# Gould MPX-32™

# Release 3.3

# Technical Manual

# Volume II

December 1986

Publication Order Number: 322-001552-200



GOULD
Electronics

## PROPRIETARY INFORMATION



## RESTRICTED RIGHTS LEGEND



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

## HISTORY

The Gould 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 Gould MPX-32 Release 3.3 Technical Manual Volume II, Publication Order Number **322-001552-200** (Revision 2, Release 3.3) was printed December, 1986.

This manual contains the following pages:

PART II - HANDLER DESCRIPTIONS

# CONTENTS

## GOULD MPX-32 TECHNICAL MANUAL

## VOLUME II OVERVIEW



## RESOURCE ALLOCATOR (H.ALOC)

## BLOCKED DATA MANAGEMENT MODULE (H.BKDM)



## EXECUTIVE (H.EXEC)

## Figures

## FILE SYSTEM EXECUTIVE (H.FISE)

Section Page

### 1 - OVERVIEW



### 2 - ENTRY POINTS

## INPUT/OUTPUT CONTROL SYSTEM (H.IOCS)

Section Page

### 1 - OVERVIEW



### 2 - ENTRY POINTS

## INTERRUPT AND TRAP PROCESSORS (H.IP?? and H.SVC?)

### 1 - OVERVIEW



### 2 - INTERRUPT PROCESSORS



### 3 - TRAP PROCESSORS

## RAPID FILE ALLOCATION (H.MDT)



## MEMORY MANAGEMENT (H.MEMM)

## MEMORY POOL MANAGEMENT (H.MEMM2)

## SYSTEM SERVICES (H.MONS)

Section Page

### 1 - OVERVIEW



### 2 - ENTRY POINTS

## MULTIVOLUME MAGNETIC TAPE (H.MVMT)

### 1 - OVERVIEW



### 2 - ENTRY POINTS

## RESOURCE MANAGEMENT (H.REMM)

Section Page

### 1 - OVERVIEW



### 2 - ENTRY POINTS



### 3 - SUBROUTINES

## RESIDENT EXECUTIVE SERVICES (H.REXS)

### 1 - OVERVIEW



### 2 - ENTRY POINTS

## TASK MANAGEMENT (H.TAMM)

## TERMINAL SERVICES (H.TSM)

Section Page

## VOLUME MANAGEMENT MODULE (H.VOMM)

Section Page

### 1 - OVERVIEW



### 2 - ENTRY POINTS



### 3 - SUBROUTINES

## EXTENDED I/O DISC HANDLER (H.DCXIO)

### 1 - OVERVIEW



### 2 - COMMANDS

## Figures



## Table



## IOP EIGHT-LINE FULL DUPLEX HANDLER (H.F8XIO)

## GENERAL PURPOSE MULTIPLEXER SUPPORT (H.GPMCS)

Section Page

### 1 - OVERVIEW



### 2 - USAGE

## HIGH-SPEED DATA HANDLER (H.HSDG)

## MEMORY DISC HANDLER (H.MDXIO)

## XIO COMMON SUBROUTINE PACKAGE & DEVICE HANDLERS (H.XIOS)

## Figure



## Table

# GOULD MPX-32 TECHNICAL MANUAL

## VOLUME II OVERVIEW

### SECTION 1 - USING THE MANUAL

The information in this manual is divided into two parts:

| | |
|---|---|
| Part I | Module Descriptions |
| Part II | Handler Descriptions |

In Part I, each module has a self-contained description that is prefaced by an amber tabbed page. Part I contains the following module descriptions:

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

Each module description has the following format as applicable:

Overview
Entry Points
Subroutines

In Part II, each handler has a self-contained description that is prefaced by a blue tabbed page. Part II contains the following descriptions:

H.DCXIO
H.F8XIO
H.GPMCS
H.HSDG
H.MDXIO
H.XIOS

Each handler description has the following format, as applicable:

Overview
Usage
Entry Points
Subroutines

To tailor this manual to a particular system, remove the descriptions of modules or handlers that are not installed on the system. Any tabbed pages that are out of sync can be flipped over.

## SECTION II - DOCUMENTATION CONVENTIONS

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

**lowercase letters**

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

```
!ACTIVATE taskname
```

means replace taskname with the name of a task, e.g.,

```
!ACTIVATE DOCCONV
```

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

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

means a break occurred on the specified task.

**UPPERCASE LETTERS**

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

```
SAVE filename
```

means enter SAVE followed by a filename, e.g.,

```
SAVE DOCCONV
```

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

```
taskname,taskno ABORTED

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

**Braces { }**

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

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

means enter the value for either counter or startbyte.

**Brackets [ ]**

An element inside brackets is optional. For example,

