information about HSD mode two, refer to the High-Speed Data Interface Technical Manual.
- When external mode is active and the software issues a CD start I/O, the device rejects the command with a privilege violation. No indication is given to software. The operation times out, and a halt I/O instruction is issued that kills the currently running external operation. This can occur when the device is operated in a combination of internal (normally used) and external control modes. The user device can force an error at the end of each external mode transfer to cause an interrupt each time. If H.HSDG had started a transfer, it restarts the operation.

  Any device time out must be set long enough to ensure that the external transfer can complete. This prevents the time out routine from aborting the transfer.
- Interrupts on TIC IOCBs must be avoided. To determine if the HSD has executed a TIC, software should set the interrupt on end-of-block bit (bit 4, word 0) in the previous IOCB. If the TIC causes a channel program loop and the HSD completes the IOCB preceding the TIC before the software deactivates the interrupt, the HSD waits until the level becomes inactive. Software can perform a loop counting operation and modify the branch address. The interrupt service routines must run with the interrupt level active until software modifies the address.

Inter-Bus Link Handler (H.IBLG)
MPX-32 Technical Manual

Volume II

Contents

1 H.IBLG Overview

1.1 Introduction

H.IBLG is a modified version of the H.HSDG handler that is tailored for use with the HSD when it is jumpered for inter-bus link (IBL) mode. IBL mode allows two CPUs to communicate via two HSDs cabled together. For more information about the H.HSDG handler, refer to the High-Speed Data Handler section in this manual behind the H.HSDG tab.

The HSD is a D-class I/O controller that performs handshaking with the CPU. It also fetches and stores the data and status for I/O operation requested by the user task. H.IBLG, in turn, performs the following:

- issues channel programs
- collects status about the I/O operation from the HSD and reports the status to the user task and MPX-32

1.2 H.IBLG Operation

H.IBLG uses link interrupts to control the I/O operations between the two CPUs. If one CPU posts an output operation, H.IBLG issues a link interrupt to the other CPU. H.IBLG always sends the link interrupt from the CPU that is posting a write operation to the CPU posting the read operation.

The CPU that posts the read operation waits for a link interrupt before starting the read operation. If a link interrupt was issued prior to the read operation, the CPU starts the read operation at once; otherwise, the CPU holds the read operation until the link interrupt is issued. Both CPUs must have an I/O operation posted for the I/O operation to occur.

1.3 SYSGEN Considerations

To include H.IBLG as part of MPX-32, specify H.IBLG in the SYSGEN directive file. For more information about the SYSGEN directive file, refer to the System Generation (SYSGEN) chapter in Volume III of the MPX-32 Reference Manual.

Following are examples of the CONTROLLER and DEVICE directives that configure MPX-32 for the HSD when it is in IBL mode:

```
CONTROLLER=U040,PRIORITY=09,CLASS=D,HANDLER=(H.IBLG,I)
DEVICE=00,DTC=U0
```

2 H.IBLG Data Structures

2.1 Introduction

This section describes the following data structures:

- I/O command block (IOCB)
- file control block (FCB)

Other data structures that H.IBLG uses to perform I/O are the controller definition table (CDT), I/O queue (IOQ), and file assignment table (FAT). For more information about these data structures, refer to the MPX-32 Technical Manual, Volume I, Chapter 2.

2.2 IOCB

The IOCB contains the information required for data transfers. This information includes the type of operation, the amount of data (in words) to transfer, the address in CPU memory where data is to be moved to or from, and status about the I/O operation or external device.

One or more IOCBs make up an I/O command list (IOCL). H.IBLG initiates I/O operations by loading the address of the IOCL into the transfer interrupt (TI) location assigned to the device. It then issues the start I/O command device instruction to begin IOCL execution.

Following is the IOCB format:

	0 7	8 15	16 23 24 31
Word 0	HSDCMD See Note 1.	UDDCMD See Note 2.	Transfer count. See Note 3.
1	Data address or device command. See Note 4.		
2	Not used by hardware. See Note 5.		
3	HSD/device status word. See Note 6.		

Notes:

1. HSDCMD is the HSD command. It defines the operation of the controller as follows:

Bit	Meaning When Set
0	input transfer (read), reset indicates output transfer (write)
1	command transfer, word 1 of IOCB is sent to device
2	device status request, store in word 3 of IOCB
3	continue on error
4	interrupt when IOCB processing completes
5	transfer in channel, branch to specified IOCB
6	command chain, execute next IOCB
7	data chain, continue transfer with address and count specified in the next IOCB

2. UDDCMD is the user device-dependent command byte. It is passed by the HSD to the user device.
3. Transfer count is the amount of data (in 32-bit words) to transfer when the operation is not a command transfer or a transfer in channel.
4. Data address is the physical address of the data buffer to be read into or written from for data transfer operations, or the address of the next IOCB to process for a transfer in channel request. Device command is a 32-bit value sent to the device if HSDCMD bit 1 is set. For more information see the description for FCB word 2, bit 9 and FCB word 10 in the FCB section in this chapter.
5. This word is not used and is available for use by software. The exception is the subtract one and branch nonzero (SOBNZ) request. For more information about the SOBNZ request, refer to the Start I/O Request Format section in chapter 3.
6. This word stores operation status when an interrupt on end-of-block is requested or stores external device status returned if bit 2 of word 1 is set.

2.3 FCB

The FCB contains information about a requested I/O operation. H.IBLG uses the FCB to report status about the I/O operation to the requesting task.

When a resource is opened, the I/O control system (IOCS) links the FCB associated with the task (located in the task's address space) to a FAT entry. This creates a logical connection between the task and the HSD that can be used for subsequent I/O requests. When a tasks requests an I/O operation for that logical connection, IOCS links the FCB to an I/O queue (IOQ) entry.

Note: To minimize I/O overhead, used the opened FCB for all I/O to a specific logical file code (LFC). Using alternate FCBs is possible, but not recommended because it changes the links among the FCB, file pointer table (FPT), and FAT that were established at open time. If alternate FCBs are used, an explicit close must be performed for each LFC used. Because MPX-32 relies on information in the FCB during I/O processing, do not modify the FCB from the time the I/O operation is issued until it completes end-action processing.

After the I/O operation completes, the HSD posts status in the FCB associated with the task.

The following section details the 16 words that make up the FCB for the HSD.

	0	7	8	15	16	23	24	31
Word 0	*	Opcode (FCB.OPCD)	Logical file code (FCB.LFC)					
1	Reserved							
2	Control flags (FCB.GCFG)		Special flags (FCB.SCFG)		Random access address (FCB.CBRA)		UDDCMD of IOCB if bit 11 of word 2 is set	
3	Status flags (FCB.SFLG)							
4	Record length in bytes (FCB.RECL)							
5	Reserved		IOQ address (FCB.IOQA)					
6	Special status (FCB.SPST)		Wait I/O error return address (FCB.ERRT)					
7	Index to FPT		FAT address (FCB.FATA)					
8	Reserved		Expanded data address (FCB.ERWA)					
9	Number of bytes to transfer (FCB.EQTY)							
10	Device command for non-EXCPM (FCB.ERAA)							
11	Reserved							
12	Extended I/O status (FCB.IST2)							
13	Reserved		No-wait I/O normal end-action address (FCB.NWOK)					
14	Reserved		No-wait I/O error end-action address (FCB.NWER)					
15	Reserved							

Word 0

Bit 0 is a system flag that is set by H.IBLG when an execute channel program has an asynchronous notification packet associated with it.

Bits 1-7 contain the operation code, which is a hexadecimal digit set by IOCS that specifies the type of function requested of H.IBLG.

Bits 8-31 contain the logical file code associated with the device for the I/O operation.

Word 1

This word is reserved. For more information, refer to bit 6 of word 2.

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.IBLG and IOCS do not perform error processing. IOCS ignores the error return address and takes a normal return to the caller. H.IBLG posts device status in the FCB (unless bit 3 is set). If reset, H.IBLG 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	not applicable, should be zero.
6	specifies expanded FCB (words 8 through 15). This enables larger I/O byte transfers, a 24-bit addressing field, and a 32-bit random access address. When this bit is set, IOCS assumes the FCB is 16 words long. IOCS uses the information in words 8 and 9 instead of the data in word 1. This bit should be set.
7	reserved for internal IOCS use.

Bits 8-15 contain the following special flags:

Bit	Meaning When Set
8	specifies request device status after a transfer. H.IBLG adds an IOCB to the IOCL to retrieve device-specific status after the data transfer completes. This bit applies only to FCB format. For more information about FCB format, refer to chapter 3.
9	specifies send device command prior to data transfer. H.IBLG 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. This bit applies only to FCB format.
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. This bit applies to FCB and SIO format. For more information about SIO format, refer to chapter 3.
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. This bit applies only to FCB format.
12	specifies disable asynchronous status notification during no-wait I/O. This bit applies only to SIO format.
13-23	reserved

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
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

Bit	Meaning When Set
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	EP5 error precluded request queuing. For a list of the EP5 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/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	reserved
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 This field, 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 This field, supplied by the IOCS, points to the file assignment table (FAT) entry associated with this FCB.

Note: Words 8 through 15, described in the following text, are valid only if bit 6 of word 2 is set.

Word 8

Bits 0-7 reserved

Bits 8-31 These bits are used as an expanded data address, a logical IOCL address, or a physical IOCL address as follows:

Expanded data address – This is the starting address of the data area for FCB format I/O operations. This address must be a word address. For more information about FCB format, refer to section 3.3.

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 section 3.2. 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

Bits	Definition
0-31	expanded quantity – number of bytes of data to be transferred (or) for XIO requests (execute channel program), bits 0-7 indicate the number of bytes of sense and bits 8-31 contain the user-specified sense buffer address.

Word 10

For nonexecute channel program format, this word defines a device command.

Word 11

Reserved.

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

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.

3 H.IBLG - I/O Request Processing

3.1 Introduction

H.IBLG accepts requests for I/O operations in two formats: start I/O (SIO) and file control block (FCB). This chapter describes these formats and the I/O functions that can be requested with these formats.

3.2 SIO Format

In SIO format, the user constructs an I/O command list (IOCL) and places the address of the IOCL in FCB word 8. An IOCL is comprised of one or more I/O command blocks (IOCBs) and can be logical or physical.

To initiate IOCL execution, the user puts the address of the user-supplied FCB in register 1 and issues the following call to the I/O control system (IOCS):

```
M.CALL H.IOCS,10
```

3.2.1 Physical IOCLs

A physical IOCL is ready to be processed by the device. All addresses are physical and all map block crossings are assumed to be resolved. Tasks must be privileged and resident to use a physical IOCL. To indicate that an IOCL is physical, the task must set bit 2 in FCB word 2.

3.2.2 Logical IOCLs

A logical IOCL represents the desired operation in terms of the logical address space of the task. It must be transformed into a physical IOCL before it can be processed by the device. H.IBLG converts logical IOCLs into physical IOCLs as follows:

- H.IBLG converts the logical addresses in data transfer IOCBs to physical addresses.
- H.IBLG breaks requests at a physical map block boundary (2KW) when a transfer crosses a boundary. This action resolves discontinuities in the logical-to-physical mapping by breaking the request into physically contiguous segments.
- H.IBLG saves the address of word 2 of the logical IOCB in word 2 of the physical IOCB. This allows control and status to pass back to the logical IOCL once the I/O operation specified by the physical IOCL executes. For more information, refer to the SIO Format Asynchronous Status Presentation and Notification section in this chapter.

After the conversion of the logical to physical IOCL is complete, H.IBLG updates the transfer in channel (TIC) request addresses to reflect the final address of the target IOCB.

3.2.3 SIO Format-Specific IOCBs

This section describes the following SIO format IOCBs:

- transfer in channel (TIC)
- subtract one and branch nonzero (SOBNZ)

3.2.3.1 TIC

The TIC command instructs the HSD to continue processing the IOCL at the address specified in IOCB word 1. This is an unconditional branch in the IOCL. A TIC can be used to link IOCLs to form one logically contiguous command set or cause the device to re-execute an IOCL.

3.2.3.2 SOBNZ

The SOBNZ command is a special version of the TIC command. It permits the task to specify that a series of IOCBs may be executed a specific number of times. Following is the IOCB format of the SOBNZ command:

	0 7	8 15	16 23	24 31
Word 0	HSDCMD See Note 1.	UDDCMD See Note 2.)	Not used	
1	Address of IOCB to branch to.			
2	Initial count. See Note 3.		Current count. See Note 3.	
3	Address of IOCB to branch to.			