```
[CURR]
```

means the term CURR is optional.

Items placed one under the other within brackets specify you may optionally enter one of the group of options or none at all. For example,

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

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

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

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

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

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

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

could be coded as

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

or

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

or

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

**Horizontal Ellipsis . . .**

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

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

means you may enter one or more name values separated by commas.

**Vertical Ellipsis**

.
.
.

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

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

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

**Numbers and Special Characters**

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

(value)

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

**Underscore**

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

ACTIVATE taskname

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

RESET

means type either RESET or RST.

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

TSM > EDIT

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

**Subscript Delta** $_{\Delta}$

A subscript delta specifies a required space. For example,

EDT > STO$_{\Delta}$TSSPGM

means a space is required between O and T.

# Resource Allocator (H.ALOC)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# RESOURCE ALLOCATOR (H.ALOC)

## SECTION 1 - 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 disc 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 |

# SECTION 2 - 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, register one is unchanged. |
| | CC1 | Set if allocation denied:<br>R6 contains the scan mask<br>R7 contains the device requirements mask |
| | CC2 | Set if allocation error:<br>R6 contains an error code<br>R7 contains zero |

### Error Conditions

Register: 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 References

| | |
|---|---|
| System Services: | H.REMM,6<br>H.REXS,20<br>H.REXS,76 |
| System Subroutine: | S.REXS8 |
| System Macros: | M.CALL<br>M.RTRN |

### 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 | One- to three-character right-justified ASCII logical file code |

Exit Conditions

| | |
|---|---|
| Return Sequence: | M.RTRN |
| Registers: | None |

Error Conditions

| | |
|---|---|
| CC1 | Set if unrecoverable I/O error to directory |

External References

| | |
|---|---|
| System Service: | H.REMM,7 |
| System Subroutine: | S.REXS8 |
| System Macros: | M.CALL<br>M.RTRN |

### 2.8 Entry Point 8 - Get Dynamic Extended Data Space

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

External References

| | |
|---|---|
| System Service: | H.REMM,12 |
| System Subroutines: | S.REXS8<br>S.REXS9<br>S.ALOC91 |
| System Macros: | M.CALL<br>M.RTRN |

## 2.13 Entry Point 13 - Get Shared Memory (INCLUDE)

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

External References

| | |
|---|---|
| System Service: | H.REMM,12 |
| System Subroutines: | S.REXS8<br>S.REXS9 |
| System Macros: | M.CALL<br>M.RTRN |

## 2.14 Entry Point 14 - Free Shared Memory (EXCLUDE)

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

External References

| | |
|---|---|
| System Service: | H.REMM,14 |
| System Subroutine: | S.REXS8 |
| System Macros: | M.CALL<br>M.RTRN |

## 2.15 Entry Point 15 - Reserved

## 2.16 Entry Point 16 - Reserved

## 2.17 Entry Point 17 - Allocate Disc File By Space Definition

This entry point creates a multiword 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 References

| | |
|---|---|
| System Service: | H.REXS,21 |
| System Subroutine: | S.REXS8 |
| System Macros: | M.CALL<br>M.RTRN |

## 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 for a detailed description of this entry point.

External References

| | |
|---|---|
| System Service: | H.REMM,24 |
| System Subroutine: | S.ALOC91 |
| System Macros: | M.CALL<br>M.RTRN<br>M.SHUT<br>M.OPEN |

## 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 References

| | |
|---|---|
| System Subroutines: | S.REMM12<br>S.REXS8 |
| System Macro: | M.RTRN |

## 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.

# SECTION 3 - 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 | 0 |
| | 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

# BLOCKED DATA MANAGEMENT MODULE (H.BKDM)

## SECTION 1 - 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 processinn |

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 |

# SECTION 2 - 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 |
| Register: | 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 |
| Register: | R1 | FCB address |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Register: | R1 | FCB address |

# SECTION 3 - 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 |
| Register: | 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 |
| Register: | 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 from the Scratchpad

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 |
| Register: | R1 | FCB address |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Registers: | R2 | Address of saved parameters |
| | R4,R6 | Destroyed |

# Executive (H.EXEC)

# MPX-32 Technical Manual

# Volume II

## CONTENTS

## 3 - SUBROUTINES

## Figures

# EXECUTIVE (H.EXEC)

## SECTION 1 - OVERVIEW

### 1.1 General Information

The Executive Module (H.EXEC) performs as a CPU scheduler, by allocating the CPU and IPU to tasks.

### 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 disc file space |
| H.EXEC,15 | N/A | Report peripheral resource available |
| H.EXEC,16 | N/A | Report disc 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 (MPXDB) |
| H.EXEC,23 | N/A | Run user break receiver (MPXDB) |
| H.EXEC,24 | N/A | Restart debug (MPXDB) |
| 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 MPXDB mode break |
| H.EXEC,31 | N/A | Hold current task |
| H.EXEC,32 | N/A | Hold specified task |

| Entry point | SVC number | Description |
|---|---|---|
| H.EXEC,33 | N/A | Disable MPXDB 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 |
| 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. 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.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.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.EXEC20 | CPU scheduler |
| S.EXEC21 | Process task interrupt |
| S.EXEC22 | Wait for completion of all no-wait operations |
| S.EXEC23 | Terminate messages in receiver queue |
| S.EXEC24 | Reserved |
| S.EXEC25 | Terminate next run request in receiver queue |
| S.EXEC26 | Remove task gating |
| S.EXEC27 | Transfer control to abort receiver |
| S.EXEC28 | Delete task processing control |
| S.EXEC29 | Exit task processing control |

| Subroutine | Description |
|---|---|
| S.EXEC30 | Reserved |
| S.EXEC31 | Report no-wait run request postprocessing complete |
| S.EXEC32 | Report wait mode run request complete |
| S.EXEC33 | Report no-wait mode run request complete |
| S.EXEC34 | Reserved |
| S.EXEC35 | Report no-wait mode message postprocessing 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.EXEC40 | Reserved |
| S.EXEC41 | Exit run receiver |
| S.EXEC42 | Exit message receiver |
| S.EXEC43 | Reactivate run receiver task |
| S.EXEC44 | Change priority level of current task |
| S.EXEC45 | Change priority level of specified task |
| S.EXEC46 | Reserved |
| S.EXEC47 | Reserved |
| 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.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.EXEC58 | Link MRRQ to message receiver of destination task |
| S.EXEC59 | Reserved |
| S.EXEC60 | Validate PRB |
| S.EXEC61 | Transfer parameters from MRRQ to receiver buffer |
| S.EXEC62 | Validate RXB |
| 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.EXEC68 | Construct and vector context to end action PSD |
| S.EXEC69 | Common no-wait postprocessing merge point |
| S.EXEC70 | Terminate all run requests in receiver queue of current task |
| S.EXEC71 | Insure start-up of destination run receiver task |
| S.EXEC72 | Report wait I/O starting |
| S.EXEC73 | Replace context on TSA stack |
| S.EXEC74 | Reset stack to user level |
| S.EXEC75 | Situational priority increment |
| S.EXEC76 | Update task execution accounting value |
| S.EXEC77 | Reserved |
| S.EXEC78 | Move context from stack to T.CONTXT |
| S.EXEC79 | Push current context onto stack for deferred EA pull |
| S.EXEC80 | Start IPU and verify |

# SECTION 2 - 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 will be removed from the associated ready-to-run list, and placed in the wait for interactive input list. A return to the calling routine will be made upon completion of the input request.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | M.SHUT<br>UEI<br>M.CALL | <br><br>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 will be removed from the associated ready-to-run list, and placed in the wait for terminal output list. A return to the calling routine will be made upon completion of the output request.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | M.SHUT<br>UEI<br>M.CALL | <br><br>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 will be removed from the associated ready-to-run list, and placed in the wait for I/O list. A return to the calling routine will be made upon completion of the I/O request.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | M.SHUT<br>UEI<br>M.CALL | <br><br>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<br>UEI<br>M.CALL | <br><br>H.EXEC,4 |
| Registers: | R0 Bit 0 | One indicates task is swappable during I/O processing |

Exit Conditions

| | |
|---|---|
| Return Sequence: | M.OPEN<br>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 will be processed before a return is made to the calling routine. The purpose of this entry point is for IOCS usage when waiting for the completion of a particular no-wait I/O request.

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 may 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 will be processed by H.EXEC,7 according to request type. The request type information is contained in the DQE. 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 such that it is either deallocated, or becomes more eligible for swapping. The purpose of this routine 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: | 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. The purpose of this routine 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 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 posttransfer 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. 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, 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 will be 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 Disc 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 disc list. A return to the calling routine is made when any disc file space is deallocated. The calling routine may then make another attempt to allocate the required disc file space.

Entry Conditions

Calling Sequence: M.CALL H.EXEC,14

Registers: R6 Disc device requirements specification:

| Bits | Definition |
|---|---|
| 0-7 | Reserved |
| 8-15 | Device type code |
| 16-23 | Channel address |
| 24-31 | Subchannel address |

R7 Disc device requirements mask:

| Value | Definition |
|---|---|
| X'00000000' | Any disc |
| X'00FF0000' | Any disc of the specified type code |
| X'00FFFF00' | Any disc of the specified type code on the specified channel |
| X'00FFFFFF' | The specific disc 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. The purpose of this routine is to resume 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. It is expected that when these tasks resume execution, they will 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 Disc File Space Available

This entry point is called when disc space is deallocated. The purpose of this routine is to resume the execution of the tasks in the wait for disc list which have specified requirements that may be satisfied by the deallocated disc 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. It is expected that when these tasks resume execution, they will 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 | Disc 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 (MPXDB)

This entry point is called by MPXDB 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 MPXDB 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 | Registers 0-7 |
| 8-9 | PSD |

The context block must be word bounded.

Exit Conditions

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

## 2.23 Entry Point 23 - Run User Break Receiver (MPXDB)

This entry point is called by MPXDB to initiate execution of the user break receiver. The contents of T.CONTXT are pushed onto the TSA stack. The MPXDB 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 Condition:

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

## 2.24 Entry Point 24 - Restart MPXDB (MPXDB)

This entry point is called by MPXDB to restart MPXDB execution. All outstanding I/O requests are terminated; any outstanding messages are discarded; the stack is cleared, and control is passed to MPXDB entry point 2.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | M.CALL | H.EXEC,24 |
| Registers: | None | |

Exit Conditions

| | |
|---|---|
| Return Sequence: | M.RTNA |
| Registers: | None |

## 2.25 Entry Point 25 - Wait for Any No-Wait Operation Complete, Message Interrupt or Break Interrupt

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 may be either indefinite in length, or may have an associated time-out value. A return will not be 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 will be reset. A return to the calling routine will then be made.

Entry Conditions

| | | |
|---|---|---|
| Return 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 will remain 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 MPXDB Mode Break

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

This entry point is called by the MPXDB program to disable a break while the task is in MPXDB mode. This routine is provided for use in conjunction with H.EXEC,30 (Enable MPXDB Mode Break). Normally, MPXDB 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 will be made upon completion of message processing by the destination task.

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: R7 Memory request definition word:

Byte zero specifies the memory request type:

| Value | Definition |
|---|---|
| 0 | Inswap task |
| 1 | Preactivation 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) |

Byte one specifies the type of memory required:

| Value | Memory Class |
|---|---|
| 1 | E |
| 2 | H |
| 3 | S |
| 4 | H1 (CPU shadow) |
| 5 | H2 (IPU shadow) |

Byte two specifies the map register to be used; subtract the contents of C.MSD from the map register number (0-31).

Byte three specifies the number of memory blocks required

or

R7 Shared memory request definition word:

Byte zero specifies the memory request type:

| Value | Definition |
|---|---|
| 5 | Shared memory request |

Bytes one through three specify the address of 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 for a detailed description of this entry point.

External references

| | |
|---|---|
| System Macro: | M.RTRN |

## 2.41 Entry Point 41 - Get User Context

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

## 2.43 Entry Point 43 - Reserved for Symbolic Debugger/X32

# SECTION 3 - 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<br>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<br>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 will be 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 pre-emptive system service. Pre-emptive 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 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: | 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)<br>BL | S.EXEC9 |
| Registers: | R1<br>R2 | Head cell address<br>Address of entry to be unlinked |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Registers: | R2,R4-R7<br>R1,R3 | Saved<br>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)<br>BL | S.EXEC10 |
| Registers: | R1<br>R2 | Head cell address<br>Address of entry to be linked |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Registers: | R1,R2,R4-R7<br>R3 | Saved<br>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.

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 |

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

**Figure 3-1. Standard Linked List Head Cell Format**

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, it 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, it 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.

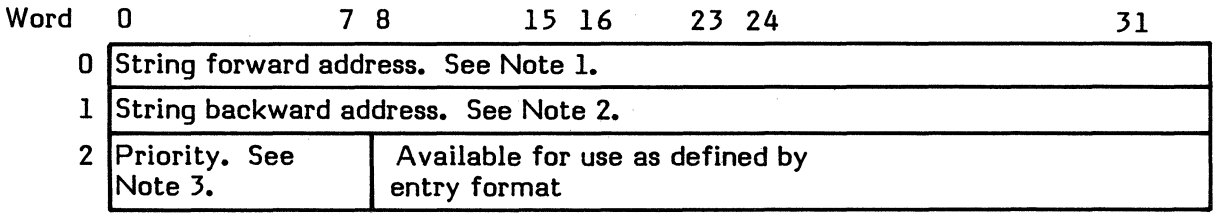



**Figure 3-2. Standard Linked List Entry Header Format**

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.

Misc. 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 so that 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 MPXDB-break requested flag in the DQE if MPXDB is associated with the task and if the MPXDB mode active task interrupt flag is not already set. If MPXDB 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 MPXDB 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 may 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.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

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

### 3.18 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 | |

### 3.19 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 |

### 3.20 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 is 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 |

## 3.21 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 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.REXS,38 | Disconnect interrupt (if interrupt connected) |
| S.EXEC76 | Update execution accounting value |
| [ 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<br>BU | C.PRNO<br>S.EXEC20 |
| Registers: | None | |

### 3.22 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

Dispatch of CPU control

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

1. Task scheduled for execution (see Figure 3-3)
2. Context switch inhibited, current task resumed (see Figure 3-4)
3. Time quantum two has expired (see Figure 3-5)
4. Higher priority task requested processor (see Figure 3-6)
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.

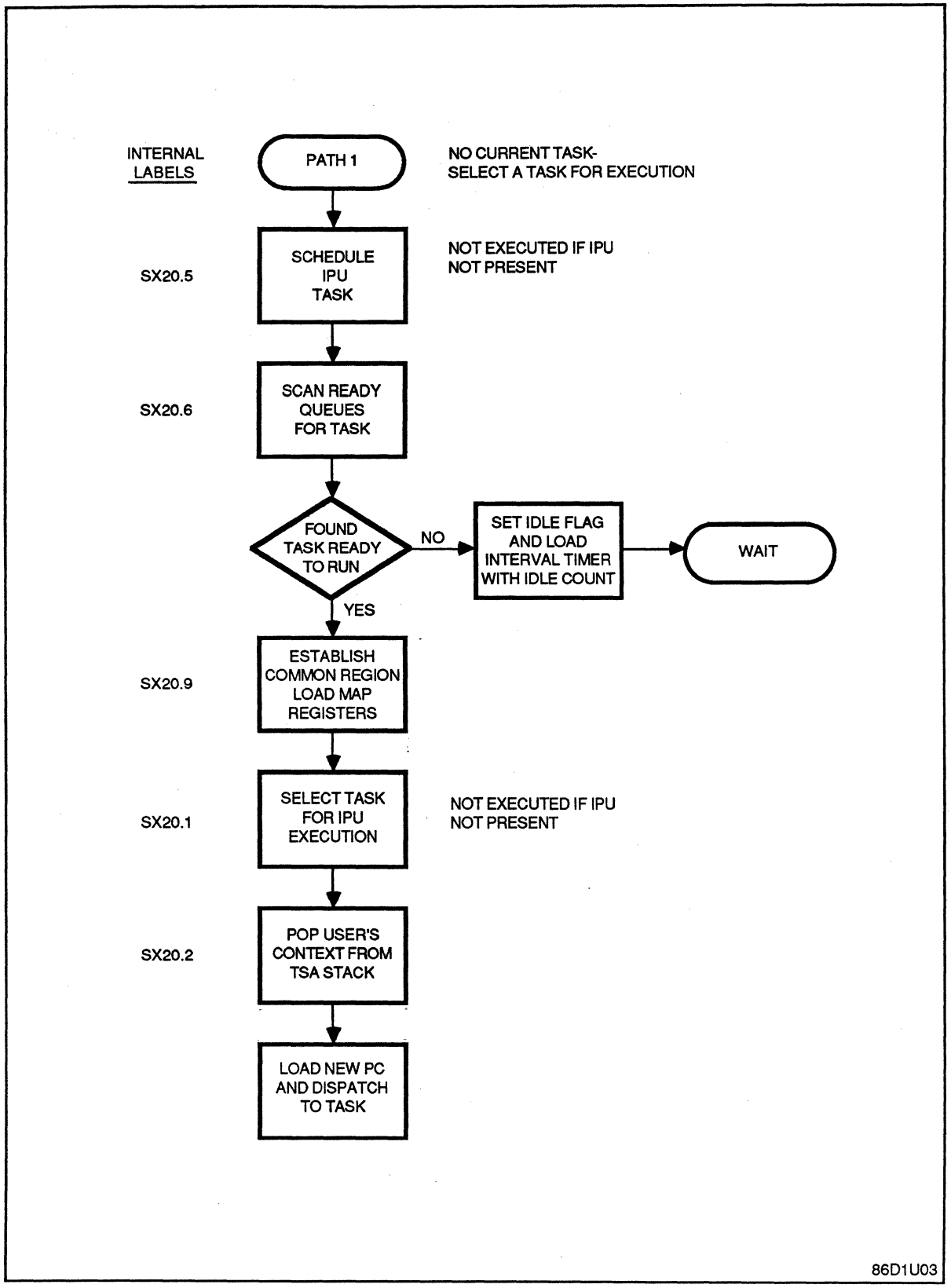



**Figure 3-3. S.EXEC20 Path One**

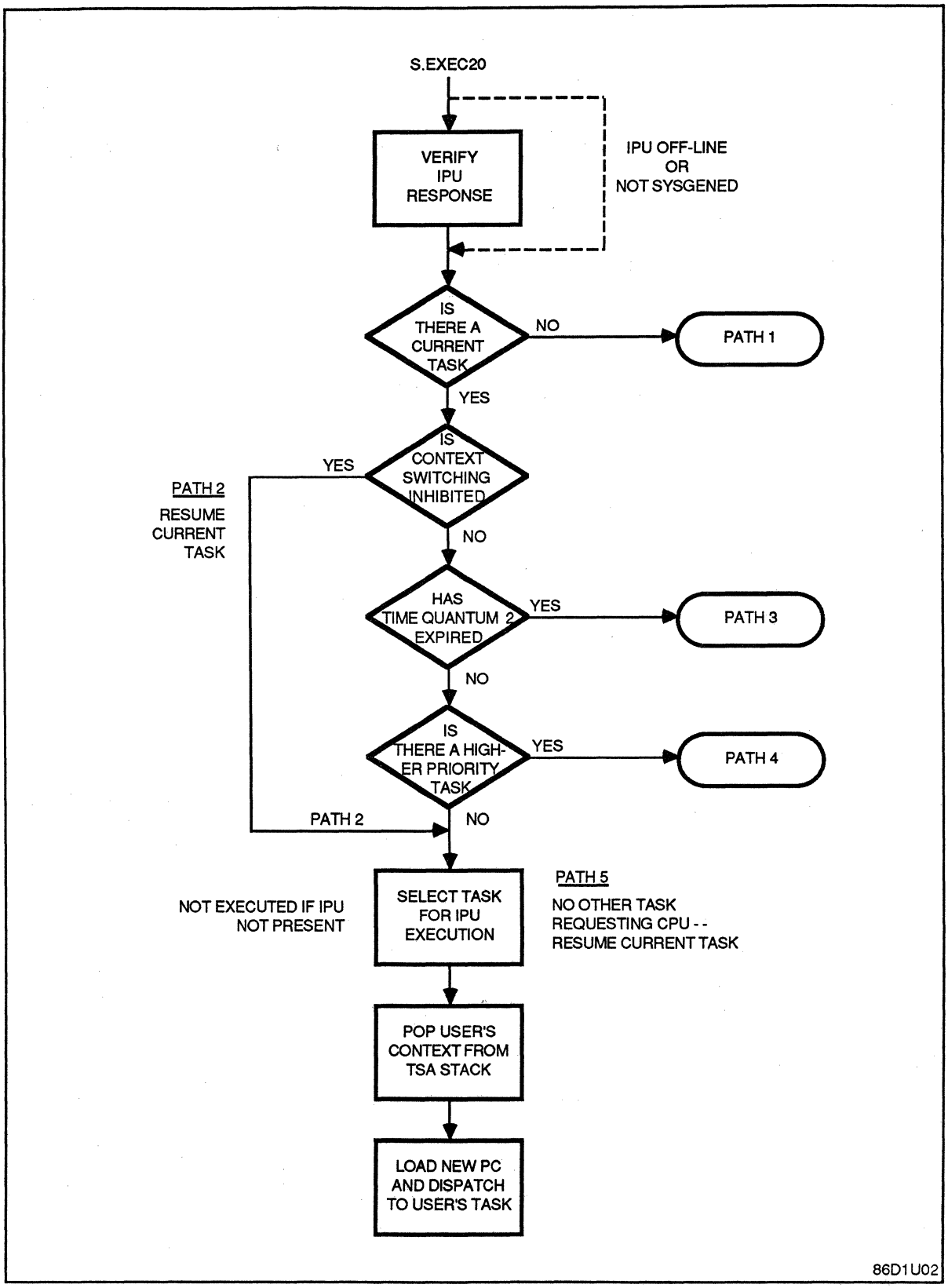



**Figure 3-4. S.EXEC20 Paths Two and Five**

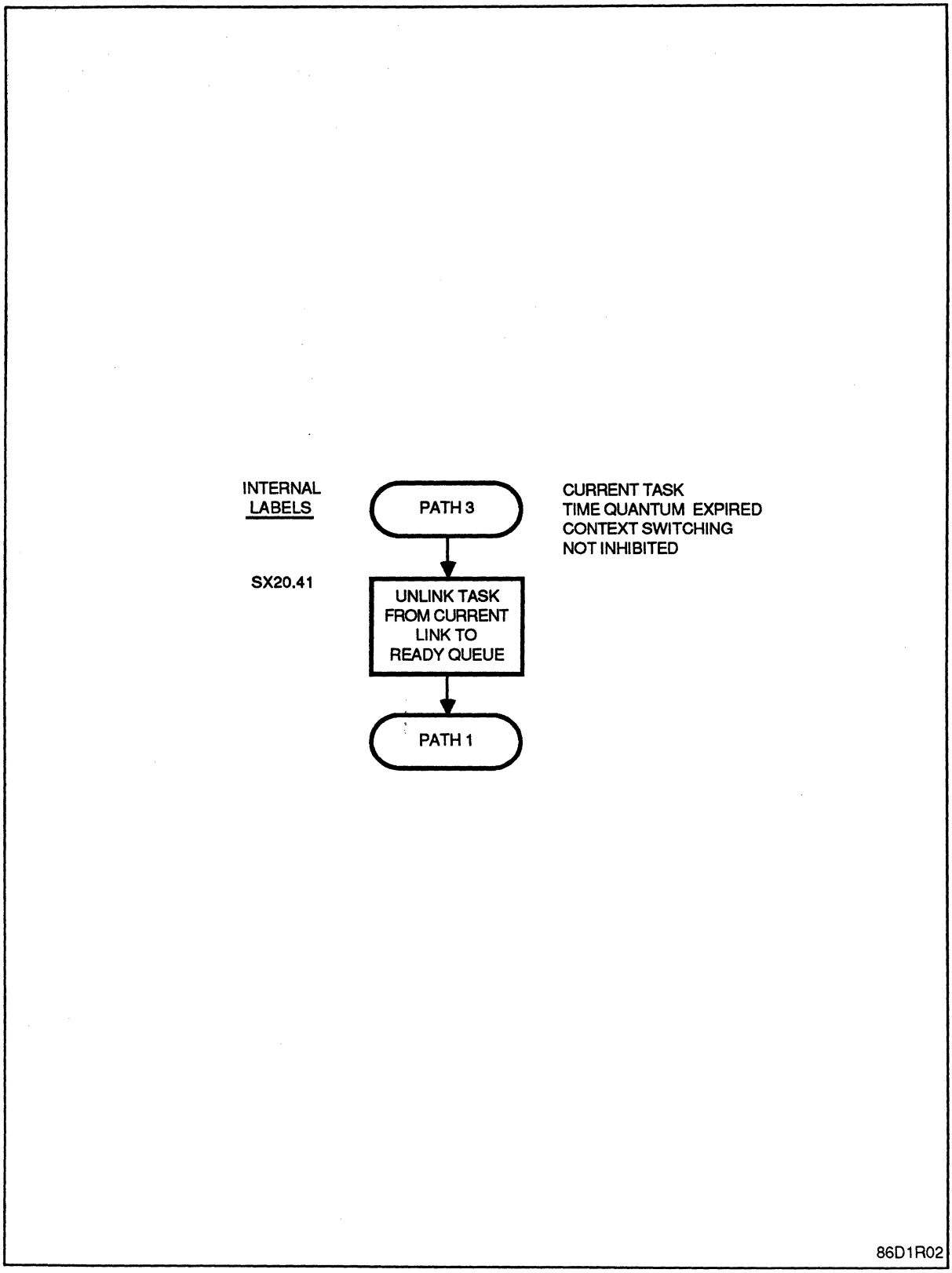



**Figure 3-5. S.EXEC20 Path Three**

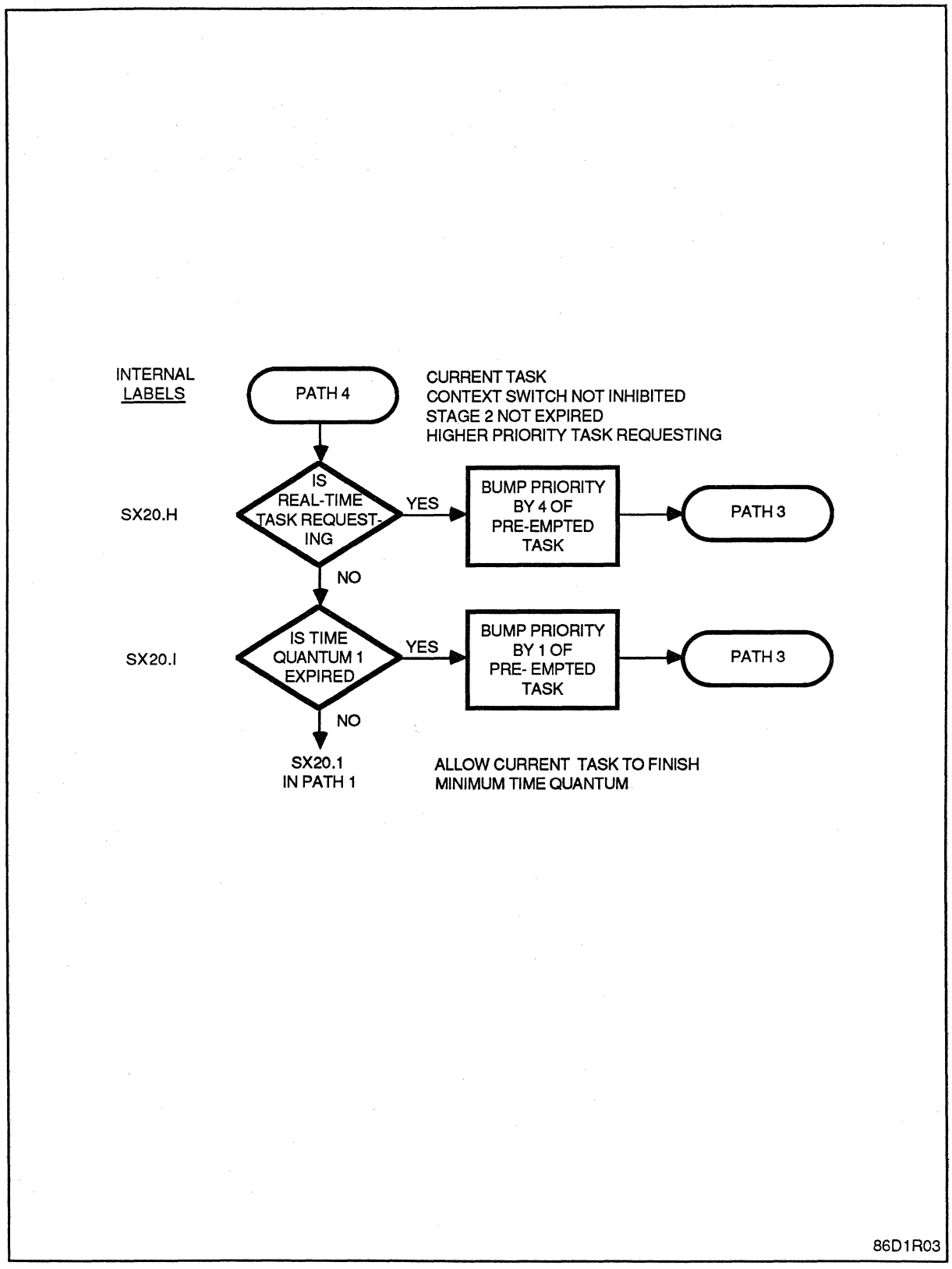



**Figure 3-6. S.EXEC20 Path Four**

### 3.23 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 |

### 3.24 Subroutine S.EXEC22 - Wait for 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 |

### 3.25 Subroutine S.EXEC23 - Terminate 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 | |

### 3.26 Subroutine S.EXEC24 - Reserved

### 3.27 Subroutine S.EXEC25 - Terminate Next Run Request 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 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 |

### 3.28 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 preserved | |

### 3.29 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 | |

### 3.30 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 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. |
| 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<br>BU | C.PRNO<br>S.EXEC20 |
| Registers: | None | |

### 3.31 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 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 |
| S.EXEC22 | Defer continued processing until all outstanding no-wait operations are complete |
| S.EXEC25 | Terminate active run request, if any |
| H.REMM3 | 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<br>BU | C.PRNO<br>S.EXEC20 |
| Registers: | None | |

### 3.32 Subroutine S.EXEC30 - Reserved

### 3.33 Subroutine S.EXEC31 - Report No-Wait Run Request Postprocessing Complete

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 will discard one level (the most recent) of push down in the TSA. A M.RTRN will then be 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 | |

### 3.34 Subroutine S.EXEC32 - Report Wait Mode Run Request Complete

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 |

### 3.35 Subroutine S.EXEC33 - Report No-Wait Mode Run Request Complete

This routine is called to report 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 pre-emptive 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 |

### 3.36 Subroutine S.EXEC34 - Reserved

### 3.37 Subroutine S.EXEC35 - Report No-Wait Mode Message Postprocessing Complete

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 one level (the most recent) 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 | |

### 3.38 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 |

### 3.39 Subroutine S.EXEC37 - Report No-Wait Mode Message 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 pre-emptive 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 |

### 3.40 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 |

### 3.41 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 |

### 3.42 Subroutine S.EXEC40 - Reserved

### 3.43 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) |

### 3.44 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) |

### 3.45 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 destroyed | |

### 3.46 Subroutine S.EXEC44 - Change Priority Level 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 |

### 3.47 Subroutine S.EXEC45 - Change Priority Level 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 will be 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 destroyed | |

### 3.48 Subroutine S.EXEC46 - Reserved

### 3.49 Subroutine S.EXEC47 - Reserved

### 3.50 Subroutine S.EXEC48 - Convert Task 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 |

### 3.51 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 |

### 3.52 Subroutine S.EXEC50 - Link MRRQ to Run Receiver of Destination Task

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 will be 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 |

### 3.53 Subroutine S.EXEC51 - Link Current Task to Designated 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 |

### 3.54 Subroutine S.EXEC52 - Message or Run Request Postprocessing

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 |

### 3.55 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 |

### 3.56 Subroutine S.EXEC54 - Move Byte String

This routine is a register re-entrant 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 |
| 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 |

### 3.57 Subroutine S.EXEC55 - Unlink Task from Designated List and Link to Ready List

This routine is called to link a task to the ready-to-run queue. It will unlink the task from the designated list, then link 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 will be linked to the memory request queue and the memory scheduler will be 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 |

### 3.58 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 |

### 3.59 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 |

### 3.60 Subroutine S.EXEC58 - Link MRRQ to Message Receiver of Destination Task

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 |

### 3.61 Subroutine S.EXEC59 - Reserved

### 3.62 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, register six is destroyed |

### 3.63 Subroutine S.EXEC61 - Transfer Parameters from MRRQ to Receiver Buffer

This routine is called for the destination task after a request for message or run request parameters has been made. The sent parameters are transferred by S.EXEC61 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 |

### 3.64 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, register six is destroyed |

### 3.65 Subroutine S.EXEC63 - Transfer Return Parameters from Destination Task to MRRQ

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 |

### 3.66 Subroutine S.EXEC64 - No-Wait Mode Message Postprocessing

This routine is invoked as a pre-emptive 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: | Pre-emptive 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 |

### 3.67 Subroutine S.EXEC65 - No-Wait Mode Run Request Postprocessing

This routine is invoked as a pre-emptive 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: | Pre-emptive 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 |

### 3.68 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 |

### 3.69 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 pre-emptive 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 |

### 3.70 Subroutine S.EXEC68 - Construct and Vector Context to End Action PSD

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 |

### 3.71 Subroutine S.EXEC69 - Common No-Wait 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 |

### 3.72 Subroutine S.EXEC70 - Terminate All Run Requests in Receiver Queue of Current 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 |

### 3.73 Subroutine S.EXEC71 - Insure Start-up of Destination Run Receiver Task

This routine is called to unlink the destination task from the RUNW or PREA list, unless register 5 is not zero, and to link the destination task to the ready list at the priority specified in register six.

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 |

### 3.74 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 register zero bit one 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 |

### 3.75 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 |

### 3.76 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 |

### 3.77 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 |

### 3.78 Subroutine S.EXEC76 - Update Task 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 |

### 3.79 Subroutine S.EXEC77 - Reserved

### 3.80 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 |

### 3.81 Subroutine S.EXEC79 - Push Current Context onto Stack for Deferred EA Pull

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 one. Word two 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 |

### 3.82 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 |

# File System Executive (H.FISE)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# FILE SYSTEM EXECUTIVE (H.FISE)

## SECTION 1 - 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 disc space |
| H.FISE,4 | N/A | Deallocate temporary disc 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

# SECTION 2 - ENTRY POINTS

## 2.1 Entry Point 1 - Reserved

## 2.2 Entry Point 2 - Reserved

## 2.3 Entry Point 3 - Allocate Temporary Disc Space

This entry point is used to effect the allocation of temporary disc 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 disc 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 disc 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 | Register 4 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 References

| | |
|---|---|
| System Services: | H.VOMM,19 |
| System Macros: | M.RTRN<br>M.RTNA |

## 2.4 Entry Point 4 - Deallocate Temporary Disc Space

This entry point is called to release temporary disc 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 disc on which the file resides |
| | R6 | Starting disc address |
| | R7 | Number of 192 word blocks to release |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | M.RTRN | |
| Registers: | None | |
| or | | |
| Return Sequence: | M.RTNA 4 | Register 4 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 References

| | |
|---|---|
| System Services: | H.VOMM,20 |
| System Macros: | M.RTRN<br>M.RTNA |

## 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 one- to eight-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 References

| | |
|---|---|
| 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 for a detailed description of this entry point.

External References

| | |
|---|---|
| System Services: | H.VOMM,1<br>H.REXS,20 |
| System Subroutines: | S.REXS8<br>S.REXS9 |
| System Macros: | M.CALL<br>M.RTRN |

## 2.13 Entry Point 13 - Change Temporary File To Permanent

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

External References

| | |
|---|---|
| System Services: | H.VOMM,9<br>H.VOMM,11<br>H.VOMM,12<br>H.REXS,20 |
| System Subroutines: | S.REXS8<br>S.REXS9 |
| System Macros: | M.CALL<br>M.RTRN |

## 2.14 Entry Point 14 - Delete Permanent File or Non-SYSGEN Memory Partition

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

External References

| | |
|---|---|
| System Services: | H.VOMM,5<br>H.REXS,20 |
| System Subroutines: | S.REXS8<br>S.REXS9 |
| System Macros: | M.CALL<br>M.RTRN |

## 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 for a detailed description of this entry point.

External References

| | |
|---|---|
| System Services: | H.VOMM,10 |
| System Subroutines: | S.REXS8<br>S.REXS9 |
| System Macros: | M.CALL<br>M.RTRN |

## 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 disc 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<br>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 |

| | | Bits | Contents |
|---|---|---|---|
| | | 16-23 | Device type code |
| | | 24-31 | Optional device address (if present, bit 16 is also set) |
| | R4,R5 | Password or register 4 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 | Disc 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 disc allocation map | |
| Output Messages: | None | | |

External References

| | |
|---|---|
| System Macros: | M.RTRN |
| | M.CALL |

## 2.21 Entry Point 21 - RTM CALM Change Temporary File to Permanent

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

### 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 register 4 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 errror to disc allocation map |
| Output Messages: | None | |

### External References

| | |
|---|---|
| System Macros: | M.RTRN |
| | M.CALL |

### 2.22 Entry Point 22 - Set Exclusive File Lock

See M.FXLS in the MPX-32 Reference Manual 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 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 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 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

## 3 - SUBROUTINES

# INPUT/OUTPUT CONTROL SYSTEM (H.IOCS)

## SECTION 1 - 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 disc/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 disc/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 |

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

| Entry point | SVC number | Description |
|---|---|---|
| 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 |
| 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.IOCS19 | 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 |
| S.IOCS27 | Validate FCB and perform implicit open |
| S.IOCS28 | Initialize IOQ entry |
| S.IOCS29 | Report I/O complete |
| S.IOCS30 | Initialize FCB |

| Subroutine | Description |
|---|---|
| S.IOCS31 | Mark units off-line |
| S.IOCS32 | Predevice access request validation |
| S.IOCS33 | Disc FAT processor |
| S.IOCS34 | Allocate variable IOQ entry |
| S.IOCS35 | Reserved |
| S.IOCS36 | File segment processor |
| S.IOCS37 | Update FAT and ART for disc I/O |
| S.IOCS38 | Reserved |
| S.IOCS39 | Reserved |
| S.IOCS40 | Build IOCDs for XIO transfers |

# SECTION 2 - ENTRY POINTS

## 2.1 Entry Point 1 - Open File

See M.FILE in the MPX-32 Reference Manual 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 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 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 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 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 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 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 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 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 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 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 desired or bit 0 is reset if no zeroing 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 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 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 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 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 for a detailed description of this entry point.

### 2.24 Entry Point 24 - Reserve Dual-ported Disc/Set Single-channel 8-line Mode

See the M.RESP or M_RESP system service in the MPX-32 Reference Manual 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 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 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 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 Start 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 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 0-11 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 | 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.

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 |
| | | ≠ 0 | Transfer address is within E-memory |
| | | 1 | Transfer address is in operating system portion of E-memory (virtual = absolute) |

| Value | Definition |
|---|---|
| 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 |
| 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 Zero

Registers: R0

| Value | Definition |
|---|---|
| 0 | Transfer address out of E-memory, or transfer crosses noncontiguous memory blocks |
| ≠ 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

## 2.41 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 | | |

# SECTION 3 - SUBROUTINES

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

This routine reestablishes 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.<br><br>Note: User exits through H.IOCS,34. |
| | (or) | | |
| | X1 | FCB address | |
| | BL | NORMAL-END-ACTION | No-wait, complete without error. Address from FCB.NWOK.<br><br>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.<br><br>Note: Post I/O processing is reinitiated 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<br><br>Note: Post I/O processing will be reinitiated when I/O completes |

Registers: None unless noted.

Abort Cases:

| | |
|---|---|
| IO06 | Blocking buffer error |
| IO21 | Unrecoverable I/O error |

Output Messages: *dtchsa INOP: R,A?

| | |
|---|---|
| dt | device type code (for example, LP) |
| ch | channel number |
| sa | subaddress |

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

| | |
|---|---|
| dt | device type code |