Notes:

1. HSDCMD is the HSD command. It defines the operation of the controller as follows:

Bit	Meaning When Set
0	input transfer (read), reset for output (write)
1	command transfer, IOCB word 1 sent to device
2	device status request, store in IOCB word 3
3	continue on error
4	interrupt on end of block (when IOCB processing completes). For a TIC or SOBNZ command, this bit should be set in the IOCB that immediately precedes the TIC or SOBNZ command if an interrupt is required.
5	TIC, branch to specified IOCB
6	command chain, execute next IOCB
7	data chain, continue transfer with address and count specified in the next IOCB

Bits 0 through 4 and 7 should be reset.

2. UDDCMD is the user device dependent command byte and is passed by the HSD to the user device.
3. This field should be set to the number of times the loop is to be executed plus one.

For physical IOCLs, the IOCB preceding the SOBNZ must have the following bits set: interrupt on end-of-block bit (bit 4 of word 0), and the most high order bits in word 2 (bits 0 and 1). If an interrupt on end-of-block is desired for the IOCB preceding the SOBNZ, bit 4 of word 0 and only the most high order bit in word 2 (bit 0) should be set in that IOCB. This is done by H.IBLG for logical IOCLs.

3.2.4 SIO Format Asynchronous Status Presentation and Notification

Asynchronous status presentation allows the user to generate an interrupt and receive status from H.IBLG upon completion of any IOCB within the IOCL. It is specified by setting bit 4 of IOCB word 0.

H.IBLG notifies software of asychronous status presentation via notification packets. These packets are a logical extension to the I/O end-action routines in MPX-32. They are delivered with the same priority as I/O end-action routines so that notification routines and end-action routines will not interrupt each other.

The prerequisites for asynchronous status notification are as follows:

- The I/O operation is in SIO format.
- No-wait I/O is specified. Asynchronous status notification is the default when performing no-wait I/O. To disable the notification at the completion of each IOCB that has the interrupt on end of block bit set (bit 4, word 0), set bit 12 of FCB word 2.
- Asynchronous status notification is enabled in the FCB (bit 12 of FCB word 2 is reset).
- FCB word 13 is supplied and is non-zero.
- Asynchronous status presentation is requested (bit 4 of IOCB word 0 is set).

The user can verify that status was posted by checking the right halfword of word 2 of the IOCB. If the value of word 2 in the IOCB has been incremented by one, status was posted in word 3 of the IOCB.

3.3 FCB Format

The FCB format provides an interface to the user device compatible with standard devices. The user issues requests with the FCB data structure. A device command and/or a data transfer can be initiated from an FCB request. For more information about the FCB, refer to section 2.3.

For data transfer requests, the buffer address and byte count describe the user buffer for the operation. The user places the address of this buffer in the expanded random access address field (bytes 1, 2, and 3 in FCB word 8).

To initiate the I/O request, the user places the address of the user-supplied FCB in register 1 and issues a M.READ or a M.WRIT service call.

H.IBLG processes the FCB request in two phases. First, H.IBLG computes the required size of the IOCL. It then adds the size of the IOCL to the size of the I/O queue (IOQ) entry. IOCS allocates the IOQ from memory pool and the IOCS routine INIT.IOQ initializes the IOQ. H.IBLG constructs the IOCL according to the table based on the task's operation code. It then queues the request to the CDT and initiates the request. When the request completes, H.IBLG places the final status in the FCB.

3.4 IOCS Entry Points

Following is a list of the IOCS entry points and the applicable HSD I/O function. All of the entry points, with the exception of EXCPM, are used with FCB format.

IOCS Entry Point	Name	Use
1	OPEN	device/handler init (note 1)
2	RWND	rewind device (note 5)
3	READ	input data transfer (note 2)
4	WRITE	output data transfer (note 2)
5	WEOF	write end-of-file (note 5)
10	EXCPM	SIO request format (note 3)
7	ADVR	advance record (note 5)
8	ADVF	advance file (note 5)
9	BKSR	backspace record (note 5)
19	BKSF	backspace file (note 5)
20	UPSP	up space (note 5)
21	ERPT	erase or punch trailer (note 5)
22	EJCT	eject (note 5)
13	CLOSE	device/handler reset (note 4)
24	RSVP	reserve port (note 5)
27	RLSP	release port (note 5)

Notes:

1. Only called once until close. This function is intended for user device-specific processing as required. H.IBLG contains a test device instruction that ensures the presence of the HSD. This can be expanded by the user as needed.
2. May cause device command transfer and/or device status request.
3. Applicable only to SIO format.
4. This is the complementary function to device open. It consists of a null routine.
5. This entry point can be expanded by the user to perform user-defined processing. The entry point dispatches to a common entry point in H.IBLG. H.IBLG uses an internal table, which can be specified by the user at open, to set the UDDCMD field of the IOCB. H.IBLG uses FCB word 10 as the device command word (IOCB word one) and sends it to the device.

3.5 Status

The HSD posts controller status about the transfer in the IOCB. To obtain specific information about the device, perform the following:

- SIO format – construct an IOCB and set bits 2 and 4 in word 0. This causes the HSD to request the device status, and store it in IOCB word 3. (Bit 0 of IOCB word 3 of the returned status is set to indicate device status as opposed to controller status.)
- FCB format – set bit 8 of FCB word 2. When the I/O operation completes, FCB word 12 contains the device status. For more information about the status returned, refer to FCB word 3 in section 2.3.

3.6 Device Considerations

Following is a list of device considerations:

- The device will not interrupt at end-of-list unless explicitly instructed. To ensure that the device will interrupt when its idle, software must set bit 4 of IOCB word 0 (if command or data chaining, bits 6 and 7 of word 0, respectively, are not specified).
- If there is a FIFO overflow or non-present memory and the IOCB has the continue on error bit (bit 3, word 0) set, the device posts status and interrupts only if the interrupt on end-of-block bit (bit 4, word 0) is set. The HSD should interrupt when it posts error status. When the HSD is operated in mode two, only nonpresent memory errors are affected when the continue on error bit is set. Therefore, it is recommended that mode two be used and continue on error bit be reset. For more information about HSD mode two, refer to the High-Speed Data Interface Technical Manual.

- If the device stops a transfer due to device end-of-block and is executing an IOCB in a data chain sequence that is not the last IOCB in that sequence, it stops processing the IOCL. The device will not interrupt unless bit 4 of word 0 was set in the IOCB currently being processed or it is operating in mode two. Interrupt on end-of-block for an IOCB of this nature is not the normal case. However, this varies depending on whether it is the last data-chained IOCB.

  If the IOCB where device end-of-block is posted also specifies command chain and no interrupt on end-of-block, the device does not post the residual byte count. No residual byte count is provided when there is no interrupt. To avoid this, operate the device in mode 2. For more information about HSD mode two, refer to the High-Speed Data Interface Technical Manual.

- When external mode is active and the software issues a CD start I/O, the device rejects the command with a privilege violation. No indication is given to software. The operation times out, and a halt I/O instruction is issued that kills the currently running external operation. This can occur when the device is operated in a combination of internal (normally used) and external control modes. The user device can force an error at the end of each external mode transfer to cause an interrupt each time. If H.IBLG had started a transfer, it restarts the operation.

  Any device time out must be set long enough to ensure that the external transfer can complete. This prevents the time out routine from aborting the transfer.

- Interrupts on TIC IOCBs must be avoided. To determine if the HSD has executed a TIC, software should set the interrupt on end-of-block bit (bit 4, word 0) in the previous IOCB. If the TIC causes a channel program loop and the HSD completes the IOCB preceding the TIC before the software deactivates the interrupt, the HSD waits until the level becomes inactive. Software can perform a loop counting operation and modify the branch address. The interrupt service routines must run with the interrupt level active until software modifies the address.

- All I/O operations must be matched with the opposite operation by both CPUs. For example, a read IOCB by one CPU must match a write IOCB from the other CPU. The following example illustrates this restriction with IOCLs that contain more than one IOCB:

CPU A (one IOCL)	CPU B (one IOCL)
read IOCD	write IOCB
write IOCB	read IOCB
write IOCB	read IOCB
read IOCB	write IOCB

- IOCLs can have any number of IOCBs, as long as the IOCLs match with the opposite operations and supply the same transfer count.

- The first IOCB in an IOCL must be a read or a write IOCB. If a write IOCB is the first IOCB in the IOCL, H.IBLG issues a start I/O for the IOCL and generates a link interrupt to the other CPU. If a read IOCB is the first IOCB in the IOCL, H.IBLG waits for a link interrupt from the other CPU before starting the IOCL.

- H.IBLG sends only one link interrupt per IOCL. The following example illustrates this restriction:

CPU C (one IOCL)	CPU D (one IOCB per IOCL)
read IOCD	write IOCB (one IOCL)
write IOCB	read IOCB (one IOCL)
write IOCB	read IOCB (one IOCL)
read IOCB	write IOCB (one IOCL)

  In this example, CPU C holds the first read IOCB until CPU D performs the first write IOCB. When CPU D executes the write IOCB, H.IBLG sends a link interrupt to CPU C. CPU C then waits for a read IOCB from CPU D. CPU D's read IOCB is held, waiting for H.IBLG to send a link interrupt. H.IBLG will not issue this interrupt, because it issues only one link interrupt per IOCL. CPU D will hold the read IOCB indefinitely and CPU C's IOCL never completes successfully.

- To halt an I/O operation, call entry point 4 of H.IBLG. (A CD terminate (halt I/O) will not work.) The user must get the handler address table (HAT) address from the controller definition table (CDT) and perform a BL instruction to the fourth entry in the HAT. Word 3 of the FCB associated with the I/O operation contains status that indicates an error has occurred and a CD terminate was issued. The following example shows the statements that call entry point 4 and halt any outstanding I/O:

```
LW R3,CDT.SIHA,X3   R3 contains the CDT address
BL *4W,X3
```

Memory Disk Handler (H.MDXIO)

MPX-32 Technical Manual

Volume II

Contents

List of Figures

1 H.MDXIO Overview

1.1 General Information

The Memory Disc Handler (H.MDXIO) is a software component of MPX-32 that controls a memory disk. H.MDXIO also translates disk sector numbers to memory locations within a specified partition. Reads and writes can then be performed by copying inline. There is no asynchronous I/O when using a memory disk. No-wait I/O is supported for memory disks, as are all functions previously available to users of disk files. Unlike other disks, memory disk no-wait end-action routines are entered immediately following an I/O request. This difference allows the CPU to retain control instead of directing the I/O request to a controller.

H.MDXIO can support up to eight memory disks of various sizes. The design supports IOCS callable I/O service requests as described in the MPX-32 Reference Manual, Volume I.

An execute channel program capability allows users to execute their IOCD list. Error conditions are detected and noted in the FCB; however, error correction and error retry are the responsibility of the user. Reserve and release IOCDs should never be included within an execute channel program IOCD list.

Warning: Memory disk cannot be used as a physical cache disk.

1.2 Hardware/Software Relationship

The memory disk handler consists of four parts: H.IFXIO, H.MDXIO, XIO.SUB, and the DCAs. H.IFXIO is the interrupt fielder and corresponds one for one with the channel. H.MDXIO is a system handler which controls memory disks. XIO.SUB is the XIO common subroutine package the disk handler calls to perform all common XIO functions. The common XIO subroutine package is described in detail in the XIO Common Subroutines and Device Handlers Chapter. Device context areas (DCAs) are areas of storage and record keeping and correspond one for one with the number of subchannels configured. They are physically located at the end of H.DCXIO.

Up to eight memory disks can be configured on channel 00 on even only subchannels (starting on subchannel 00). However, a memory disk cannot be configured on the same channels and subaddress as the null device. This reduces the number of memory disks to seven for most systems.

1.3 Size of Memory Discs

The maximum size of a memory disk is limited by the physical memory that is available and the memory requirements of system tasks.

The minimum size of the memory disk is limited by the fixed amount of space occupied on a disk after formatting that disk. To calculate the memory size of the files residing on the disk after formatting, add the following items:

- the total number of bytes for the required data files on the disk (i.e. directories and temporary files)
- 768 bytes multiplied by the number of resource descriptors specified by the MAXRES and MAXROOT parameters of J.VFMT.
- 17KB are required for listing directories and resource descriptors on the disk.