| 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.IOCS19, 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.IOCS19, 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.IOCS19, 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.IOCS19, 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 data address or transfer count |
| Output Messages: | None | |

### 3.15 Subroutine S.IOCS14 - Abort Code Formatting

This routine is used to determine what operation caused an abort and on what 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 four-character abort code |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | BU | S.IOCS16 |
| Registers: | R1 | FCB address |
| | R4 | Destroyed |
| | R5 | Abort code |
| | R6 | Four-character (ASCII) operation causing the abort |
| | R7 | One- to three-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 | Four-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<br>R2 - R7 | I/O queue address<br>Destroyed |
| Abort Cases: | None | |
| Output Messages: | None | |

### 3.20 Subroutine S.IOCS19 - Move Memory

This routine moves memory from one memory location to a different memory location. The move is performed using the largest transfer quantity possible, such as, doublewords, words, halfwords, or bytes. The transfer quantity used is dependent upon the alignment of the to and from address locations.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | BL | S.IOCS19 |
| 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 address after the from address (for example, entry registers R1 + R5) |
| | R2 | Contains the next available address after the to address (for example, entry registers R2 + R5) |
| | R3 | Unchanged |
| | R4 - R7 | Destroyed |
| Abort Cases: | None | |
| Output Messages: | None | |

### 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 S.IOCS17 | No-wait I/O requests |
| Registers: | R2-R7 | Destroyed |
| Abort Cases: | IO41 | Blocking buffer error |
| | IO44 | Nondevice access abort |

### 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'FFFFFF') |
| | 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 | Transfer address is not mapped |
| IO03 | Reading into a protected area |
| IO08 | Device assignment is required for unprivileged user |
| IO09 | Illegal operation to the SYC |
| IO28 | Illegal operation to an output active file |
| IO38 | Illegal operation on a read-only file |

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 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 disc files. It is called by the disc 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 disc position |
| | R6 | Zero if operation within file bounds;<br>-1 if operation caused an EOF, EOM, or BOM;<br>+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 disc 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 disc file. This routine updates the current disc 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

# INTERRUPT AND TRAP PROCESSORS (H.IP?? and H.SVC?)

## SECTION 1 - OVERVIEW

### 1.1 General Description

### 1.2 Interrupt Processors

| Interrupt processor | Description |
|---|---|
| H.IP13 | Console attention interrupt processor |
| H.IP0A | Call monitor interrupt processor |
| H.IPCL | Real-time clock interrupt processor |

### 1.3 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 |
| H.SVC0 | M.CALL SVC |
| H.SVC1 | 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 |

# SECTION 2 - INTERRUPT PROCESSORS

## 2.1 Console Attention Interrupt Processor (H.IP13)

The console interrupt processor is directly connected to the console interrupt and is thereby activated upon its occurrence. This processor has one primary function, that of recognizing a request for console services, with the console interrupt, and issuing a break request for the associated task.

Entry Conditions

| | |
|---|---|
| Calling Sequence: | The processor is entered directly upon the occurrence of the console interrupt (priority level 13). All registers are saved by this processor. |
| Exit Conditions | All registers are restored to their content at the time of interrupt. Exit is made to the system with the standard interrupt exit sequence (BL S.EXEC5). |
| Abort Cases: | None |
| Output Messages: | None |

External References

| | |
|---|---|
| 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 | CALM instruction was not located |
| | CM02 | Expected CALM instruction does not have CALM (X'30') opcode |
| | CM03 | Invalid CALM number |
| Output Messages: | None | |

### External References

| | |
|---|---|
| System Macros: | 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: (1) maintains the clock interrupt counter (C.INTC) which is used for time of day calculations and (2) 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<br>R6,R7 | Address of register save area<br>PSD |
| Abort Cases: | None | |
| Output Messages: | None | |

External References

| | |
|---|---|
| System Macro: | M.EQUS |

# SECTION 3 - TRAP PROCESSORS

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

Power fail trap processing performed at this level is considered to be dependent upon the installation and application. Therefore, MPX-32 provides a minimal routine 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 considered to be application and installation dependent and, 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 considered to be 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 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 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 |
| Output Messages: | None | |

### External References

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 | Nonpresent Memory Trap (all cases) |
| 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 | Unimplemented instruction |
| 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 |
| Registers: | R2 | Register save area address |
| | R5 | Abort code |
| | R6,R7 | PSD |
| Abort Cases: | PV01 | Privilege violation (all cases) |
| 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 at priority level X'06'

Exit Conditions

Return Sequence:

| SVC Type | Description |
|---|---|
| 0 | M.CALL processor |
| 1 | SVC Type '1' processor |
| 2 | SVC Type '2' processor |
| 3 | M.OPEN processor |
| 4 | M.RTRN/M.RTNA processor |
| 5-6 | Reserved |
| 7 | Communications software |
| 8 | Custom products |
| 9 | Diagnostics |
| A-D | Event trace |
| E | Available for customer use |
| F | 'CALM' replacement SVC |

## 3.8 M.CALL SVC Processor (H.SVC0)

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

Entry Conditions

| | | | |
|---|---|---|---|
| Calling Sequence: | The requestor must be privileged or must have the System Administrator attribute. Execution of the following code: | | |
| | M.CALL | mm,ee (SVC type '0' instruction) | |
| | mm | is the module number being called (bits 20-23 of SVC instruction) | |
| | ee | is the entry point or SRID | |

Exit Conditions

| | | | |
|---|---|---|---|
| Return Sequence: | LSPD | SVC0.T1 | SVC0.T1 = Address of the entry point within the called module |
| Registers: | All registers remain intact | | |
| Abort Cases: | SV01 | Unprivileged task attempted use of M.CALL | |

## 3.9 Supervisor Call Trap Processor (H.SVC1)

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 and entry point to enter. 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 | SVC0.T1 | SVC0.T1 = Address of MOD., E.P. |
| Registers: | Same as upon detection of trap signal. | | |
| Abort Cases: | SV02<br>SV03 | Invalid SVC number<br>Attempted use of privileged only service by unprivileged task | |
| 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 the control transfer mechanism for 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 5, 6, 7, 8, or E) |

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 References

| | |
|---|---|
| System Macros: | M.EQUS<br>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 References

System Macros: 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 | Map fault has occurred. Rerun task. |
| Output Messages: | None | |

External References

| | |
|---|---|
| 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 exception has occurred |
| Output Messages: | None | |

External References

| | |
|---|---|
| System Macros: | 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. |

### Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | BL | S.EXEC5A |
| Registers: | R2 | Register save area address |
| | R5 | Abort code |
| | R6,R7 | PSD |
| Abort Case: | HT01 | Privileged user halt has occurred |
| Output Messages: | None | |

### External References

| | |
|---|---|
| System Macros: | 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 32/27 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 has occurred |
| Output Messages: | None | |

External References

| | |
|---|---|
| System Macros: | M.EQUS |
| | M.TBLS |

# Rapid File Allocation (H.MDT)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# RAPID FILE ALLOCATION (H.MDT)

## SECTION 1 - OVERVIEW

### 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 |

# SECTION 2 - 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 an MDT Entry and Copy to a 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 register seven.

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) |
| Registers: | R7 | Zero or error status |

| | |
|---|---|
| 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 References

| | |
|---|---|
| System Subroutines: | S.VOMM28<br>S.VOMM29 |
| System Macro: | M.BWORD |

### 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: |

| | |
|---|---|
| 0 | Update |
| 1 | Create |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | M.RTRN | (CC1 set indicates error) |
| Register: | R7 | Zero or error status: |

| | |
|---|---|
| 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 References

| | |
|---|---|
| System Subroutines: | S.VOMM23<br>S.VOMM28<br>S.VOMM29 |
| System Macro: | M.BWORD |

### 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: |

| | |
|---|---|
| 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 References

System Subroutines: S.VOMM23
S.VOMM28
S.VOMM29

System Macro: M.BWORD

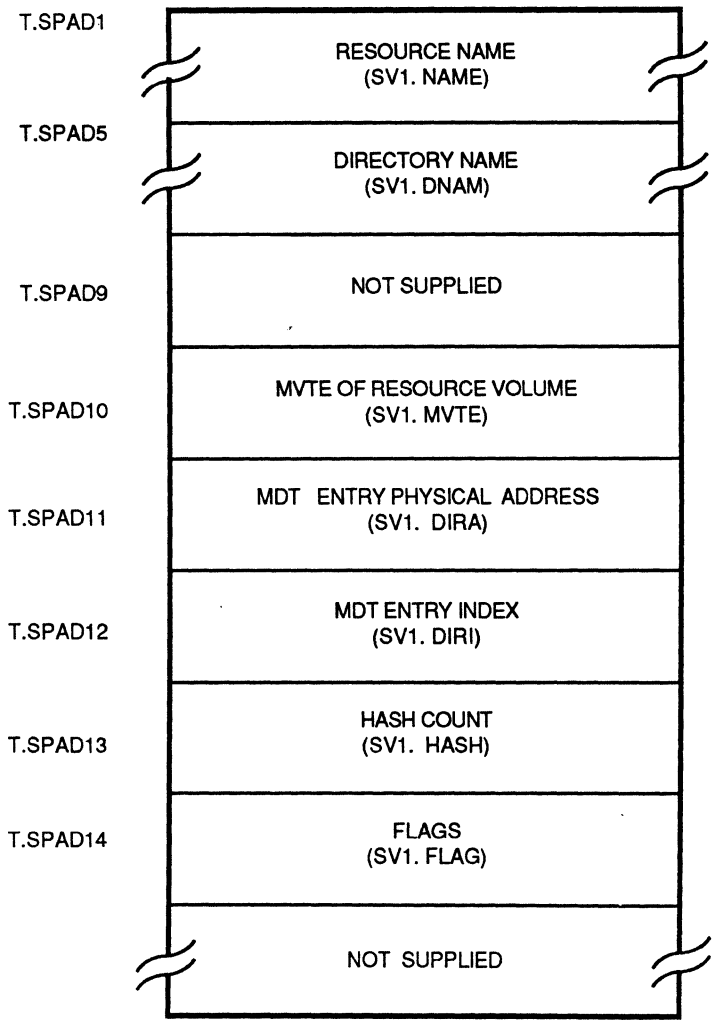
# SECTION 3 - SUBROUTINES

## 3.1 Subroutine S.MDT1 - Locate an MDT Entry and Copy Entry Information to the Scratchpad

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:




860451

## Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | BL | S.MDT1 |
| Registers: | R1 | Pathname vector or pathname block vector |
| | R7 | Mode of the call: |
| | | 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: |
| | | 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 References

| | |
|---|---|
| System Macros: | 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: |
| | | 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 References

| | |
|---|---|
| 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<br>T.SPAD12 | Directory address<br>Hash index |
| Abort Cases: | None | |
| Output Messages: | None | |

External References

| | |
|---|---|
| System Macro: | M.PUSH |

### 3.4 Subroutine S.MDT4 - 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<br>(or)<br>BU | SV1.XOK (Normal)<br><br>SV1.XXX (Error) |
| Registers: | R1-6<br>R7 | Unchanged<br>Unchanged or error status<br><br>7 Resource does not exist<br>8 Resource name in use<br>10 MDT entry unavailable |
| Scratchpad Usage: | T.SPAD11<br>T.SPAD12 | Directory address<br>Hash index |
| Abort Cases: | None | |
| Output Messages: | None | |

External References

| | |
|---|---|
| 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: |
| | | 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: |
| | | 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 References

| | |
|---|---|
| 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 References

| | |
|---|---|
| System Macro: | M.PUSH |

## 3.7 Subroutine S.MDT7 - Build a Pathname Block Vector in the MDT Resource Descriptor Buffer

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 References

| | |
|---|---|
| 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 References

| | |
|---|---|
| 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 References

| | |
|---|---|
| System Macro: | M.PUSH |

# Memory Management (H.MEMM)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# MEMORY MANAGEMENT (H.MEMM)

## SECTION 1 - 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 | N/A | Reserved |
| 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.MEMM3 | Allocate shared memory swap file |
| 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 |

# SECTION 2 - 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 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

Registers:

R1 Address of MIDL (halfword bounded)

R5 Right halfword is the number of map blocks required
Left halfword:

| Value | Memory Class |
|---|---|
| 1 | E |
| 2 | H |
| 3 | S |

R3 Halfword-bounded address of MEML

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) that are required inputs to this routine. 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.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | M.CALL | H.MEMM,2 |
| Registers: | R1 | Halfword-bounded address of MIDL |
| | R5 | Number of map blocks to deallocate |
| | R3 | Halfword-bounded address of MEML |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | M.RTRN | |
| Registers: | DQE.CME, DQE.CMH, and DQE.CMS are decremented for each swappable map. | |
| Scratchpad Usage: | T.SPAD1 | Used to build RID for resource deletion |

## 2.3 Entry Point 3 - Get Dynamic Extended Data Space

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

## 2.9 Entry Point 9 - Reserved

## 2.10 Entry Point 10 - Reserved

## 2.11 Entry Point 11 - Deallocate Memory Due to Swapping

This entry point is called only by 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 | Halfword-bounded address of MIDL |
| | R3 | Halfword-bounded 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 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 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.

# SECTION 3 - 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 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 | Eight character owner name (dynamic only) |
| | R6,R7 | First eight characters of partition or shared image name if the search parameter in register one is a pathname identifier; otherwise, registers not used. |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Registers: | R1 | Address of shared memory table entry or next available entry if not found |
| | R2 | Current stack pointer |
| | R3 | TSA address |
| | R4-R6 | Destroyed |
| | R7 | Error status; otherwise, register 7 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.MEMM3 - Allocate Shared Memory Swap File

This subroutine allocates temporary disc space for use as a swap file. The shared memory table is updated to reflect the space information for the swap file. If a swap file is successfully allocated, the swappable bit in the shared memory table is set.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | BL | S.MEMM3 |
| Registers: | R1 | Address of Shared Memory Table |
| | R7 | CNP address or zero |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Registers: | R2,R4-R6 | Destroyed |
| | R7 | Error status; otherwise, register seven is destroyed |
| Status: | CC1 set | |

| R7 | Value | Definition |
|---|---|---|
| | 8 | Unrecoverable I/O error to volume |
| | 38 | Timeout occurred waiting for space |
| | 52 | Volume space unavailable |

Note: Partition is not marked swappable if an error occurs.

Scratchpad Usage: None

### 3.4 Subroutine S.MEMM5 - Update Map Segment Descriptor for Memory Increase

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<br>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 | |

### 3.5 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) |

### 3.6 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 | Register 3 is copied to register 1 (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 |

### 3.7 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 destroyed | |

### 3.8 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 | |

### 3.9 Subroutine S.MEMM13 - Reserved

### 3.10 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 |

### 3.11 Subroutine S.MEMM15 - Reserved

### 3.12 Subroutine S.MEMM16 - Get System Buffer Space

This subroutine is used by system services requiring temporary buffer space. Logically 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, register seven is destroyed |
| Error Conditions: | 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 |

### 3.13 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

| | | |
|---|---|---|
| Registers: | All destroyed | |
| Error Conditions: | CC1 | Invalid system buffer deallocation; only those maps within the deallocation request range marked as system space are deallocated. |

### 3.14 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. Information pertinent to the allocation or deallocation of memory is returned as a result of the scan.

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,R7 | Destroyed |
| Status: | CC1 | Requested space not available |
| Scratchpad Usage: | None | |

### 3.15 Subroutine S.MEMM19 - Update Map Segment Descriptor for Memory Decrease

This subroutine updates the map segment page count in DQE.MSD after memory deallocation.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequences: | 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 | |

### 3.16 Subroutine S.MEMM20 - Check New Task Size

This subroutine is called whenever a memory allocation is about to take place. 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 Sequences: | TRSW | R0 |
| Registers: | R4 | Number of map blocks in excess of physical memory if CC1 is set; otherwise, register four is destroyed |
| Status: | CC1 | Task size exceeds total memory configured |

### 3.17 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 register one, on entry, is a path name 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 |

### 3.18 Subroutine S.MEMM22 - Get Specified Physical Map Block

This subroutine acquires physical map blocks for the operating system. The availability of the specified physical map block is checked. If available, the block is allocated. If not, 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 |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Registers: | All preserved (if CC1 not set) | |
| Status: | CC1 set | |

| 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

# Memory Pool Management (H.MEMM2)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# MEMORY POOL MANAGEMENT (H.MEMM2)

## SECTION 1 - 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.MEMM10 | Release memory pool buffer |

# SECTION 2 - 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, indiscriminate use of this subroutine should be avoided 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 register 3 is zero |
| Scratchpad Usage: | None | |

## 2.2 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 register 1 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 |

# System Services (H.MONS)

# MPX-32 Technical Manual

# Volume II

## CONTENTS

# SYSTEM SERVICES (H.MONS)

## SECTION 1 - 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 disc 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 |

* Implies that this service is available to privileged users only.

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

| Entry point | SVC number | Description |
|---|---|---|
| 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 |
| 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.

# SECTION 2 - ENTRY POINTS

## 2.1 Entry Point 1 - Physical Device Inquiry

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

External References

| | |
|---|---|
| System Service: | H.REMM,27 |
| System Subroutine: | S.REXS8 |
| System Macros: | M.CALL<br>M.RTRN |

## 2.2 Entry Point 2 - Permanent File Address Inquiry

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

External References

| | |
|---|---|
| System Service: | H.FISE,10 |
| System Macros: | M.CALL<br>M.RTRN |

## 2.3 Entry Point 3 - Memory Address Inquiry

See M.ADRS or M_ADRS in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

| | |
|---|---|
| System Macro: | M.RTRN |

## 2.4 Entry Point 4 - Create Timer Entry

See M.SETT or M_SETT in the MPX-32 Reference Manual 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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

## 2.6 Entry Point 6 - Delete Timer Entry

See M.DLTT or M_DLTT in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

## 2.12 Entry Point 12 - Memory Dump Request

See M.DUMP or M_DUMP in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macros: M.SPAD
M.CALL
M.RTRN

## 2.13 Entry Point 13 - Load Overlay Segment

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

External References

System Macros: M.CALL
M.RTRN

## 2.14 Entry Point 14 - Load and Execute Overlay Segment

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

External References

System Macros: M.CALL
M.RTRN

## 2.15 Entry Point 15 - Activate Task

See M.ACTV or M_ACTV in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macro: M.RTRN

## 2.16 Entry Point 16 - Resume Task Execution

See M.SUME or M_SUME in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.19 Entry Point 19 - Abort Specified Task

See M.BORT or M_BORT in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

| | |
|---|---|
| System Macros: | M.RTRN |
| | M.CALL |
| | M.IOFF |
| | M.ION |
| | M.OPEN |

### 2.20 Entry Point 20 - Abort Self

See M.BORT or M_BORT in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

| | |
|---|---|
| System Macro: | M.CALL |

### 2.21 Entry Point 21 - Allocate File or Peripheral Device

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

External References

| | |
|---|---|
| System Service: | H.ALOC,6 |
| System Subroutine: | S.REXS8 |
| System Macro: | M.CALL |

### 2.22 Entry Point 22 - Deallocate File or Peripheral Device

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

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.25 Entry Point 25 - Program Hold Request

See M.HOLD or M_HOLD in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

System Macro: M.RTNA

### 2.30 Entry Point 30 - Delete Interactive Debugger

See the H.REXS,30 entry point description in the MPX-32 Technical Manual Volume II 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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

System Macros: M.RTRN
M.RTNA

### 2.33 Entry Point 33 - Permanent File Log

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

External References

| | |
|---|---|
| System Service: | H.VOMM,10 |
| System Subroutine: | S.REXS8 |
| System Macros: | M.CALL<br>M.RTRN |

### 2.34 Entry Point 34 - Username Specification

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

External References

| | |
|---|---|
| System Macros: | M.CALL<br>M.RTRN |

### 2.35 Entry Point 35 - Get Message Parameters

See M.GMSGP or M_GMSGP in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

| | |
|---|---|
| System Macro: | M.RTRN |

### 2.36 Entry Point 36 - Get Run Parameters

See M.GRUNP or M_GRUNP in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

| | |
|---|---|
| System Macro: | M.RTRN |

### 2.37 Entry Point 37 - Wait for Any No-wait Operation Complete, Message Interrupt or Break Interrupt

See M.ANYW or M_ANYWAIT in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.40 Entry Point 40 - Parameter Task Activation

See M.PTSK or M_PTSK in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.41 Entry Point 41 - Get Address Limits

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

External References

System Macro: M.RTRN

### 2.42 Entry Point 42 - Debug Link Service

See the Debug Link system service in of the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.47 Entry Point 47 - Activate Task Interrupt

See M.INT or M_INT in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.49 Entry Point 49 - Exit Run Receiver

See M.XRUNR or M_XRUNR in the MPX-32 Reference Manual 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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

## 2.52 Entry Point 52 - RTM CALM Terminate Task Exectution

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' |
| | (or) | |
| | 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 References

| | |
|---|---|
| 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' |
| | (or) | |
| | 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 Codes: | None |
| Output Messages: | None |

External References

| | |
|---|---|
| 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' |
| | (or) | |
| | 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 References

| | |
|---|---|
| System Macros: | M.RTRN<br>M.IOFF<br>M.IONN<br>M.OPEN<br>M.CALL |

## 2.55 Entry Point 55 - RTM CALM Allocate File or Device

This entry point dynamically allocates a peripheral device, a permanent disc file, a temporary disc file, or a SLO or SBO file, and creates a File Assignment Table (FAT) entry for the allocated unit and specified logical file code. This entry point may also be used to equate a new logical file code with an existing logical file code.

Entry Conditions

| | | | |
|---|---|---|---|
| Calling Sequence: | CALM | X'40' | |
| | (or) | | |
| | M.CALL | H.MONS,55 | |
| Registers: | R1 | Denial return address | |
| | R5 | byte 0 | Function code as follows: |
| | | | 1 = Assign file code to a user or system permanent file |
| | | | 2 = Assign file code to a system file code |
| | | | 3 = Assign file code to a peripheral device |
| | | | 4 = Assign file code to a defined file code |
| | | 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 |
| Registers: | None | |
| | (or) | |
| Return Sequence: | M.RTNA 1 | For denial returns if the requested file or device cannot be allocated. |
| Registers: | None | |
| | (or) | |
| Return Sequence: | 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' |
| | (or) | |
| | M.CALL | H.MONS,56 |
| Registers: | R5 | Three 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 |
| | (or) | |
| Return Sequence: | M.RTRN | 5,6,7 |
| Registers: | | |

| | | |
|---|---|---|
| | R5 | Disc = number of 192-word blocks in file<br>Magnetic tape = reel identifier<br>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 |

| | | |
|---|---|---|
| | bits 24-31 | Disc = number of 192-word blocks per allocation unit<br>Magnetic tape = volume number (0 if single volume)<br>All other devices = 0 |

Note: If the file is a 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 References

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 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 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' |
| | (or) | |
| | 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 |

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 eight-word area where the 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 space. |
| Output Messages: | None | |

External References

| | |
|---|---|
| System Macros: | M.CALL<br>M.RTRN |

### 2.62 Entry Point 62 - Resourcemark Lock

See M.RSML or M_RSML in the MPX-32 Reference Manual 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 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 for a detailed description of this entry point.

External References

| | |
|---|---|
| System Macros: | M.RTRN<br>M.IOFF<br>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 for a detailed description of this entry point.

External References

| | |
|---|---|
| System Service: | H.REXS,4 |
| System Macros: | M.CALL<br>M.RTRN |

### 2.67 Entry Point 67 - Set Synchronous Task Interrupt

See M.SYNCH or M_SYNCH in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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

# MULTIVOLUME MAGNETIC TAPE (H.MVMT)

## SECTION 1 - OVERVIEW

### 1.1 General Information

The Multivolume Magnetic Tape Module (H.MVMT) performs all data management operations pertaining to 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 release 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

# SECTION 2 - ENTRY POINTS

## 2.1 Entry Point H.MVOP - Predevice Access Processing

This entry point performs predevice access processing on behalf of multivolume magnetic tape requests.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | BL | H.MVOP |
| Register: | R1 | FCB address |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Register: | 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 |

# Resource Management (H.REMM)

# MPX-32 Technical Manual

# Volume II

# CONTENTS

# RESOURCE MANAGEMENT (H.REMM)

## SECTION 1 - 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. Mechanisms for coordinating concurrent access to shared resources are also provided by this function. 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.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 |
| 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 | Enqueue 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 |

## 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.REMM15 | S.MEMM3 | Allocate shared memory swap file |
| S.REMM17 | S.MEMM5 | Update map segment descriptor for memory increase |
| S.REMM19 | S.MEMM7 | Remap 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 |

# SECTION 2 - 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 in the MPX-32 Reference Manual.

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 available |
| | | 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 dynamic 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 |

| Error Code | Definition |
|---|---|
| 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 in the MPX-32 Reference Manual.

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 for a detailed description of this entry point.

## 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 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 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 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 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 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 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 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 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.

# SECTION 3 - SUBROUTINES

## 3.1 Subroutine S.REMM4 - Issue Dismount Request

This subroutine issues a dismount request to the operators console with a no-wait run request to J.MOUNT. 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<br>R3 | Destroyed<br>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 |
| | | 254 | Hold request |
| | | 255 | Abort 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 and all mount assignments in the Volume Assignment Table (VAT).

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.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 register four. CC1 is set to indicate an unsuccessful scan, and the resulting status is returned in register seven.

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, register two is unchanged |
| | R4 | Device mask |
| | R5 | Destroyed |
| | R6 | Device specification |
| | R7 | Status if an error; otherwise, register seven 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 | |

### 3.5 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 | |

### 3.6 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

Assumptions: Context switching is inhibited

Calling Sequence: BL S.REMM9

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, register seven is destroyed |
| Status: | CC1 set | Error (status in register seven) |
| | CC2 set | Resource allocated: |

| Error Code | Definition |
|---|---|
| 55 | ART is full |

| | | |
|---|---|---|
| Scratchpad Usage: | T.SPAD22 | Used for temporary storage |

### 3.7 Subroutine S.REMM10 - Locate FPT/FAT/SMT With Preprocessing

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, register five is unchanged (volume resource or partition) |
| | R7 | Status if an error; otherwise, register seven 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 | |

### 3.8 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) |

### 3.9 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 eight 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, register five is unchanged | |
| | R7 | Status if an error; otherwise, register seven is unchanged | |
| Status: | CC1 set | Error Code | Definition |
| | | 29 | LFC not assigned |
| | | 30 | Invalid allocation index |
| Scratchpad Usage: | None | | |

## 3.10 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, register seven is destroyed |

| Status: | CC1 set | Error Code | Definition |
|---|---|---|---|
| | | 3 | FPT/FAT space unavailable |
| | | 17 | Duplicate LFC assignment |

Scratchpad Usage: None

### 3.11 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, register seven is destroyed |
| Status: | CC1 set | Volume is not assigned (error code is 20) |
| Scratchpad Usage: | None | |

### 3.12 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.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | BL | S.REMM24 |
| Registers: | R6,R7 | 6-bit ASCII coded name right-justified in lower 48 bits of R6,R7 |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Registers: | R2-R5 | Destroyed |
| | R6,R7 | 8-bit ASCII coded name |

### 3.13 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 |

### 3.14 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 |

### 3.15 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 |

### 3.16 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

| | | |
|---|---|---|
| Assumptions: | Context switching is inhibited | |
| Calling Sequence: | BL | S.REMM36 |
| 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, register seven is destroyed | |
| Status: | CC1 set | Error Code | Definition |
| | | 38 | Time out occurred during enqueue |
| | | 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 | |

## 3.17 Subroutine S.REMM37 - Caller Notification Packet M.RTRN Processing

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 with CNP: | M.RTRN<br>(or)<br>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 | |

### 3.18 Subroutine S.REMM38 - Enqueue 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

| | | |
|---|---|---|
| Assumptions: | Context switching inhibited | |
| Calling Sequence: | BL | S.REMM38 |
| 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 | Enqueuing for |
|---|---|
| Zero | System table space |
| ART address | Volume resource |
| MVTE address | Volume mount |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| | Context Switching Enabled | |
| Registers: | R3,R4,R7 | Destroyed |
| Status: | CC1 set | Returning due to enqueue time out (status code posted in R7) |
| | CC2 set | CNP indicated no enqueue |
| Scratchpad Usage: | None | |

### 3.19 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

Context Switching Inhibited

| | | | |
|---|---|---|---|
| Return Sequence: | TRSW | R0 | |
| Registers: | R4 | Destroyed | |
| | R7 | Status, if error; otherwise, register seven 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 | |

### 3.20 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

Context Switching Enabled

| | | | |
|---|---|---|---|