2 H.MDXIO Usage

2.1 Dual-Port Memory Disc Support

Dual porting allows two CPUs to share a single memory disk through the multiprocessor shared memory system (MSMS). A dual-ported memory disk is only located in MSMS memory because the memory disk semaphore (bit 0 of the memory disk) must be cached to the other CPU. Memory bus controller (MBC) shared memory does not cache bit settings.

In order to maintain disk and system integrity, mechanisms must exist to prevent both CPUs from accessing the memory disk at the same time. This is accomplished through device reservation and makes the device inaccessible to the nonreserving CPU. Device reservation can be implicit or explicit.

2.1.1 Implicit Device Reservation

Implicit device reservation is processed by the memory disk dual-port semaphore. When dual-port access to the memory disk is required, the processor attempts to set the semaphore. When successful, the memory disk I/O is performed.

2.1.2 Explicit Device Reservation

Explicit device reservation makes a memory disk inaccessible to the opposing CPU for a user-requested period of time. The explicit device reservation is user invoked through the M.RESP service request. The device remains unavailable to the opposing CPU until the user releases the device through the M.RELP service request. If more than one user on the same CPU has a device explicitly reserved at the same time, the drive is not released until the last such user explicitly releases it.

2.2 Dual Subchannel I/O

For memory disk drives, the odd subchannel address is unusable and should never be assigned on the SYSGEN DEVICE directive.

2.3 System Failure in Dual-Port Memory Disc Environment

In a dual-port memory disk environment, one of the systems may fail while the shared memory disk is reserved. If this happens, the shared memory disk can be accessed by the opposing processor through the J.UNLOCK system task. See Chapter 3 of the Technical Manual, Volume I.

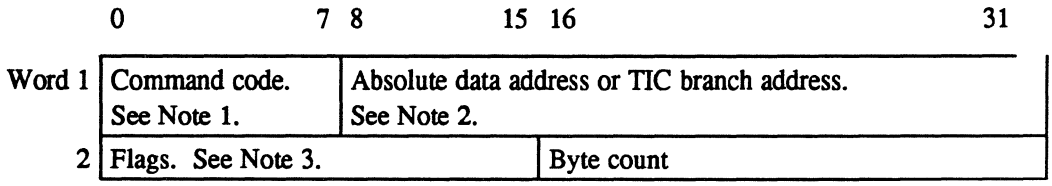
2.4 Maximum Byte Transfer and IOCD Generation

The MPX-32 services available for user read and write requests allow for a maximum transfer of 65K bytes per request. Requests larger than this are truncated to this amount.

S.IOCS40 processes read and write requests by building data-chained IOCDs as necessary to span map blocks. The number of IOCDs generated for any transfer request depends on how large the transfer is and where the buffer begins within the MAP block.

2.5 Extended I/O Commands

Extended I/O (XIO) provides channel commands for completing I/O requests. All channel commands have the following IOCD format:



Notes:

1. The command code field defines the operation that will be performed during command execution.
2. The absolute data address must be a 24-bit absolute address. The TIC branch address must be a 24-bit word-bounded absolute address.
3. Flag bits have the following significance:

Bits	Description
0	data chain (DC)
1	command chain (CC)
2	suppress incorrect length indication (SLI)
3	skip read data (SKIP)
4	post program-controlled interrupt (PPCI)
5-15	zero

XIO channel commands are:

Channel Command	Hexadecimal Command Code	MPX-32 Service Call	Used by MPX-32 Software
sense (SENSE)*	04	none	yes
transfer in channel (TIC)	08	none	yes
write data (WD)	01	M.WRIT	yes
read data (RD)	02	M.READ	yes
read track label (RTL)	52	none	no
seek cylinder (SKC)	07	none	yes
reserve (RES)	23	M.RESP	yes
release (REL)	33	M.RELP	yes
rezero (XEZ)	37	none	yes

*When the sense command is used, IOQ.IST1 is cleared.

While an IOCD containing a command code not listed is ignored, the command chain and data chain bits of that IOCD are interpreted.

2.5.1 Transfer in Channel (TIC)

The TIC command causes input/output command doubleword (IOCD) execution to continue at the address specified in the TIC command. TIC serves as a branch for IOCD execution. A TIC command cannot point to another TIC command and TIC cannot be the first command in an IOCD list.

2.5.2 Write Data (WD)

The WD command is a user write request that transfers data to the disk from the address specified in the IOCD. WD is used for a user write request.

2.5.3 Read Data (RD)

The RD command transfers data from the disk to the address specified in the IOCD. RD is used in the user read request.

2.5.4 Read Track Label (RTL)

When the RTL command is specified, the H.MDXIO handler returns a pseudo-track label with the following information changed in the track label buffer:

Bytes	Contents
12-15	number of sectors on the memory disk
27	number of sectors per track (20)
28	number of heads (1)

2.5.5 Reserve (RES)

The RES command reserves a memory disk for the requesting CPU until a release (REL) is issued. RES is user callable through the M.RESP service routine and is associated with dual-port memory disk operations. Execute channel programs must never include a reserve command and should use the M.RESP service when device reservation is desired.

2.5.6 Release (REL)

The REL command releases a reserved memory disk from the reserving CPU. The release is not issued if more than one task has the memory disk reserved. REL is user callable through the M.RELP service routine and is associated with dual-port memory disk operations. Execute channel programs must never include a release command and should use the M.RELP service routine when memory disk release is desired.

2.5.7 Rezero (XEZ)

The XEZ command is a recalibration request to the memory disk which resets the handler's seek logic to cylinder and track zero.

2.6 Related Data Structures

The following data structures are used by the memory disk handler and are documented in the Systems Table chapter of the Technical Manual, Volume I:

- I/O queue (IOQ)
- unit definition table (UDT)
- controller definition table (CDT)
- file control block (FCB)
- file assignment table (FAT)

Other data structures used by the memory disk handler are the device context area (DCA) and the input/output control doubleword (IOCD).

2.6.1 Device Context Area (DCA)

The device context area (DCA) is a data structure that exists for each subchannel and serves as a storage area for subchannel and subchannel operation information. The DCA contains a common section and a device-dependent section. See Technical Manual, Volume I Chapter 2 for a description of the common section.

Word	Hex byte	0 — 31
36-37	90	Rezero IOCD used for error retry (DCA.REZO)
38-39	98	TIC IOCD used with rezero (DCA.TIC)
40-41	A0	Load mode IOCD prototype (DCA.LMOD)
42-43	A8	Read ECC IOCD (DCA.RECC)
44	B0	ECC data buffer (DCA.ECC)
45	B4	Reserved
46	B8	EOF buffer for nondata transfer command (DCA.EOFB)
47	BC	Address of initialize controller routine (DCA.INCA)
48-49	C0	NOP IOCD for error retry (DCA.NOP)
50	C8	NOP TIC IOCD for error retry (DCA.NOPT)
52	D0	Byte address of memory disk (DCA.OFFS)
53	D4	Size of memory disk in bytes (DCA.MSIZ)
54	D8	Flags for memory disk (DCA.MDFL)
55	DC	Address of shared memory table (SMT) for memory disk (DCA.SMTA)
56	E0	Sector or cylinder for disk (DCA.SCYL)

2.6.2 Input/Output Control Doubleword (IOCD)

The IOCD format and information is located in the extended I/O Command section of this chapter.

2.6.3 Status Returned to User's FCB

The handler places the following information into the IOQ. This information is relayed to the user's file control block (FCB) by IOCS. See Figure 2-1.

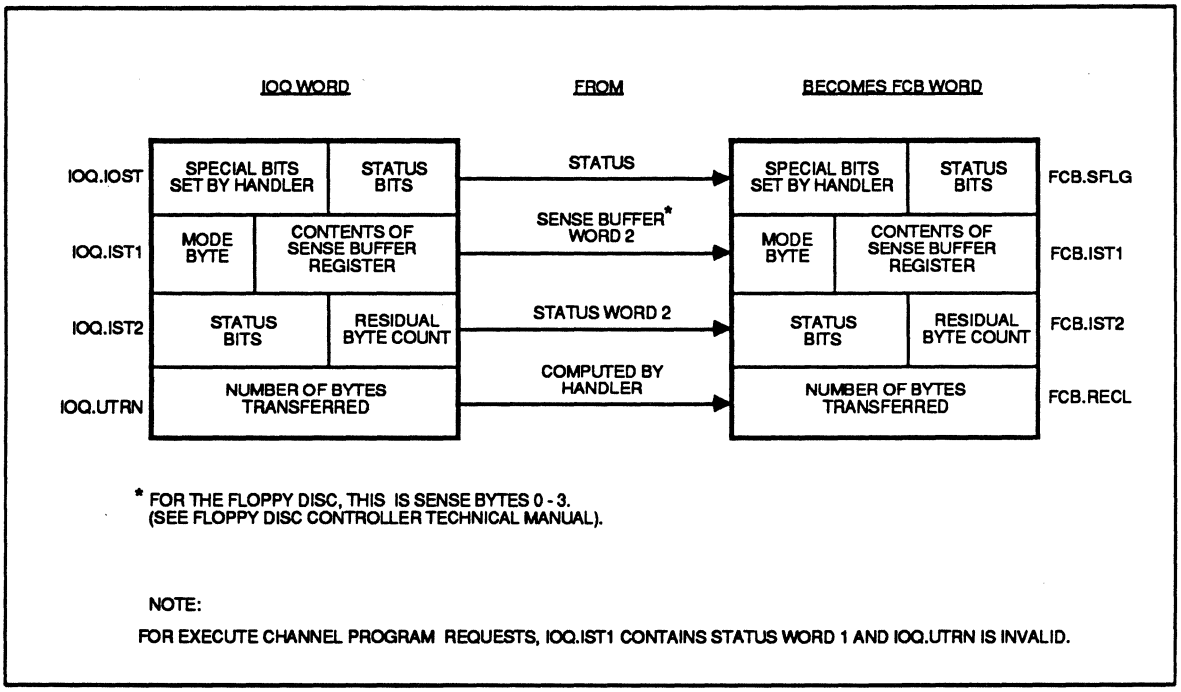


Figure 2-1
Status Returned to User's FCB

2.7 Error Processing for Execute Channel Program Requests

Requests for I/O outside the memory disk boundaries result in an MM01 abort. Information returned consists of status words 1 and 2 passed to FCB words 11 and 12. If bits other than channel end (CE) and device end (DE) are present, bit 1, error condition found, of word 3 in the FCB is set. Bits 16-31 of word 3 are valid. Error correction and error retry are the responsibility of the user.

2.8 SYSGEN Considerations

SYSGEN CONTROLLER and DEVICE directives are used to define XIO disks configured in an MPX-32 system. See the SYSGEN chapter in Volume III of the MPX-32 Reference Manual.

2.8.1 Memory Disc Subaddressing

Each memory disk is assigned a unique device subaddress. Only even subaddresses can be specified in SYSGEN DEVICE directives. If more than one device is specified in a directive, the increment field (INC) must be specified and must be an even number.

2.8.2 Sample XIO Disc Processor SYSGEN Directives

```
1. CONTROLLER=DM00,CLASS=M
2. DEVICE=02,DTC=DM,HANDLER=(H.MDXIO,S),DISC=1024
3. DEVICE=04,DTC=DM,DISC=(1024,D),HANDLER=(H.MDXIO,S),START=256
```

Notes:

1. This `CONTROLLER` directive allows up to 8 memory disks on channel `0`.
2. This `DEVICE` directive assigns a `1024` KB memory disk to subaddress `02`. The handler is `H.MDXIO` and is system reentrant (one copy per system).
3. This `DEVICE` directive assigns a `1024` KB dual-ported memory disk to subaddress `04` starting at decimal map block `256`. The handler is `H.MDXIO`.

3 H.MDXIO Entry Points

3.1 H.MDXIO Entry Points

See the XIO Common Subroutine Package and Handlers (H.XIOS) chapter for a description of XIO device-dependent entry points. Note the following differences:

- Entry points SI., LI.XIO, and PRE.XIO are invalid and cause a branch to ILOPCODE.
- The OP. entry point only supports returns through ILOPCODE and SERVCOMP because this entry point processes the IOCDs.
- The IQ.XIO entry point processes execute channel requests.

Multi-Function Processor Tape Handler (H.MTSCI)

MPX-32 Technical Manual

Volume II

Contents

Contents

List of Tables

1 H.MTSCI Overview

1.1 Introduction

The multi-function processor (MFP) tape handler (H.MTSCI) provides user tasks with an I/O path to small computer system interface (SCSI) tape drives connected to an MFP. H.MTSCI performs the following:

- builds the SCSI command data blocks (CDBs)
- issues channel programs
- collects and reports status about the I/O operation to the user task and MPX-32
- queues I/O operations for a particular tape drive and issues the next queued I/O operation

This section discusses the modules used by H.MTSCI and the SYSGEN considerations.

1.2 Modules Used by H.MTSCI

H.MTSCI calls the following modules: H.IFXIO, XIO.SUB, and H.IOCS. H.IFXIO is the interrupt fielder and corresponds one-to-one with the channel. XIO.SUB is the extended I/O common subroutine package that performs extended I/O functions. H.IOCS performs device-independent I/O request management. This includes preprocessing and postprocessing of I/O requests and I/O queue (IOQ) management. For more information about H.IFXIO, XIO.SUB, and H.IOCS refer to the H.XIOS and H.IOCS sections in this manual.

1.3 SYSGEN Considerations

To include H.MTSCI as part of MPX-32, specify H.MTSCI in the SYSGEN directive file. The SYSGEN CONTROLLER and DEVICE directives are used to specify the MFP and H.MTSCI, respectively. Each SCSI tape drive has a unique device subaddress that is specified with the DEVICE directive. H.MTSCI is system reentrant, which means that only one copy should be configured into MPX-32. For more information about the SYSGEN directive file, refer to the System Generation (SYSGEN) Chapter in Volume III of the MPX-32 Reference Manual.

Following are examples of the CONTROLLER and DEVICE directives that configure MPX-32 for SCSI tape drive support:

```
CONTROLLER=M976,PRIORITY=13,CLASS=F,MUX=MFP,SUBCH=7,CACHE
DEVICE=70,DTC=M9,HANDLER=(H.MTSCI,S)
```

2 H.MTSCI Structures and Entry Points

2.1 Introduction

This section describes the data structures and the entry points that H.MTSCI uses to perform I/O.

2.2 Data Structures

This section describes the following data structures:

- device context area (DCA)
- small computer system interface (SCSI) command data block (CDB)
- handler address table (HAT)
- status doubleword

Other data structures that H.MTSCI uses to perform I/O are the unit definition table (UDT), controller definition table (CDT), I/O queue (IOQ), file control block (FCB), and file assignment table (FAT). For more information about these data structures, refer to the MPX-32 Technical Manual Volume I, Chapter 2.

2.2.1 DCA

The DCA stores subchannel operation information. It contains a common section (words 0 through 35) and a device-dependent section. A DCA must be specified for each subchannel. Table 2-1 lists the H.MTSCI device-dependent information. For more information about the DCA common section, refer to the MPX-32 Technical Manual, Volume I, Chapter 2.

Table 2-1
H.MTSCI Device-Dependent DCA Information

Word	Hex Byte	Meaning
33	84	Time out (DCA.MAX)
34/41	88/A4	Buffer for inquiry data (DCA.INQ)
42	A8	Handler flag word (DCA.CDBF)
43	AC	Error queue (DCA.ERRQ)

2.2.2 SCSI CDB

H.MTSCI uses the SCSI CDB to communicate I/O requests to SCSI device controllers. For more information about the SCSI CDB, refer to ANSI SCSI Committee Working Document X3T9.2/82.2.

2.2.3 HAT

The HAT contains the addresses and the names of the entry points to the H.MTSCI I/O processing routines. It is used by modules such as SYSINIT and the I/O control system (IOCS) to access H.MTSCI.

2.2.4 Status Doubleword

A status doubleword contains the result of the last executed I/O command doubleword (IOCD) when an I/O termination occurs. The MFP generates a status doubleword when an interrupt occurs or when it receives a status stored response from a start I/O (SIO) or halt I/O (HIO) instruction.

When an I/O operation completes, H.MTSCI checks the 16 status bits in the status doubleword for error conditions. H.MTSCI considers the I/O operation complete if the status doubleword has the channel end and device end bits set. If other bits are set, H.MTSCI issues a sense IOCD for additional information about the error.

H.MTSCI stores the sense information in the DCA and maps the status bits, sense bits, and drive status bits to the user's file control block (FCB). For more information about the FCB, refer to the System Tables and Variables Chapter in the MPX-32 Technical Manual, Volume I. A status doubleword has the following format:

	0 7	8 15	16 23	24 31
Word 1	Subchannel. See Note 1.	IOCD address. See Note 2.		
2	Status. See Note 3.		Residual byte count. See Note 4.	

Notes:

1. This field contains the subchannel address of the interrupting device.
2. The IOCD address points 8 bytes past the last executed IOCD.
3. This field contains the following status bits:

Bit	Definition
0	reserved
1	post program-controlled interrupt
2	incorrect length
3	channel program check
4	channel data check
5	reserved
6	interface control check
7	reserved
8	device busy
9	status modifier
10	controller end
11	attention
12	channel end
13	device end
14	unit check
15	unit exception

4. This field contains the number of bytes not transferred for the last IOCD processed.

2.3 Entry Points

Entry points are H.MTSCI routines that perform specific I/O processing. The HAT contains the addresses and names of these routines. This section describes the processing performed by H.MTSCI in the following entry points:

- opcode processor (OP.)
- IOQ driver
- service interrupt processor
- lost interrupt processor (LI.XIO)
- post-transfer processor (PX.)
- SYSGEN initialization (SG.)
- execute channel program opcode processor (XCHANP). For more information about this entry point, refer to the H.XIOS section in this manual.
- pre-SIO processor (PRE.SIO)

2.3.1 Entry Point OP. – Opcode Processor

Entry point OP. processes the opcode placed in the FCB by the I/O service originally called by the user. (OP. is a subroutine extention of H.IOCS,29.) It then indicates the appropriate action for H.IOCS,29 by taking one of the following returns to H.IOCS,29:

BU ILOPCODE	opcode is illegal for this device
BU SERVCOMP	service complete, no device access required
BU IOLINK	link request to IOQ

If OP. takes return IOLINK, it must first call IOCS subroutine S.IOCS13 to allocate and initialize an IOQ. OP. then builds an IOCL into the IOQ using IOCS subroutines S.IOCS12 and IOCS entry point S.IOCS40. If necessary, OP. also obtains space to build the SCSI CDB.

Entry Conditions

Calling Sequence

BL	*1W,X2	X2 contains the HAT address. The 1W offset from this address is the address of OP.

Registers

R1	FCB address
R2	HAT address
R3	UDT address

Exit Conditions

Return Sequence

See descriptions of ILOPCODE, SERVCOMP, and IOLINK.

Registers

R1 FCB address

2.3.2 IOQ Driver

The IOQ driver issues SIO instructions for the IOCLs. H.MTSCI can queue a maximum of 32 outstanding requests per channel. The IOQ driver has two entry points: IQ.XIO and IQ.XIO.1. These entry points are identical except that IQ.XIO activates the interrupt level upon entry and deactivates it before exiting. IQ.XIO and IQ.XIO.1 issue an SIO instruction for the first request in the IOQ.

2.3.3 Entry Point IQ.XIO

H.IOCS,29 calls entry point IQ.XIO when H.IOCS,29 queues an I/O request. It blocks external interrupts and enters IQ.XIO with the following calling syntax.

Entry Conditions

Calling Sequence

BL	*2W,X2	X2 contains the HAT address. The 2W offset from this address is the address of IQ.XIO.

Registers

R0	return address
R3	UDT address of the device to start
R7	IOQ address

Exit Conditions

Return Sequence

DACI		deactivate interrupt level
TRSW	R0	return to calling routine

Registers

R7	IOQ address

2.3.4 Entry Point IQ.XIO.1

The service interrupt processor calls IQ.XIO.1 to drive the IOQ when an I/O request completes. LI.XIO calls IQ.XIO.1 to clear the IOQ when a HIO instruction times out.

Entry Conditions

Calling Sequence

```
BL      IQ.XIO.1      return to call

(or)

BU      IQ.XIO.1      return is set up
```

Registers

R0	return address SI.
R3	UDT address of device to start

Exit Conditions

Return Sequence

```
TRSW  R0  R0 set up prior to return if call is from LI.XIO
```

Registers

R1	CHT address
R2	DCA address

2.3.5 Service Interrupt Processor

The service interrupt processor performs postaccess processing associated with the device access which just completed. It has one entry point, SI. and two routines, SI.UNLNK and SI.EXIT that perform the logic sequence described in the following section.

2.3.6 Entry Point SI.

SI. services interrupts and performs device-dependent logic. It is entered directly from the extended I/O interrupt fielder program (H.IFXIO) with the interrupt level active when one of the following conditions occur:

- an I/O request completes normally
- status checking is inhibited
- status contains a channel end with no device end
- a sense IOCD, unexpected interrupt, or device time out occurs

The following paragraphs describe the SI routines that H.MTSCI enters when one of these conditions occur.

Normal Completion or Status Checking Inhibited Routine – H.MTSCI enters this routine when an I/O request completes with no errors or when status checking is inhibited. It performs any required device-specific processing. An example is the collection of sense information about an I/O operation that just completed.

The extended I/O common routines collect sense information only when an I/O request produces an error or when the I/O request was for an execute channel program and sense information was requested.

This routine:

- checks for a reserve request. If the request exists, the routine increments the reserve count.
- checks for an advance or backspace file request. If the request exists, the routine sets EOF and EOM or BOM and continues processing at SI.EXIT. If no request exists, the routine updates the FAT.
- computes the transfer count
- continues processing at SI.UNLNK

Channel End with No Device End Routine – H.MTSCI enters this routine when an I/O request produces an interrupt whose status contains channel end and no device end. This condition is treated as an unexpected interrupt.

This routine checks for a channel end from a reserve request. If there is no channel end, the routine continues processing as an unexpected interrupt. It then updates the time out in the UDT, shows the I/O as active, and continues processing at SI.EXIT.

Normal Sense Command with IOQ Routine – H.MTSCI enters this routine following an interrupt caused by issuing a sense IOCD for an I/O request that completed with an error indication. This routine examines the status and sense information, initiates error recovery if applicable, and sets the appropriate indicators based on the sense data. It then computes the actual transfer count (if an error condition is indicated), updates the IOQ transfer count, and continues processing at SI.UNLNK.

Unexpected Interrupt Routine – H.MTSCI enters this routine when an interrupt occurs that was not expected. This routine increments the spurious interrupt count for the device and the channel and continues processing at SI.EXIT.

Device Time Out Routine – H.MTSCI enters this routine when an HIO instruction generates an interrupt. (The HIO instruction was issued for a timed-out device.) This routine sets the error condition and the time out flag in the IOQ. It then continues driving the IOQ and processing at SI.EXIT.

Table 2-2 lists the cause of interrupts and the response by SI. in performing conditional service interrupt processing.

Table 2-2
Interrupts and Responses by SI.

Cause of Interrupt	Response by SI.
Channel end with no device end	checks for a reserve request, continues processing at SI.EXIT
Device time out	marks unrecoverable error condition and actual transfer count in the IOQ, H.MTSCI continues to drive the queue
Execute channel program	sets error condition in the IOQ if an error is indicated. If sense information is required, issues a sense IOCD and continues processing at SI.EXIT. If no error is found and no sense information is required, continues processing at SI.UNLNK.
I/O request completes with error	issues sense IOCD and continues processing at SI.EXIT
Normal sense command with IOQ	if execute channel program was requested, continues processing at SI.UNLNK, otherwise, completes the actual transfer count computed and updates the IOQ
Rewind or seek complete	clears device rewinding or seeking bit, continues processing at SI.EXIT
Spurious interrupt when device is not configured	increments spurious interrupt count, and exits the interrupt level. Continues processing at SI.EXIT
Spurious interrupt when device is configured but interrupt is not expected	increments spurious count for the device and the channel, continues processing at SI.EXIT

Entry Conditions

Calling Sequence

BU	*3W,X2	X2 contains the HAT address. The 3W offset from this address is the address of SI.