| Return Sequence: | TRSW | R0 | |
| Registers: | R4 | Destroyed | |
| | R7 | Status, if error; otherwise, register seven 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 | |

### 3.21 Subroutine S.REMM44 - Check for 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 requesting task |

### 3.22 Subroutine S.REMM45 Deallocate Blocking Buffers from the TSA

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 |

### 3.23 Subroutine S.REMM46 - Allocate a Blocking Buffer Head Cell from the TSA

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: | None | |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Registers: | R3 | Head cell address is available; otherwise, register three is destroyed |
| | R4 | Destroyed |
| Status: | CC1 set | No free head cells are found |

# Resident Executive Services (H.REXS)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

## 3 - SUBROUTINES

# RESIDENT EXECUTIVE SERVICES (H.REXS)

## SECTION 1 - 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 | 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 |
| H.REXS,63 | 1A | Resourcemark unlock |

* 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,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<br>Acquire current date/time in binary format<br>Acquire current date/time in byte binary format<br>Acquire system date/time in any format<br>Get current date/time |
| H.REXS,75 | 51**** | Convert ASCII date/time to byte binary format<br>Convert ASCII date/time to standard binary<br>Convert binary date/time to ASCII format<br>Convert binary date/time to byte binary<br>Convert byte binary date/time to ASCII<br>Convert byte binary date/time to binary<br>Convert system date/time format<br>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 | | Reserved |
| 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 | | Reserved |
| 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 |
| 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 |

# SECTION 2 - 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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

## 2.4 Entry Point 4 - Create Timer Entry

See M.SETT or M_SETT in the MPX-32 Reference Manual 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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

## 2.6 Entry Point 6 - Delete Timer Entry

See M.DLTT or M_DLTT in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

## 2.12 Entry Point 12 - Memory Dump Request

See M.DUMP or M_DUMP in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macros: M.SPAD
M.CALL
M.RTRN

## 2.13 Entry Point 13 - Load Overlay Segment

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

External References

System Macros: M.CALL
M.RTRN

## 2.14 Entry Point 14 - Load and Execute Overlay Segment

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

External References

System Macros: M.CALL
M.RTRN

## 2.15 Entry Point 15 - Activate Task

See M.ACTV or M_ACTV in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macro: M.RTRN

## 2.16 Entry Point 16 - Resume Task Execution

See M.SUME or M_SUME in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

## 2.19 Entry Point 19 - Abort Specified Task

See M.BORT or M_BORT in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

System Macros: M.CALL

## 2.21 Entry Point 21 - Assign and Allocate Resource

See M.ASSN or M_ASSIGN in the MPX-32 Reference Manual 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 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 for a detailed description of this entry point.

External References

System Macros: M.RTRN

## 2.24 Entry Point 24 - Task Option Word Inquiry

See M.PGOW or M_OPTIONWORD in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macro: M.RTRN

## 2.25 Entry Point 25 - Program Hold Request

See M.HOLD or M_HOLD in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macros: 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 for a detailed description of this entry point.

External References

| | |
|---|---|
| System Services: | H.VOMM,13<br>H.REXS,22<br>H.REXS,45<br>H.IOCS,23 |
| System Subroutines: | S.REXS8<br>S.VOMM23<br>S.REMM12<br>S.REMM37 |
| System Macros: | M.CALL<br>M.RTRN |

## 2.28 Entry Point 28 - Abort with Extended Message

See M.BORT or M_BORT in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

| | |
|---|---|
| System Macros: | M.RTRN<br>M.CALL<br>M.IOFF<br>M.IONN<br>M.OPEN |

## 2.29 Entry Point 29 - Load and Execute Interactive Debugger

See M.DEBUG or M_DEBUG in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

| | |
|---|---|
| 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 for a detailed description of this entry point.

External References

| | |
|---|---|
| System Macros: | M.RTRN<br>M.CALL<br>M.IOFF<br>M.IONN<br>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 for a detailed description of this entry point.

External References

| | |
|---|---|
| System Macros: | M.RTRN<br>M.RTNA |

## 2.33 Entry Point 33 - Validate Address Range

See M.VADDR or M_VADDR in the MPX-32 Reference Manual 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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.36 Entry Point 36 - Get Run Parameters

See M.GRUNP or M_GRUNP in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.37 Entry Point 37 - Wait for Any No-wait Operation Complete, Message Interrupt, or Break Interrupt

See M.ANYW or M_ANYWAIT in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.40 Entry Point 40 - Parameter Task Activation

See M.PTSK or M_PTSK in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.41 Entry Point 41 - Get Address Limits

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

External References

System Macro: M.RTRN

### 2.42 Entry Point 42 - Debug Link Service

See the Debug Link system service in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.47 Entry Point 47 - Activate Task Interrupt

See M.INT or M_INT in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.49 Entry Point 49 - Exit Run Receiver

See M.XRUNR or M_XRUNR in the MPX-32 Reference Manual 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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macros: 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 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 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 eight word boundary and returns the memory contents of that eight word block to the callers eight word buffer area, which must be on an eight 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 eight 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 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 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 References

| | |
|---|---|
| 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 for a detailed description of this entry point.

External References

| | |
|---|---|
| System Macros: | M.RTRN<br>M.IOFF<br>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 for a detailed description of this entry point.

External References

| | |
|---|---|
| System Services: | H.REXS,4 |
| System Macros: | M.CALL<br>M.RTRN |

### 2.67 Entry Point 67 - Set Synchronous Task Interrupt

See M.SYNCH or M_SYNCH in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

| | |
|---|---|
| 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 for a detailed description of this entry point.

External References

| | |
|---|---|
| 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 for a detailed description of this entry.

External References

| | |
|---|---|
| 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 for a detailed description of this entry point.

External References

| | |
|---|---|
| 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 for a detailed description of this entry point.

External References

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 for a detailed description of this entry point.

External References

System Macro: M.RTRN

### 2.74 Entry Point 74 - Acquire Date/Time Services

See the MPX-32 Reference Manual 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 References

System Macro: M.RTRN

### 2.75 Entry Point 75 - Conversion Services

See the MPX-32 Reference Manual 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 References

System Macro: M.RTRN

### 2.76 Entry Point 76 - Reformat RRS Entry

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

### 2.80 Entry Point 80 - Reserved

### 2.81 Entry Point 81 - Set Exception Return Address

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

### 2.87 Entry Point 87 - Reserved

### 2.88 Entry Point 88 - Get Command Line

See M.CMD or M_CMD in the MPX-32 Reference Manual 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 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 for a detailed description of this entry point.

### 2.91 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.

# SECTION 3 - 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 register three and its task number in register seven. 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 | one to eight 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 | One to eight 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 | |
| | 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 register seven.

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 zero through nine which are the register save area and the return PSD, are then zeroed. All condition codes located in scratchpad word eight, 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 six and seven 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 eight-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 disc file associated with the specified LFC is read only or read/write restricted.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | BL | S.REXS10 |
| Registers: | R5 | Bytes one through three contain the left-justified, blank-filled, one to three ASCII character LFC |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Registers: | R3 | FAT address or zero if an error |
| | R4,R7 | Destroyed |
| | R5 | Bit zero set if read only<br>Bytes one through three unchanged if LFC found; otherwise, register five is zero |

### 3.11 Subroutine S.REXS11 - Create System Pathname in Word 24

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 six and seven 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 eight-word pathname | |

# Task Management (H.TAMM)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# TASK MANAGEMENT (H.TAMM)

## SECTION 1 - 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.TAMM7 | Set VOMM stack and buffers FPT and FAT in the TSA |

# SECTION 2 - 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) spad 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 zero of word zero 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.

Special Case: 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 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 contains 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<br>Bit 0 not set: checksum all sections |
| | R2 | Preamble address |

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 |

# SECTION 3 - 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 Condition

| | | |
|---|---|---|
| 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, register seven is destroyed |
| Status: | CC1 set | |

| R7 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 | |
| | 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 |

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 information for H.TAMM,1 |

### 3.4 Subroutine S.TAMM7 - Set VOMM Stack and Buffers FPT and FAT in the TSA

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 |
| Normal Return: | R1,R2,R4,R6 | Destroyed |
| | R3 | TSA address (C.REGS) |

# Terminal Services (H.TSM)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# TERMINAL SERVICES (H.TSM)

## SECTION 1 - OVERVIEW

### 1.1 General Information

The Terminal Services Module (H.TSM) provides services for interfacing TSM terminals to MPX-32.

### 1.2 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,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 Subroutine Summary

| Subroutine | Description |
|---|---|
| S.TSM01 | Format abort message |
| S.TSM02 | Convert binary to ASCII decimal |
| S.TSM03 | Report task completion/suspension |
| S.TSM04 | Enqueue real-time SLO/SBO |
| S.TSM05 | Build MRRQ and link to J.TSM with no current task |

# SECTION 2 - 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 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 References

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 three-character task abbreviation

## 2.2 Entry Point 2 - Syntax Scanner

See M.TSCAN in the MPX-32 Reference Manual 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 on-line 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 on-line 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 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 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 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 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 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 unchanged |

### 2.13 Entry Point 13 - Set Lower Option

See M.SOPL or M_SOPL in the MPX-32 Reference Manual 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 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 for a detailed description of this entry point.

### 2.16 Entry Point 99 - SYSGEN Initialization

There is no SYSGEN initialization required. Return is made with TRSW through register zero.

# SECTION 3 - 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 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 destroyed | |

### 3.4 Subroutine S.TSM04 - Enqueue 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 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 destroyed | |

### 3.5 Subroutine S.TSM05 - Build MRRQ and Link to J.TSM with No Current Task

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 Volume I, Chapter 2.

Entry Conditions

| | | |
|---|---|---|
| Calling Sequence: | LA<br>BL | R1,addr<br>S.TSM05 |
| | addr | is the address of a message buffer containing the character count in byte zero |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Abort Cases: | CC1 set if message not linked | |

# Volume Management Module (H.VOMM)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# VOLUME MANAGEMENT MODULE (H.VOMM)

## SECTION 1 - OVERVIEW

### 1.1 General Information

The Volume Management Module (H.VOMM) is responsible for the dynamic manipulation of the disc data structures in MPX-32. The basic disc volume layout is defined by the Volume Formatter (J.VFMT) when a disc volume is initialized; this layout is verified prior to access by the mount program (J.MOUNT).

H.VOMM provides the following facilities:

- Disc 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 disc-resident structures (file space allocation maps, directories and descriptors). The MVT is set up at volume mount time; the disc-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.

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 Release 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 |
| 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 |

# SECTION 2 - ENTRY POINTS

## 2.1 Entry Point 1 - Create Permanent File

See M.CPERM or M_CREATEP in the MPX-32 Reference Manual for a detailed description 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 for a detailed description 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 for a detailed description 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 for a detailed description 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 for a detailed description 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 for a detailed description 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 for a detailed description 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 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 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 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 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 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 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 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 for a detailed description of this entry point.

External References

| | |
|---|---|
| Scratchpad Usage: | T.SPAD2<br>T.SPAD3 |

## 2.16 Entry Point 16 - Reconstruct Pathname

See M.PNAM or M_CONSTRUCTPATH in the MPX-32 Reference Manual for a detailed description of this entry point.

External References

| | |
|---|---|
| System Service: | H.REMM,7 |
| Scratchpad Usage: | T.SPAD1<br>T.SPAD2<br>T.SPAD3<br>T.SPAD4 |

## 2.17 Entry Point 17 - Allocate Resource Descriptor

See the Allocate Resource Descriptor system service in the MPX-32 Reference Manual 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 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 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 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 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 disc address |
| | (or) | |
| | Noncontiguous creation: | |
| | R4 | Number of segments created |
| | R5 | End of medium block number created |
| | R6 | MVTE |
| | R7 | RD disc address |
| | CC1 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 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 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 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.