Registers

R1	CHT address
R3	UDT address

Exit Conditions

Return Sequence

BU	SI.UNLNK	SI unlinks the I/O request, reports the I/O request as complete and IOQ processing continues
(or)		
BU	SI.EXIT	SI. exits the interrupt level without unlinking the I/O request and reporting the I/O as complete. It then initiates error retry or collects sense data.

Registers

R1	IOQ address for SI.UNLNK
R2	DCA address

2.3.7 SI.UNLNK Routine

The SI.UNLNK routine unlinks the IOQ entry from the UDT and reports I/O complete to MPX-32. SI.UNLNK is entered with the interrupt level active. SI.UNLNK is also entered when a device times out due to a kill request or if a device malfunctions.

Entry Conditions

Calling Sequence

BU	SI.UNLNK

Registers

R1	IOQ address
R2	DCA address
R7	IQ. return address

Exit Conditions

Return Sequence

Continues with remainder of SI. logic.

Registers

The registers are unchanged.

2.3.8 SI.EXIT Routine

The SI.EXIT routine continues driving the IOQ and exits the interrupt level. SI.EXIT is entered with the interrupt level active.

Entry Conditions

Calling Sequence

```
BU      SI.EXIT
```

Registers

R2 DCA address

Exit Conditions

Return Sequence

Continues with remainder of SI. logic.

Registers

The registers are unchanged.

2.3.9 Entry Point LI.XIO – Lost Interrupt Processor

S.IOCS5 calls LI.XIO to take corrective measures when an expected interrupt fails to occur. It is also called from H.IOCS,38 when a kill request is issued to a task and the task has an I/O operation in progress. In both cases, the I/O request terminates with an HIO instruction. If the controller responds to the HIO instruction, SI.A performs the required interrupt handling. LI.XIO performs the following:

- activates the interrupt level
- increments the lost interrupt count (if LI.XIO is entered due to a lost interrupt)
- issues the HIO if it has not already been issued
- branches to service interrupt routine if the HIO instruction produces a status stored condition. If status is not stored, LI.XIO blocks external interrupts, deactivates the interrupt level, and returns to the calling routine.

If the HIO instruction has already been issued but fails to generate an interrupt, LI.XIO is entered again and takes the following actions:

- activates the interrupt level
- increments lost interrupt count (if LI.XIO is entered due to a lost interrupt)
- marks the device as offline and malfunctioning
- unlinks the I/O request from the IOQ
- reports the I/O request as complete with errors (if the HIO instruction was issued because of a lost interrupt)
- branches and links to IQ.XIO.1 to clear any pending I/O requests to the failing device
- blocks external interrupts, deactivates the interrupt level, and returns to calling routine

Entry Conditions

Calling Sequence

BL	*4W,X1	X1 contains the HAT address. The 4W offset from this address is the address of LI.XIO.

Registers

R0	return address
R3	UDT address of device to halt

Exit Conditions

Return Sequence

```
TRSW  R0
```

Registers

The registers are unchanged.

2.3.10 Entry Point PX. – Post-Transfer Processor

S.IOCS1 calls entry point PX. to perform processing after completion of the I/O request and before returning to the requesting task. PX. executes at the task priority and with low system overhead.

Entry Conditions

Calling Sequence

BL	*5W,X2	X2 contains the HAT address. The 5W offset from this address is the address of PX.

Registers

R1	FCB address
R2	HAT address
R3	UDT address

Exit Conditions

Return Sequence

TRSW R0

Registers

R1	FCB address

Note: When PX. is called after an open command completes, H.MTSCI issues an inquiry channel command to the tape unit. (The inquiry channel command returns information about the vendor.) If the vendor is Exabyte or Fujitsu, PX. sets a flag that prevents H.MTSCI from setting the immediate bit when building a rewind CDB for these drives.

2.3.11 Entry Point SG. – SYSGEN Initialization

SYSGEN calls entry point SG. to initialize certain handler parameters and data structures during the construction of an MPX-32 image. One DCA is initialized for each UDT entry containing the name of the handler. SYSGEN overlays any remaining DCA's and the remainder of the code in H.MTSCI. SG. updates the DCA with the default time out for I/O. SG. also updates the CHT, CDT, and the UDT to identify the device as a SCSI tape.

Entry Conditions

Calling Sequence

BL	*7W,X2	X2 is the HAT address. The 7W offset from this address is the address of SG.

Registers

Not applicable.

Exit Conditions

Return Sequence

M.XIR	standard handler SYSGEN exit macro

Registers

The register are unchanged.

2.3.12 Entry Point PRE.SIO – Pre-SIO Processor

Entry point PRE.SIO returns control to entry point IQ.XIO of the IOQ driver.

Entry Conditions

Calling Sequence

BL	*5W,X2	X2 contains the HAT address. The 5W offset from this address is the address of PRE.SIO.

Registers

R1	IOQ address
R2	DCA

Exit Conditions

Return Sequence

TRSW R0

Registers

The registers are unchanged.

3 H.MTSCI Issuing I/O Operations

3.1 Overview

There are two ways to issue I/O operations to a SCSI tape drive via H.MTSCI: device-dependent I/O and device-independent I/O. Both methods use the same data structures, channel commands, and CPU instructions.

With device-dependent I/O, the user performs direct channel I/O to the SCSI tape drive via H.MTSCI. The user builds a channel program that H.MTSCI accesses with a physical or logical address. The user also builds an accompanying SCSI command data block (CDB) if there is a transfer command packet (TCP) channel command in the channel program. To initiate the I/O operation, the user issues an execute channel program (EXCPM) request.

With device-independent I/O, the user initiates I/O operations by issuing service (SVC) calls to the I/O control system (IOCS). IOCS verifies the logical address that the user places in the task's file control block (FCB) and links the I/O request to H.MTSCI. H.MTSCI then constructs the necessary channel program and CDB and issues the appropriate CPU instruction to initiate the I/O operation.

This section discusses CPU instructions, channel commands, IOCS service calls, and error processing. For more information about issuing I/O operations, refer to the Resource Assignment/Allocation and I/O chapter in Volume I of the MPX-32 Reference Manual.

3.2 CPU Instructions

H.MTSCI uses CPU instructions to perform I/O to an MFP. The CPU instructions have the following format:

0 5	6 8	9 12	13 15	16 31
Opcode. See Note 1.	Register. See Note 2.	Instruction.	Augment code. See Note 3.	Constant. See Note 4.

Notes:

1. Bits 0-5 specify the hexadecimal operation code 0-FC.
2. Bits 6-8 specify the general purpose register. When these bits are nonzero, the register contents are added to the constant specified in bits 16-31 to form the logical channel and subaddress.
3. Bits 13-15 specify the augment code up to hexadecimal 7.
4. Bits 16-31 specify a constant that is added to the contents of the register specified by bits 6-8. This forms the logical channel and subaddress. If bits 6-8 are zero, this field specifies the logical channel and subaddress.

The MFP generates a condition code to report the result of a CPU instruction. Following are the possible condition codes:

Condition Code	Meaning
0001	channel busy – request denied because channel internal process queue is full
0010	channel inoperative or undefined – request denied because channel is not present, functional, or defined
0011	subchannel busy – request denied because of outstanding halt I/O (HIO), stop I/O (STPIO), or clear queue instruction
1000	Request accepted and queued – the CPU instruction was accepted by the channel for execution

Table 3-1 lists and the following text describes the CPU instructions.

Table 3-1
CPU Instructions

Instruction	Hex Opcode
Activate channel interrupt (ACI)	E
Clear queue	7
Deactivate channel interrupt (DACI)	F
Disable channel interrupt (DCI)	D
Enable channel interrupt (ECI)	C
Halt I/O (HIO)	6
Reset channel (RSCHNL)	5
Reset controller (RSCTL)	8
Start I/O (SIO)	2
Stop I/O (STPIO)	4
Test I/O (TIO)	3

3.2.1 Activate Channel Interrupt (ACI)

The ACI instruction causes the MFP to assert its interrupt priority level and actively contend for recognition from the CPU. While contending for recognition, the channel cannot request another interrupt.

3.2.2 Clear Queue

The clear queue instruction terminates all outstanding I/O operations for the specified subaddress. The current operation terminates with channel end and device end status. Queued operations terminate with channel end, device end, and unit exception status to indicate that they were not initiated.

3.2.3 Deactivate Channel Interrupt (DACI)

The DACI instruction causes the MFP to suspend contention for interrupt priority. If an interrupt request is queued and enabled, the channel may now issue an interrupt.

3.2.4 Disable Channel Interrupt (DCI)

The DCI instruction prevents the MFP from requesting interrupts from the CPU.

3.2.5 Enable Channel Interrupt (ECI)

The ECI instruction allows the MFP to request interrupts from the CPU.

3.2.6 Halt I/O (HIO)

The HIO instruction causes the MFP to terminate an I/O operation and post a status doubleword that contains device end (DE) and channel end (CE) status.

3.2.7 Reset Channel (RSCHNL)

The RSCHNL instruction causes the MFP to stop operation, reset all activity, and become idle. RSCHNL resets all subchannels and any requesting or active interrupt levels.

3.2.8 Reset Controller (RSCTL)

The RSCTL instruction causes the addressed subchannel to terminate an I/O operation. If the subchannel is hung, RSCTL resets the device so that I/O operation may resume. If the addressed subchannel is not currently being serviced, RSCTL clears the I/O request. The MFP presents any pending status for the addressed subchannel to the CPU. RSCTL does not generate final status.

3.2.9 Start I/O (SIO)

The SIO instruction initiates an I/O operation for the specified channel or subchannel. If this channel or subchannel is present and available, it accepts the SIO instruction. The channel or subchannel then performs the I/O operation specified by the SIO instruction. If the I/O operation cannot be started, the MFP returns the appropriate condition codes and status.

3.2.10 Stop I/O (STPIO)

The STPIO instruction terminates the I/O at the addressed subchannel when the current channel command has been executed. STPIO resets the data chain and command chain flags, and the MFP returns the appropriate condition codes and status.

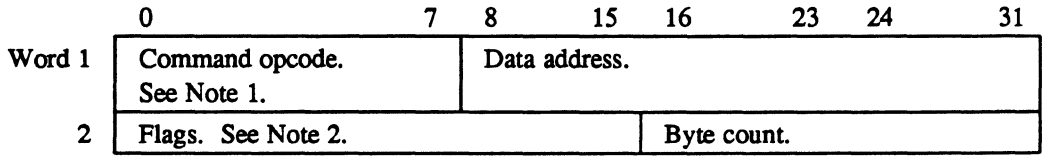
3.2.11 Test I/O (TIO)

The TIO instruction verifies the current state of the channel or subchannel and clears pending interrupt conditions. The MFP returns condition codes that reflect the status of the channel and addressed subchannel.

3.3 Channel Commands

Channel commands contain the information required for data transfers. This information includes the type of operation, the address in CPU memory where data is to be moved to or from, flags that indicate to the MFP what to do after completing execution of the channel command, and the amount of data (in bytes) to transfer.

Channel commands are specified in I/O command doubleword (IOCD) format as follows:



Notes:

1. The command opcode field specifies the command to be executed.
2. The flags are defined as follows:

Bit	Meaning When Set
0	data chain
1	command chain
2	suppress incorrect length
3	skip read data
4	post program-controlled interrupt
5-15	not used – must be zero

Table 3-2 lists and the following text describes the channel commands that can be issued for a SCSI tape.

Table 3-2
H.MTSCI Channel Commands

Command	Opcode
Backspace one filemark	x'73'
Backspace one record	x'53'
Channel control	x'80'
Initialize channel	x'00'
Initialize subaddress	x'F0'
Inquiry	x'B3'
No operation	x'03'
Read	x'02'
Release	x'C3'
Reserve	x'A3'
Rewind	x'23'
Sense	x'04'
Space forward one filemark	x'63'
Space forward one record	x'43'
Space multiple filemarks	x'E3'
Transfer command packet	x'D3'
Transfer in channel	x'08'
Write	x'01'
Write multiple filemarks	x'13'
Write one filemark	x'93'

3.3.1 Backspace One Filemark

The backspace one filemark command rewinds the tape one filemark. The data address and byte count fields are ignored.

3.3.2 Backspace One Record

The backspace one record command rewinds the tape one record. The data address and byte count fields are ignored.