# SECTION 3 - 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. Register one contains the address of a PN or PNB to be parsed. Register seven 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.DIRA | T.SPAD11 | Directory address |
| 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<br>Input Registers:<br>R4 — 0 for scan or delete, or create flag for create<br>R5 — 0 for scan (first scan only)<br>1 for create (second scan only)<br>-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 Original 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 for Modification

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 register four 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 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 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 | SMAP read error |
| 17 | SMAP write error |

### 3.15 Subroutine S.VOMM14 - Read Resource Descriptor by Resource Specification

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:<br>negative indicates delete file<br>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 |
| Register: | R7 | Resource type code |

Exit Conditions

| | | |
|---|---|---|
| Return Sequence: | TRSW | R0 |
| Registers: | None | |

### 3.19 Subroutine S.VOMM18 - Zero Resource Space if Requested

This subroutine zeroes the disc 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 |
| Register: | 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 unchanged | |

### 3.22 Subroutine S.VOMM21 - Reserved

### 3.23 Subroutine S.VOMM22 - Reserved

### 3.24 Subroutine S.VOMM23 - Determine Resource Specification Type

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<br>BL | S.VOMM24<br>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 zero 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

Registers:

| | |
|---|---|
| R7 (or) CC1 set | Unchanged if no abnormal conditions<br>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 register zero, 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 |

# Extended I/O Disc Handler (H.DCXIO)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

## 3 - USAGE



### Figures



### Tables

# EXTENDED I/O DISC HANDLER (H.DCXIO)

## SECTION 1 - OVERVIEW

### 1.1 General Information

The Extended I/O Disc Handler (H.DCXIO) is a software component of MPX-32 intended to provide support for Extended I/O Disc Processors (XIO), Input/Output Disc Processors (IOP), and Floppy Disc Controllers (IOP) connected to an MPX-based CONCEPT/32 computer.

The product is designed to support any number and mix of extended I/O disc drives listed below. These include fixed-head discs (FHD), moving-head discs (MHD), cartridge module drives (CMD), fixed module drives (FMD), minimodule drives (Winchester) (MMD) and floppy discs.

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 Discs Supported

The IOP disc controller and XIO disc processor Phase I support the single-ported versions of the following discs. The XIO disc processor Phase II supports both single and multiported versions of the following discs:

| Manufacturer | Gould 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 disc
FHD - Captive media, moving-head disc
CMD - Cartridge module drive, removable and captive media
FMD - Fixed module drive
MMD - Minimodule drive (Winchester)

Moving head discs (MHD) and the 600MB fixed module drive are available in both single and multiport versions.

The 5 megabyte Fixed Head Disc (FHD) is by definition a fixed-head device with 256 fixed heads. However, software must treat this unit as a moving-head disc with 64 cylinders and 4 heads. The fixed head disc 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 disc processor and IOP disc 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 disc processor and IOP disc are capable of supporting both track formats, only the F20 format is supported by MPX-32.

## 1.4 Dual Subchannel I/O

The disc processor and IOP disc 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. See Section 3.10.

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 Disc Processor Phase II

Multiporting allows CPUs to share a single disc drive. In order to maintain disc 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.

### Implicit Device Reservation

Implicit device reservation is a disc processor function and is transparent to the operating system. If a drive is multiported, the disc processor automatically issues a reserve command to the device before initiating an I/O request. Once the I/O is complete, the disc 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.

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 disc 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 disc processor release mentioned above. Also, the channel 1 and channel 2 inhibit switches located on the disc 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 disc is reserved. If this happens, the shared disc 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 disc 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 disc 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 disc drives can be connected to any IOP disc controller. The number of disc controllers configured per system is limited by the number of IOP channels and mechanical restrictions.

Up to eight disc drives can be connected to any disc processor. The number of disc processors configured per system is limited by the number of channels and cabinet space available.

# SECTION 2 - 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:

| Word | 0 7 | 8 15 | 16 31 |
|---|---|---|---|
| 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 disc processor.

## 2.2 Initialize Channel (INCH)

The Initialize Channel (INCH) command relays disc drive information to the disc processor, declares a buffer area, and makes the declared buffer area available to the disc processor. INCH must be the first I/O command to any channel that has a disc processor configured. INCH is performed automatically by the disc handler.

The data address specified in the INCH IOCD points to a nine word buffer that must begin on a word boundary. The first word of this nine-word buffer must contain a 24-bit address that points to a file-bounded 224 word buffer. This buffer is used by the disc processor for record keeping. The remaining eight words contain disc drive information for each configured drive. See Figure 2-1.

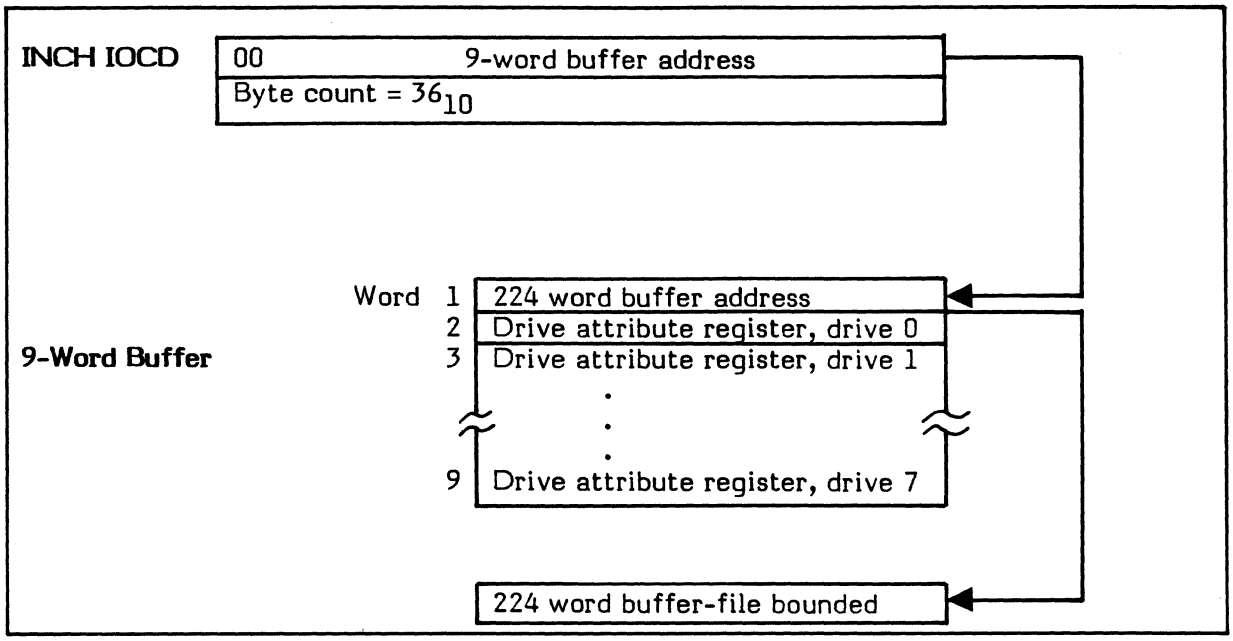



**Figure 2-1. Initialize Channel I/O Command Doubleword and Buffer**

Note: The above description applies only to the XIO disc processor. The IOP disc processor and floppy disc are handled differently.

**Drive Attribute Register** layout is:

| 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:

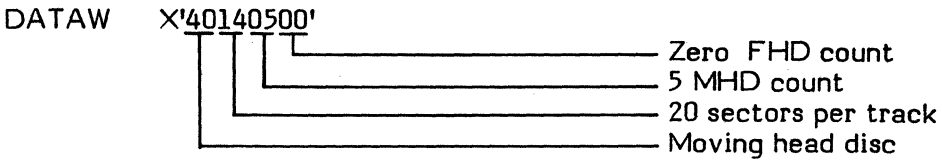
1.

| Bit | Meaning |
|---|---|
| 0 | 10 = FHD |
| 1 | 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 disc processor only) |
| 6-7 | Zero (Reserved for future use) |

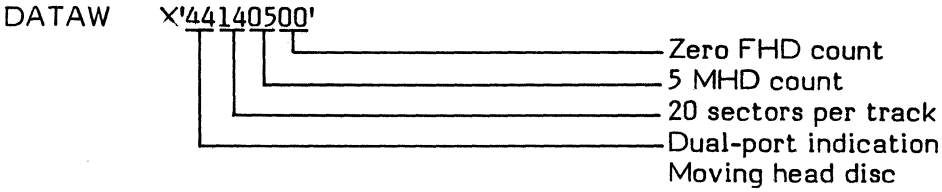
2. Sector Count is the number of sectors per track (20 decimal).
3. MHD Count is the number of heads on the MHD or the number of heads for the removable media portion of the cartridge module drive.
4. FHD Count is the number of heads for the captive media portion of the cartridge module drive.

**Generation of Drive Attribute Registers**

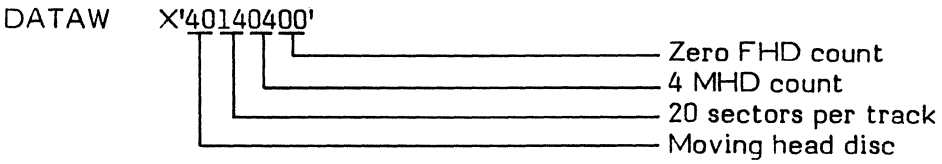
80 MB single-port moving head disc:



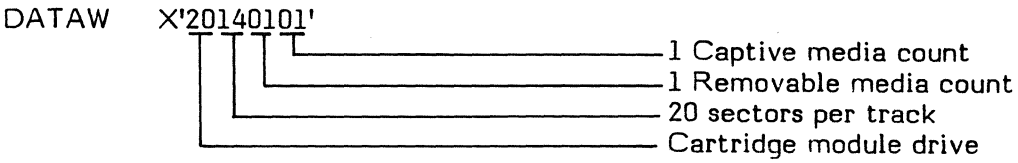

80 MB dual-port moving head disc:



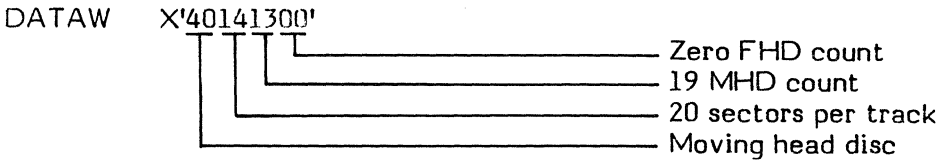

5 MB single-port fixed head disc:



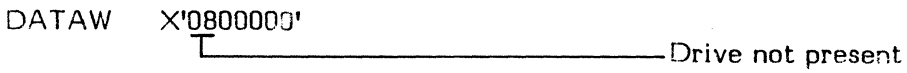

32 MB cartridge module drive:




300 MB single port moving head disc:




Drive that is not present:



## 2.3 Initialize Controller (INC)

The Initialize Controller (INC) command allows the controller initialization information to be passed to the IOP disc 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 (SENSE) command retrieves the results 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 followng diagram shows the information returned from a sense command. This does not apply to the floppy disc. See the Line Printer/Floppy Disc Controller Technical Manual for sense information.

**SENSE Information**

| Word | 0 7 | 8 15 | 16 23 | 24 31 |
|---|---|---|---|---|
| 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 discs 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 discs are zero. |

When all mode bits are set to the zero state, data operations occur between main memory and a Moving Head Disc (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 | Disc 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 | Disc 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 discs.

3. Drive Attribute Register. See Section 2.2.

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 | Disc 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 discs.

### 2.5 Transfer in Channel (TIC)

The Transfer in Channel (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 Write Data (WD) command is a user write request that transfers data to the disc from the address specified in the IOCD.

### 2.7 Write Sector Label (WSL)

The Write Sector Label (WSL) command writes sector labels to the disc. WSL is not currently used by the disc handler.

### 2.8 Write Track Label (WTL)

The Write Track Label (WTL) command writes track labels to the disc. WTL is not currently used by the disc handler.

### 2.9 Read Data (RD)

The Read Data (RD) command transfers data from the disc to the address specified in the IOCD. RD is used in the user read request.

### 2.10 Read Sector Label (RSL)

The Read Sector Label (RSL) command reads sector labels from the disc. RSL is not currently used by the disc handler.

### 2.11 Read Track Label (RTL)

The Read Track Label (RTL) command reads track labels from the disc. RTL is not currently used by the disc handler.

### 2.12 Read Angular Position (RAP)

The Read Angular Position (RAP) command reads the sector pulse counter from the disc. RAP is not currently used by the disc handler.

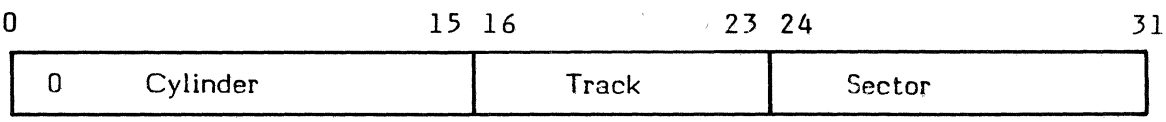
### 2.13 No Operation (NOP)

The No Operation (NOP) command is a nondata transfer command that executes without an associated disc 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 Seek Cylinder (SKC) command causes a disc 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 Format for No Skip (FNSK) command is used to format a disc. FNSK is not currently used by the disc handler.

### 2.16 Lock Protect Label (LPL)

The Lock Protect Label (LPL) command involves write lock. LPL is not currently used by the disc handler.

### 2.17 Load Mode Register (LMR)

The Load Mode Register (LMR) command identifies a byte of information that specifies the manner in which I/O is to take place with the disc. 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, word 2, byte 0 of the sense information, for interpretation of the mode bits. The disc handler automatically generates LMR as the first IOCD presented for disc access user requests. LMR physically resides in the IOQ.

### 2.18 Reserve (RES)

The Reserve (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 disc controller.

### 2.19 Release (REL)

The Release (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 disc controller.

### 2.20 Rezero (XEZ)

The Rezero (XEZ) command is a recalibration request to the disc 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 Test Star (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 disc handler.

## 2.22 Increment Head Address (IHA)

The Increment Head Address (IHA) command selects sector zero of the next sequential track in the associated disc drive. IHA is not currently used by the disc handler.

## 2.23 Priority Override (POR)

The Priority Override (POR) command provides a mechanism for overriding and disabling dual-ported disc 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 disc handler.

## 2.24 Set Reserve Track Mode (SRM)

The Set Reserve Track Mode (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 disc handler.

## 2.25 Reset Reserve Track Mode (XRM)

The Reser Reserve Track Mode (XRM) command makes all data areas designated as reserve tracks unavailable for write operations. XRM is not currently used by the disc handler.

## 2.26 Read ECC (REC)

The Read ECC (REC) command causes the disc processor/IOP disc to compute and present error correction information needed to recover from a disc read error. The information returned to the address specified in the REC IOCD contains:

| 0 — 15 | 16 — 31 |
|---|---|
| Displacement. See Note 1. | Correction Mask. See Note 2. |

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.

# SECTION 3 - 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. | R. See Note 2. | Instruction code | Aug 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 R to form the logical channel and subaddress. If R 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 disc handler varies depending on the instruction issued.

**SIO Instructions**

The following are condition code checking and disc 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 |

### HIO Instructions

| Hexadecimal instruction code | Meaning | Action |
|---|---|---|
| 4 | Status stored | Branch to XIO.SUB and process as though an interrupt had occurred |

No other condition codes are checked.

### ACI, DACI, RSCTL, RSCHNL, DCI, and ECI instructions

No condition codes are checked.

## 3.3 XIO CPU Instructions

| Hexadecimal instruction code | 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) |

### Start I/O (SIO)

The Start I/O (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.

### Test I/O (TIO)

The 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.

### Halt I/O (HIO)

The Halt I/O (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.

### Halt Channel (HCHNL) and Reset Channel (RSCHNL)

The Halt Channel (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. The RSCHNL instruction is used by the initialization entry point of H.IFXIO.

### Stop I/O (STPIO)

The Stop I/O (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 disc handler.

### Reset Controller (RSCTL)

The Reset Controller (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.

### Enable Channel Interrupt (ECI)

The Enable Channel Interrupt (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.

### Disable Channel Interrupt (DCI)

The Disable Channel Interrupt (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.

Activate Channel Interrupt (ACI)

The Activate Channel Interrupt (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.

Deactivate Channel Interrupt (DACI)

The Deactivate Channel Interrupt (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 disc 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)

Device Context Area (DCA)

A 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 the MPX-32 Technical Manual, Volume I, Chapter 2 for a description of the common section.

**Table 3-1. XIO Device-dependent Disc Information**

| Word | Hex Byte | 0 — 15 16 — 31 |
|---|---|---|
| 36<br>37 | 90 | Rezero IOCD used for error retry (DCA.REZO) |
| 38<br>39 | 98 | TIC IOCD used with rezero (DCA.TIC) |
| 40<br>41 | A0 | Load mode IOCD prototype (DCA.LMOD) |
| 42<br>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 for ECC (DCA.BITD) \| 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 disc (DCA.SCYL) |
| 57 | E4 | EOF buffer for nondata transfer command (DCA.EOFB) |
| 58 | E8 | Address of initialize controller routine (DCA.INCA) |
| 59<br>60 | EC | Not used (DCA.UNU1/DCA.UNU2) |
| 61 | F4 | Number of uncorrectable I/O errors this device (DCA.UREC) |
| 62<br>63 | F8 | NOP IOCD for error retry (DCA.NOP) |
| 64 | 100 | NOP TIC IOCD for error retry (DCA.NOPT) |

### 3.4.1 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:

| 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 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.

### 3.4.2 Input/Output Control Doubleword (IOCD)

See Section 2.1 for information on the Input/Output Control Doubleword (IOCD).

### 3.4.3 Sense Buffer

See Section 2.4 for information on the sense buffer.

### 3.4.4 INCH Buffer

See Section 2.2 for information on the INCH buffer.

### 3.4.5 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.

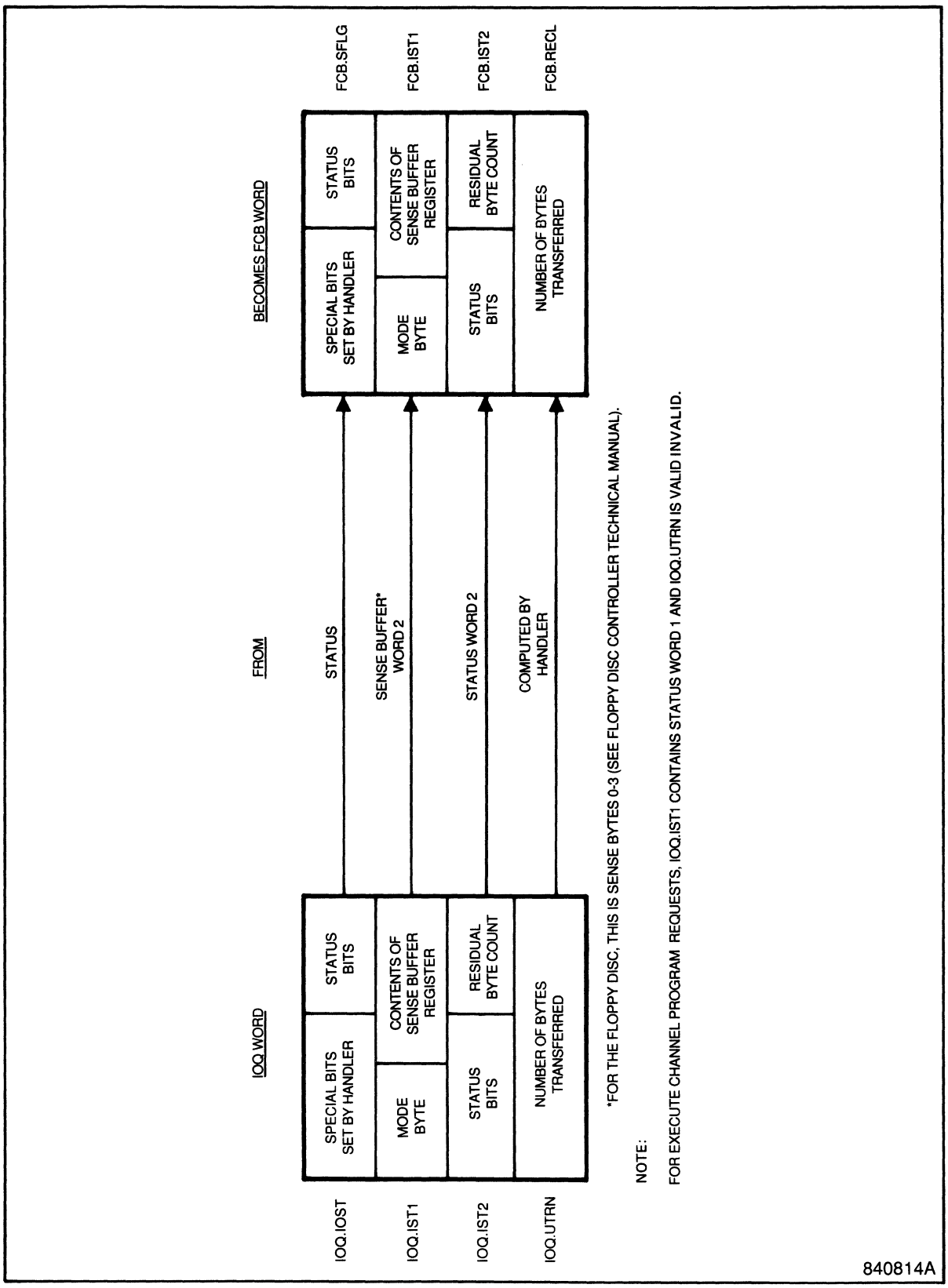



**Figure 3-1. Status Returned to User's FCB**

## 3.7 XIO/IOP Disc 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 | Disc 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 (5) retries of the entire IOCD list. If this fails, rezero and retry are performed.

| Sense Bit | Meaning |
|---|---|
| 13 | Data over or under run |
| 14 | Disc 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 | Disc addressing or seek error |
| **Drive Status Bit** | **Meaning** |
| 8 | Disc 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 Disc 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 |

| Status 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 (5) 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 discs configured in an MPX-32 system. See Volume III of the MPX-32 Reference Manual for details.

### 3.11 XIO Disc Processor/IOP Disc Processor Subaddressing

Each disc drive attached to an XIO disc processor or IOP disc 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 disc (DF). All direct references to the captive media portion of a cartridge module drive must specify fixed head disc (DF) as the device type code.

### 3.12 Floppy Disc Subaddressing

Valid floppy disc subaddresses are limited to 0 and 1. Either one or two SYSGEN DEVICE directives can be used when specifying floppy discs. 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 Disc Processor SYSGEN Directives