3.3.3 Channel Control

The channel control command returns three words of channel information. Word 1 is the board model number, word 2 is the firmware model number, and word 3 is the firmware revision level.

3.3.4 Initialize Channel

The initialize channel command transfers tape drive information to the MFP and sets the status buffer address for the MFP. The status buffer address must be doubleword bounded and specified in the data address field. The byte count field is ignored. The initialize channel command, performed by H.MTSCI, must be the first channel command to any channel that has an MFP configured.

3.3.5 Initialize Subaddress

The initialize subaddress command initializes several subaddress-specific channel operations with the information from the byte in memory specified by the data address field. The byte count must be one. Following is the format of the byte:

Bit	Meaning When Set
0	clear SIO queue on error. As each queued SIO is cleared, termination status is returned indicating this action. If reset, when a queued SIO command terminates in error, continue processing SIO queue
1	reserved
2	use drive-specified block size. If reset, convert logical block size of 256 bytes to simulate block size of 768
3-7	reserved

3.3.6 Inquiry

The inquiry command returns information about the device such as the peripheral device type, vender identification, and device-type qualifier. Refer to ANSI SCSI Committee Work Document X3T9.2/82.8 for more information about the information returned for this command.

3.3.7 No Operation

The no operation command is a non-data transfer command that executes without selecting an associated tape drive. The byte count and data address must be zero.

3.3.8 Read

The read command transfers data from the tape to the address specified in the data address field. The byte count must be greater than zero.

3.3.9 Release

The release command releases a reserved device by the reserving CPU. The release is not issued if more than one task has the device reserved. The data address and byte count fields are ignored.

3.3.10 Reserve

The reserve command reserves a device to the requesting CPU until a release command is issued. The data address and byte count fields are ignored.

3.3.11 Rewind

The rewind command rewinds the tape. The data address and byte count fields are ignored.

3.3.12 Sense

The sense command causes H.MTSCI to issue an extended sense command to the specified device for the number of bytes requested in the byte count field. The sense data returned is shown below. For more information about the returned information, refer to ANSI SCSI Committee Working Document X3T9.2/82.2.

Bit	Meaning When Set
0	no sense
1	recovered error
2	not ready
3	medium error
4	hardware error
5	illegal request
6	unit attention
7	data protect
8	blank check
9	vendor unique
A	copy aborted
B	aborted command
C	equal comparison satisfied
D	volume overflow
E	miscompare
F	reserved

3.3.13 Space Forward One Filemark

The space forward one filemark command advances the tape one filemark. The data address and byte count fields are ignored.

3.3.14 Space Forward One Record

The space forward one record command advances the tape one record. The data address and byte count fields are ignored.

3.3.15 Space Multiple Filemarks

The space multiple filemarks command advances the tape the number of filemarks specified by the byte count. The data address field is ignored.

3.3.16 Transfer Command Packet

The transfer command packet command sends a SCSI CDB to a device. The data address specifies the address of the CDB built by the user. The byte count specifies the length of the CDB. If a data transfer is required with the CDB, the command chain flag must be set. This command must be command chained to a write or a read command that specifies the data address and byte count.

3.3.17 Transfer in Channel

The transfer in channel command causes IOCD execution to continue at the address specified in the data address field. A transfer in channel command cannot point to another transfer in channel command, and it cannot be the first command in a channel program. The handler uses a transfer in channel command to link channel commands in the device context area (DCA) to channel commands in the I/O queue (IOQ).

3.3.18 Write

The write command transfers data to the tape from the address specified in the data address field. The byte count must be greater than zero.

3.3.19 Write Multiple Filemarks

The write multiple filemarks command writes to the tape the number of filemarks specified in the byte count field. The data address field is ignored.

3.3.20 Write One Filemark

The write one filemark command writes one filemark on the tape. The data address and byte count fields are ignored.

3.4 IOCS Service (SVC) Calls

Table 3-3 lists the SVC calls, their applicable functions, and the bit setting for the FCB field FCB.SCFG:

Table 3-3
IOCS SVC Calls

Function	SVC	FCB.SCFG	Notes
Advance filemark	1,x'34'	0000	
Advance filemarks	1,x'34'	0001	1
Advance record	1,x'33'	0000	
Advance records	1,x'33'	0001	1
Backspace filemark	1,x'36'	0000	
Backspace filemarks	1,x'35'	0010	1
Backspace record	1,x'35'	0000	
Backspace records	1,x'36'	0101	1
Backspace sequential filemarks	1,x'35'	0011	1
Backspace to end of data	1,x'35'	0100	
Erase	1,x'3A'	0000	
Erase long	1,x'3A'	0001	
Execute channel program	1,x'25'	n/a	11
Initialize subaddress	1,x'3E'	0010	2
Inquiry	1,x'35'	0001	3, 4
Mode select	1,x'26'	0001	5, 6
Mode sense	1,x'26'	0000	3, 6
Open	n/a	n/a	10
Read block limits	1,x'27'	0001	3
Read blocks	1,x'31'	0001	3, 7
Read data	1,x'31'	0000	3, 8
Read MFP	1,x'27'	0010	3, 9
Release unit	1,x'10'	0001	12
Reserve unit	1,x'10'	0000	12
Rewind	1,x'37'	0000	
Test unit ready	1,x'27'	0000	3
Unload unit	1,x'26'	0010	
Write data	1,x'32'	0000	3, 6
Write EOF filemark	1,x'38'	0000	
Write EOF filemarks	1,x'38'	0001	1

Notes:

1. FCB.EQTY specifies the number of filemarks or records (whichever is applicable).
2. FCB.ERWA specifies the address of the buffer that contains the initialization parameters. FCB.EQTY specifies the size of the initialization parameters.
3. FCB.ERWA specifies the buffer address.
4. FCB.EQTY specifies the buffer length.

5. FCB.ERWA specifies the address of the buffer that contains the SCSI command packet to be sent to the unit.
6. FCB.EQTY specifies the byte count.
7. FCB.EQTY specifies the block count.
8. FCB.ERWA specifies the address of the read buffer.
9. FCB.EQTY specifies a byte count of 1 to 12 bytes.
10. The SVC call is not applicable because it is performed by H.MTSCI.
11. FCB.IOQA must contain the logical address of the channel program to execute.
12. To issue a third-party reserve or release request, use the execute channel program SVC call.

3.5 Error Processing

If an I/O operation terminates abnormally, H.MTSCI posts information about the I/O operation in the FCB. This information is device status in FCB word 3 (bits 16 through 31), and sense data in FCB words 11 and 12. For more information about the returned status, refer to the status doubleword data structure in Chapter 2 of this document.

The following is a list of the sense data returned for H.MTSCI:

FCB Word	Byte	Sense Data	Notes
11	0	error class/code	1
	1	segment number	
	2	tape indicator bits	2
	3	residual byte count (msb)	
12	0-3	channel status	3

Notes:

1. This specifies extended sense data format. For more information, refer to ANSI Committee Working Document X3T9.2/82.2.
2. The following are the tape indicator bits:

Bit	Definition
0	filemark encountered
1	EOM/BOM encountered
2	illegal length
3	reserved
4-7	sense key information. refer to the sense channel command description in this chapter.

3. Contains channel status in bits 0 through 16 and the residual byte count in bits 17 through 31.

XIO Common Subroutine Package & Device Handlers (H.XIOS)

MPX-32 Technical Manual

Volume II

Contents

Contents

List of Figures

List of Tables

1 H.XIOS Overview

1.1 XIO Common Subroutine and Device Handlers

1.2 General Information

MPX-32 contains an XIO subroutine package that performs the I/O functions that are common to all XIO devices. Individual device handlers contain only device dependent logic and are structured to use the common XIO subroutines.

The concept of an XIO device handler under MPX-32 consists of: the common XIO subroutine package (XIO.SUB), the device dependent logic (H.??XIO), the interrupt fielder (H.IFXIO), and the device context area (DCA). A simplified overview of the relationship between these components and the operating system is presented in Figure 1-1.

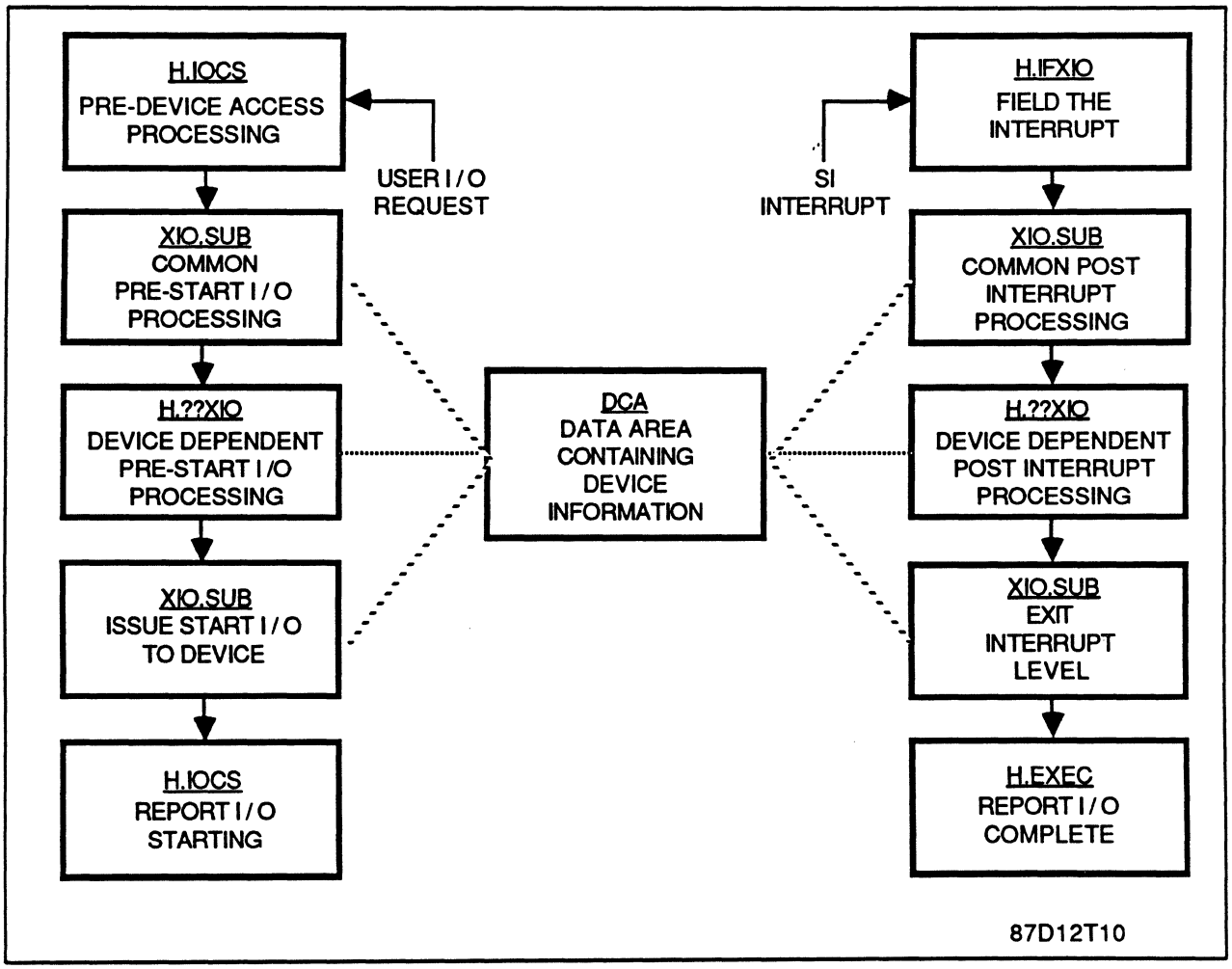


Figure 1-1
Simplified Software Block Diagram

2 XIO.SUB Subroutines

2.1 General Information

The common XIO subroutine package (XIO.SUB) is made part of the resident operating system by naming an F-class device handler within a SYSGEN directive file. The routines are re-entrant requiring only one copy per system. The common XIO subroutine package consists of four routines:

- the I/O queue driver (IQ.XIO)
- the service interrupt processor (SI.)
- the lost interrupt processor (LI.XIO)
- the execute channel program opcode processor (EXCPM).

A fifth routine, COM.INIT, is used by SYSGEN during MPX-32 image construction and is subsequently overlayed.

XIO device handler linkage to IQ.XIO and LI.XIO is accomplished by substituting the subroutine name within the handler HAT table and defining the routine as an external. System calls to these two routines are accomplished by branching indirect through the HAT table address. The service interrupt processor (SI.) is called directly from the channel interrupt fielder (H.IFXIO), while the execute channel program opcode processor (EXCPM) is called from the opcode processing entry point (OP.) of XIO device dependent handlers (H.??XIO).

2.2 I/O Queue Driver (IQ.XIO)

The I/O queue driver issues start I/O (SIO) commands for the calling routine. It has three entry points: IQ.XIO, IQXIO.1 and IQ.XIO.2. IQ.XIO and IQ.XIO.1 are identical except IQ.XIO activates the interrupt level upon entry and deactivates it before exiting. IQ.XIO and IQ.XIO.1 issue an SIO for the first request in the I/O queue. IQ.XIO.2 issues an SIO for a specific entry within the I/O queue.

The following is the basic logic sequence within the I/O queue driver with the three entry points noted:

- IQ.XIO enters here to activate the interrupt level.
- IQ.XIO.1 enters here to determine which I/O request to process.
- IQ.XIO.2 enters here to:
  - check if device is online and functioning
  - branch and link to device specific handler logic to perform any modifications to the I/O request
  - get various I/O request parameters from the IOQ and set up for an SIO. These include the time-out value, the IOCD list address, and the channel and subchannel to start.
  - issue an SIO
  - examine condition codes presented by the SIO and take appropriate action
  - deactivate the interrupt level if entered at IQ.XIO
  - return to calling routine

2.3 Entry Point IQ.XIO

Entry Point IQ.XIO is called by H.IOCS,29 each time H.IOCS,29 queues an I/O request and, depending on the queueing scheme, the channel or device is not busy. It blocks external interrupts and enters IQ.XIO by the calling sequence.

Entry Conditions

Calling Sequence

BL	*2W,X2	register X2 contains the device dependent handler device dependent handler HAT address. The address in the HAT table points to the common subroutine IQ.XIO entry point.

Registers

R0	return address
R3	UDT address of the device to start
R7	IOQ address

Exit Conditions

Return Sequence

DACI		deactivate interrupt level
TRSW	R0	returns to calling routine

Registers

R7	IOQ address

2.4 Entry Point IQ.XIO.1

Entry Point IQ.XIO.1 is called by the SI. service interrupt routine to drive the I/O queue following completion of an I/O request. LI.XIO calls IQ.XIO.1 to flush the I/O queue following a halt I/O command that times out. IQ.XIO.1 is entered with the interrupt level active by the calling sequence.

Entry Conditions

Calling Sequence

BL	IQ.XIO.1	if return to call is desired
	(or)	
BU	IQ.XIO.1	if return is set-up

Registers

R0	return address (SI. only)
R3	UDT address of device to start

Exit Conditions

Return Sequence

TRSW	R0	R0 is properly set-up prior to return if call is from LI.XIO

Registers

R1	CHT address
R2	DCA address

2.5 Entry Point IQ.XIO.2

Entry Point IQ.XIO.2 can be called by device dependent handlers to perform a start I/O on behalf of a specific I/O request to collect sense information, perform error retry, continue with the present I/O request, etc. It is entered with the interrupt level active by the calling sequence.

Entry Conditions

Calling Sequence

```
BL      IQ.XIO.2
```

Registers

R0	return address
R1	IOQ address
R2	DCA address
R3	UDT address

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1	CHT address
R2	DCA address

2.6 Service Interrupt Processor (SI.)

The service interrupt processor (SI.) performs postaccess processing associated with the device access which just completed. It has three entry points called SI., SI.UNLNK and SI.EXIT. SI.UNLNK and SI.EXIT are also callable from the device dependent logic.

The following is the basic logic sequence within the service interrupt processor with the three entry points noted.

- SI. enters here to:
  - determine which device caused the interrupt (status presentation)
  - determine cause of interrupt and branch to appropriate action. See the Conditional SI. Processing section for cause of interrupt information.
  - branch and link to device dependent handler logic to perform any normal device specific postaccess processing
  - update actual transfer quantity if required
- SI.UNLNK enters here to:
  - unlink IOQ entry from I/O queue
  - delete the IOQ if a kill request was issued
  - report I/O complete
- SI.EXIT enters here to:
  - branch and link to IQ.XIO.1 to continue driving the I/O queue (this may or may not return)
  - test for any more status pending and if so, branch back to SI.; otherwise,
  - deactivate the interrupt level
  - exit from the interrupt via S.EXEC5

2.7 Entry Point SI.

Entry point SI. is entered directly from the XIO interrupt fielder program (H.IFXIO). This entry point can also be entered from the I/O queue driver (IQ.XIO) and the lost interrupt processor (LI.XIO) as a result of a status stored response to the start I/O and halt I/O instructions. SI. is entered with the interrupt level active by the calling sequence.

Entry Conditions

Calling Sequence

```
BU      SI.
```

Registers

R3	CHT address
R6	specifies 0 if entered from the XIO interrupt fielder (H.IFXIO) (or) specifies the UDT address of the device on whose behalf the SIO or HIO was issued when the status stored response was generated.

Exit Conditions

Return Sequence

If entered from the XIO interrupt fielder (H.IFXIO):

Return Sequence

BU S.EXEC5

Registers

R2 register save area address

R6,R7 old PSD

If entered from the I/O queue driver (IQ.XIO) or the lost interrupt processor (LI.XIO):

Return Sequence

BU IQ.XIO.1

Registers

R0 return address to either IOCS or the lost interrupt fielder (LI.XIO)

R3 UDT address of completing device

2.8 Entry Point SI.UNLNK

Entry point SI.UNLINK can be entered from device dependent handlers (H.??XIO) if the I/O request should not be issued. For example, a release request is issued for a dual-ported disc while another user still has the device reserved. SI.UNLNK is entered with the interrupt level active by the calling sequence.

Entry Conditions

Calling Sequence

BU SI.UNLNK

Registers

R1 IOQ address

R2 DCA address

R7 IQ. return address as passed to device dependent handler logic

Exit Conditions

Return Sequence

Continues with remainder of SI. logic. See the Service Interrupt Processor section.

2.9 Entry Point SI.EXIT

Entry point SI.EXIT can be entered from device dependent handlers (H.??XIO) to continue driving the I/O queue and to exit the interrupt level. SI.EXIT is entered with the interrupt level active by the calling sequence.

Entry Conditions

Calling Sequence

```
BU      SI.EXIT
```

Registers

R2 DCA address

Exit Conditions

Return Sequence

Continues with remainder of SI. logic. See the Service Interrupt Processors section.

2.9.1 Conditional SI. Processing

Table 2-1 describes the cause of interrupts and response by the service interrupt processor (SI.) in performing conditional SI. processing.

Table 2-1
Interrupts and Responses by SI. Processor

Cause of interrupt	Response by SI. processor
Channel end with no device end	Branch and link H.??XIO with R2 = DCA address and R3 = UDT address. If H.??XIO returns this call, R2 = DCA address and processing continues at SI.EXIT.
Device time out	If the kill command was issued, continue processing at SI.UNLNK; otherwise, branch and link H.??XIO with R2 = DCA address and R3 = UDT address. If H.??XIO returns this call, R2 = DCA address, an unrecoverable error condition is marked in the IOQ, actual transfer count in IOQ is zeroed, and the SI.processor continues at SI.UNLNK.
Execute channel program	Set error condition in IOQ if error is indicated. If sense information is required, the sense command is issued and processing continues at SI.EXIT; otherwise, if no error is found and no sense information is required, processing continues at SI.UNLNK.
I/O request complete with error	Issue sense command and continue processing at SI.UNLNK.
Normal sense command with IOQ	If execute channel program was requested, continue processing at SI.UNLNK; otherwise, branch and link to H.??XIO with R2 = DCA address and R3 = UDT address. If H.??XIO returns this call, R2 = DCA address, actual transfer count is computed and updates IOQ, processing continues at SI.UNLNK.
Postprogram controlled interrupt	If PPCI notification is not desired, exit the interrupt level, otherwise, store status doubleword into PPCI notification packet buffer. SI. processor reports PPCI to user PPCI through S.EXEC4 asynchronous notification call. Exit the interrupt level.
Rewind or seek complete	Clear device rewinding or seeking bit, and continue processing at SI.EXIT.
Special sense command with no IOQ	Branch and link to H.??XIO with R2 = DCA address and R3 = UDT address. If H.??XIO returns this call, R2 = DCA address and processing continues at SI.EXIT.
Spurious interrupt when device is not configured	Increment spurious interrupt count, and exit the interrupt level if the service interrupt processor was entered from H.IFXIO, processing continues at SI.EXIT.
Spurious interrupt when device is configured but interrupt is not expected	Branch and link to H.??XIO with R2 = DCA address and R3 = UDT address. If H.??XIO returns this call, R2 = DCA address, SI. processor increments spurious count for the device and the channel, and continues processing at SI.EXIT.

2.10 Lost Interrupt Processor (LI.XIO)

The lost interrupt processor (LI.XIO) is called from S.IOCS5 to take corrective measures when an expected interrupt fails to occur. It is also called from H.IOCS,38 when a kill command is issued to a task and the task has I/O in progress. In both cases, the I/O request is terminated with a Halt I/O (HIO) instruction. If the controller responds to the HIO, SI. performs the required interrupt handling.

The following is the basic logic sequence within the lost interrupt processor (LI.XIO):

- activate the interrupt level
- increment the lost interrupt count if not a kill request
- issue the HIO if it has not already been issued
- branch to service interrupt routine (SI.) if the HIO instruction produced a status stored condition; otherwise,
- block external interrupts
- deactivate the interrupt level
- return to calling routine

If the HIO has already been issued but fails to generate an interrupt, the lost interrupt service routine (LI.XIO) is entered once again and the following actions are taken:

- activate the interrupt level
- increment lost interrupt count if not a kill request
- the device is marked off-line
- the device is marked malfunctioning
- the IOQ request is unlinked from the I/O queue
- the IOQ is deallocated if the HIO was issued as a result of a "KILL"request
- the I/O request is reported as complete with errors if the HIO was issued because of an interrupt which failed to occur
- branch and link to IQ.XIO.1 to flush any pending I/O requests to the failing device
- block external interrupts
- deactivate the interrupt level
- return to calling routine

Entry Conditions

Calling Sequence

The lost interrupt processor (LI.XIO) is called from S.IOCS5 and H.IOCS,38 with interrupts blocked by:

BL	*4W,X1	register X1 contains the device dependent handler HAT address. The address in the HAT table points to the lost interrupt processor (LI.XIO).

Registers

R0	return address
R3	UDT address of device to halt

Exit Conditions

Return Sequence

TRSW	R0

Registers

None

2.11 Execute Channel Program Opcode Processor (EXCPM)

The execute channel program opcode processor (EXCPM) is called by all XIO device handlers to process either physical or logical execute channel program requests.

The basic logic sequence for processing the physical execute channel program request is:

- abort request if requesting task is not privileged
- determine if notification package is required
- build and initialize an IOQ
- store user specified time-out value into IOQ
- store sense buffer information into IOQ if sense information desired
- set up notification packet if required
- return to H.IOCS to link the I/O request

The basic logic sequence for processing the logical execute channel program request is:

- validate legality of IOCD requests and transfer addresses
- compute number of IOCDs required to satisfy the request
- build and initialize an IOQ
- build IOCD list within the IOQ
- build seek data within IOQ
- convert seek data to cylinder/track/sector format
- store user specified time-out value in IOQ
- store sense buffer information into IOQ if sense information desired
- return to H.IOCS to link the I/O request

This entry point is called from the Opcode Processing entry point (OP.) of XIO device dependent handlers (H.??XIO) by branching indirect through the opcode processing table.

Entry Conditions

Calling Sequence

BU	*OPTAB,X2	register X2 contains the opcode word index into OPTAB

Registers

R1	FCB address

Exit Conditions

Return Sequence

BU	IOLINK	if IOCD list valid
(or)		
BU	CABORT	if IOCD list invalid

Registers

R1	FCB address
R3	IOQ address

2.12 SYSGEN Initialization (COM.INIT)

The SYSGEN initialization (COM.INIT) entry point is executed by SYSGEN during MPX-32 image construction and subsequently overlayed. This entry point contains only the standard M.EIR and M.XIR entry and exit macros.

3 Device Dependent Handlers (H.??XIO)

3.1 General Information

XIO device dependent handlers (H.??XIO) are used in conjunction with the XIO common subroutine package (XIO.SUB) to service I/O requests on behalf of the calling routine. Each device dependent handler contains only that logic which is specific to a device or class of devices. Only one copy of any device dependent handler is configured regardless of the number of devices or channels specified with the specific device type. The handler is made part of the resident operating system and proper linkages are established by naming the handler in the DEVICE statement within the SYSGEN directive file.

The device dependent handler HAT table is the means by which linkages are established between the input/output control system (H.IOCS) and the I/O processing routines. The XIO device dependent handlers (H.??XIO) have seven entry points which are defined in the following HAT table and described in the following sections.

```
HAT     DATAW   7         number of entries in table
        ACW     OP.       opcode processor
        ACW     IQ.XIO    I/O queue driver
        ACW     SI.       service interrupt processor
        ACW     LI.XIO    lost interrupt processor
        ACW     PX.       posttransfer processor
        ACW     PRE.SIO   prestart I/O processor
        ACW     SG.       SYSGEN initialization
```

3.2 Entry Point OP. - Opcode Processing

This entry point OP. is actually a subroutine extension of H.IOCS,29, a portion of IOCS logic common to all I/O services which are capable of initiating a physical device access. It is called to process the opcode placed in the file control block (FCB) by the I/O service originally called by the user.

The purpose of OP. is to examine the opcode and other pertinent FCB control specifications, and to indicate to H.IOCS,29 what action is to be taken. In order to indicate what action is to be taken, OP. takes one of the following returns to H.IOCS,29:

```
BU    ILOPCODE    opcode is illegal for this device
BU    SERVCOMP    service complete, no device access required
BU    IOLINK      link request to I/O queue
BU    POSTPROS    link request to I/O queue and postprocessing is required
```

If return 3 (IOLINK) or 4 (POSTPROS) is taken, OP. must first:

1. Call IOCS subroutine S.IOCS13 to allocate and initialize an IOQ.
2. Build into the IOQ entry an I/O command list (IOCL) with the proper command codes and flags using IOCS subroutines S.IOCS12 and IOCS entry point H.IOCS,40.

Taking return 4 (POSTPROS) also indicates that after the request has been completed but before return to or notification of the user, the device postprocessing entry point PX. must be called.

Entry Conditions

Calling Sequence

BL	*1W,X2	register X2 contains the address of the device dependent handler HAT table. The one word offset from this address contains the address of OP.

Registers

R1	FCB address
R2	device dependent handler HAT address
R3	UDT address

Exit Conditions

Return Sequence

See descriptions of ILOPCODE, SERVCOMP, IOLINK and POSTPROS above.

Registers

R1	FCB address

3.3 Entry Point IQ.XIO - I/O Queue Driver

The entry point IQ.XIO contains the address of the IQ.XIO common subroutine, all calls to this entry point are funneled to common processing. Refer to the Entry Point IQ.XIO section in Chapter 2.

3.4 Entry Point SI. - Service Interrupt Processor

The entry point SI. contained in the device dependent handlers is entered from the SI. common subroutine whenever device dependent logic should be considered. Six calls are made to device dependent service interrupt processing. Depending upon the specific device, these six entry points can contain executable code; however, they must be included within the device dependent handler to properly interphase with the XIO common subroutines. Device dependent service interrupt calls occur:

- when I/O completes normally or when status checking is inhibited
- when status contains channel end with no device end
- following a normal sense command (with IOQ)
- following an unexpected interrupt
- following a device time out
- following a sense command (without IOQ)

Entry Conditions

Calling Sequence

BL	NW,X1	register X1 is the SI. entry point address within the device dependent handler logic as contained in the HAT table. NW specifies an N word offset into an SI. device dependent entry point jump table. The jump table contains an unconditional branch to the desired executable code.

Registers

R1	device dependent handler SI. address
R2	DCA address
R3	UDT address

Exit Conditions

Return Sequence

TRSW	R0	if typical interrupt post access processing is desired for the interrupt type
(or)		
BU	SI.UNLNK	if the I/O request is to be unlinked, reported complete, and I/O queue processing continued
(or)		
BU	SI.EXIT	if the interrupt level is to be exited without unlinking the I/O request and reporting the I/O complete. An example of this would be following a branch and link to IQ.XIO.2 (BL IQ.XIO.2) to initiate error retry or collect sense information.

Registers

R1	IOQ address (required for SI.UNLNK only)
R2	DCA address

3.5 Normal Completion or Status Checking Inhibited

This routine is called by the XIO common subroutine when an I/O request completes with no errors or when status checking is inhibited. It can be used to perform any device specific processing necessary under these conditions. An example would be the collection of sense information pertinent to the I/O operation just completed. The XIO common routines collect sense information only when an I/O request produces an error or the I/O request was for an execute channel program and sense information was requested.

A TRSW R0 return from this routine:

- unlinks the IOQ from the I/O queue
- reports I/O complete
- continues driving the I/O queue
- exits the interrupt level by S.EXEC5

3.6 Channel End with No Device End

This routine is called by the XIO common subroutine when an I/O request produces an interrupt whose status contains channel end and no device end. This is not a normal case for most device types. However, it does occur for magnetic tapes when a rewind is initiated and for XIO discs when a reserve is issued to a dual-ported disc that is reserved to the opposing CPU.

A TRSW R0 return from this routine waits for the device end interrupt.

3.7 Normal Sense Command with IOQ

This routine is called by the XIO common subroutine following an interrupt caused by issuing a sense command on behalf of an I/O request that completed with an error indication. This is the routine that will normally examine the status and sense information and initiate error recovery if applicable.

A TRSW R0 return from this routine:

- computes actual transfer count if an error condition is indicated and update the IOQ transfer count
- unlinks the IOQ from the I/O queue
- reports I/O complete
- continues driving the I/O queue
- exits the interrupt level by S.EXEC5

It should be noted that the TRSW R0 return does not set the error condition flag in the IOQ.

3.8 Unexpected Interrupt

This routine is called by the XIO common subroutine when an interrupt occurs that was not expected. This routine allows for ring in or wake-up logic applicable to some devices. Other applications can apply.

A TRSW R0 return from this routine:

- increments spurious interrupt count for the device
- increments spurious interrupt count for the channel
- continues driving the I/O queue if required
- exits the interrupt level by S.EXEC5

3.9 Device Time Out

This routine is called by the XIO common subroutine following the interrupt generated by issuing a halt I/O (HIO) instruction on behalf of a device that timed out. This routine allows for device dependent recovery from I/O requests that fail to complete.

A TRSW R0 return from this routine:

- sets the error condition in the IOQ
- sets the time out flag in the IOQ
- zeros the transfer count in the IOQ
- unlinks the IOQ from the I/O queue
- reports I/O complete
- continues driving the I/O queue
- exits the interrupt level by S.EXEC5

3.10 Sense Command without IOQ

This routine is called by the XIO common subroutine following an interrupt generated by issuing a sense command with no IOQ associated with the request. This routine may be used in conjunction with a ring in or wake-up capability applicable to some devices. Other applications can apply.

A TRSW R0 return from this routine continues driving the I/O queue, and exits the interrupt level by S.EXEC5.

3.11 Entry Point LI.XIO - Lost Interrupt Processor

The lost interrupt processor HAT table entry point contains the address of the LI.XIO common subroutine so, all calls to this entry point are funneled to common processing. Refer to the Lost Interrupt Processor (LI.XIO) section in Chapter 2 for details.

3.12 Entry Point PX. - Posttransfer Processing

The entry point PX. is called by S.IOCS1 whenever return from the opcode processor (OP.) is by POSTPROS. It is used to perform any processing after completion of the I/O request but before returning to the requesting task. This entry point executes at the task priority and at a low level of system overhead.

Entry Conditions

Calling Sequence

```
BL      *5W,X2
```

Registers

R1	FCB address
R2	device dependent handler HAT address
R3	UDT address

Exit Conditions

Return Sequence

```
TRSW    R0
```

Registers

R1	FCB address

3.13 Entry Point PRE.SIO - Prestart I/O Processor

The entry point PRE.SIO is called by the XIO common I/O queue driver (IQ.XIO) just prior to issuing the start I/O (SIO) request. It allows for modification to the I/O request such as the IOCD list, subchannel number, time-out value, etc. It also allows for removal of the I/O request as may be necessary when a release is issued to a device that is still reserved by another task.

Entry Conditions

Calling Sequence

```
BL      *6W,X3
```

Registers

R1	IOQ address
R2	DCA address
R3	device dependent handler HAT table address
R7	IQ.XIO return address

Exit Conditions

Return Sequence

TRSW	R0	if SIO is to be issued
(or)		
BU	SI.UNLNK	if I/O request is to be removed from the I/O queue

Registers

R1	IOQ address
R2	DCA address
R7	IQ.XIO return address as passed to this routine

3.14 Entry Point SG. - SYSGEN Initialization

The entry point SG. is called by SYSGEN to initialize certain handler parameters, device context areas (DCAs), and data structure elements during the construction of an MPX-32 image. A maximum number of DCA's are created via the repeated assembly of the macro DCA.DATA. During the execution of this entry point, one DCA is initialized for each UDT entry containing the name of the handler. Any remaining DCA's and the remainder of the code in the handler will be overlayed by SYSGEN.

Entry Conditions

Calling Sequence

BL	last entry point, and it is computed from information in the HAT table

Registers

None

Exit Conditions

Return Sequence

M.XIR	this is the standard handler SYSGEN exit macro

Registers

None

4 XIO Interrupt Fielder (H.IFXIO)

4.1 General Information

The XIO interrupt fielder (H.IFXIO) is used in conjunction with the XIO common subroutine package to service I/O interrupts generated by completing I/O requests. One copy of the interrupt fielder is required for each channel connected to an extended I/O (XIO) device. The interrupt fielder is made part of the resident operating system and proper linkages are established by naming the interrupt fielder in the CONTROLLER statement within the SYSGEN directive file.

The XIO interrupt fielder (H.IFXIO) has three entry points which are defined in the following HAT table and described in the following sections.

```
HAT     DATAW    3      number of entries in table
        ACW      H1.    interrupt fielding entry point
        ACW      H2.    channel initialization
        ACW      HI.    SYSGEN initialization entry point
```

4.2 Entry Point H1. - Interrupt Fielder

The entry point H1. is entered by the service interrupt (SI) vector address contained in scratchpad and performs the following functions:

- increments the global interrupt count
- fetches the channel definition table (CHT) address associated with the interrupt level
- saves registers
- branches to the SI. common subroutine entry point

Entry Conditions

Calling Sequence

entered as a result of an interrupt

Registers

None

Exit Conditions

Return Sequence

BU	SI.	this is the SI. common subroutine entry point

Registers

R3	CHT address associated with the interrupt level

4.3 Entry Point H2. - Initialize Channel

The entry point H2. is entered from the opcode processing (OP.) entry point of a device dependent handler (H.??XIO) as a result of the first open request to a device connected to the channel. H2. can reset, initialize and enable the channel.

Entry Conditions

Calling Sequence

BL	*CHT.INCH,X2

Registers

R1	FCB address
R2	CHT address
R3	DCA address

Exit Conditions

Return Sequence

TRSW	R0	error return, channel not initialized and all devices have been marked off-line
TRSW	R0+1W	initialization complete

Registers

R1	FCB address
R2	CHT address (initialization complete return only)
R3	DCA address (initialization complete return only)

4.4 Entry Point HI. - SYSGEN Initialization

The entry point HI. is called by SYSGEN to initialize certain interrupt fielder parameters and data structure elements during the construction of an MPX-32 image. It also provides memory locations required for disc and tape controllers at channel initialization. Once executed, this entry point is overlayed by SYSGEN, except for the memory locations needed by the disc and tape.

Entry Conditions

Calling Sequence

BL last entry point

Registers

R3 CHT address

Exit Conditions

Return Sequence

M.XIR this is the standard handler SYSGEN exit macro

Registers

None

MPX-32™ Resident Modules & Handlers
Technical Manual
No. 322-001552-500

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