```
1. CONTROLLER=DM08,PRIORITY=14,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 disc processor on channel eight at priority level 14. The handler name (interrupt fielder) is H.IFXIO and is channel reentrant, one copy per channel.

2. This DEVICE directive specifies a 80MB moving head disc 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 discs 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 be 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 disc assigned to subaddress 0C.

### 3.14 Sample IOP Disc 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 disc 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 disc 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 Disc 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 disc 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 disc 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 disc assigned to controller address F and device subaddress 1.

4. This DEVICE directive specifies two floppy discs assigned to controller address F. The floppy discs are assigned device subaddresses of 0 and 1. The handler name is H.DCXIO and is channel reentrant, one copy per system.

# IOP Eight-Line Full Duplex Handler (H.F8XIO)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# IOP EIGHT-LINE FULL DUPLEX HANDLER (H.F8XIO)

## SECTION 1 - OVERVIEW

### 1.1 General Information

The IOP Eight-Line Full Duplex Handler (H.F8XIO) is system level re-entrant, i.e., only one copy is required regardless of the number of eight-line communication multiplexers configured in a system. Re-entrancy 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.

## SECTION 2 - ENTRY POINTS

### 2.1 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 fourteen 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 8-Line Asynchronous Communications Multiplexer Technical Manual.

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 | 1 = 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 |

## 2.2 I/O Queue Processor (IQ.XIO)

This entry point performs standard I/O queue scheduling.

## 2.3 Service Interrupt Processor (SI.)

This entry point consists of the six standard subroutines described in the Subroutines Section.

## 2.4 Lost Interrupt Processor (LI.XIO)

This entry point performs standard lost interrupt and kill I/O processing.

## 2.5 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.

## 2.6 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.

## 2.7 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.

# SECTION 3 - OPTIONS

## 3.1 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.

## 3.2 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.

## 3.3 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.

## 3.4 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 four in FCB.SCFG before issuing the read. This option is forced following a ring or break interrupt.

## 3.5 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).

## 3.6 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.

## 3.7 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.

## 3.8 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.

# SECTION 4 - SUBROUTINES

## 4.1 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.

## 4.2 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.

## 4.3 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.

## 4.4 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.

## 4.5 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 eight-line asynch.

## 4.6 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 and returning to XIO.SUB.

# General Purpose Multiplexer Support (H.GPMCS)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# GENERAL PURPOSE MULTIPLEXER SUPPORT (H.GPMCS)

## SECTION 1 - OVERVIEW

### 1.1 General Information

The General Purpose Multiplexer Controller (GPMC) is structured as follows:

- Because all General Purpose Device Controller (GPDC) handlers supplied are system re-entrant, only one copy of a handler is needed per system. Customers writing their own handlers do not need to make them re-entrant; however, if they are not, interrupt re-entrancy 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.

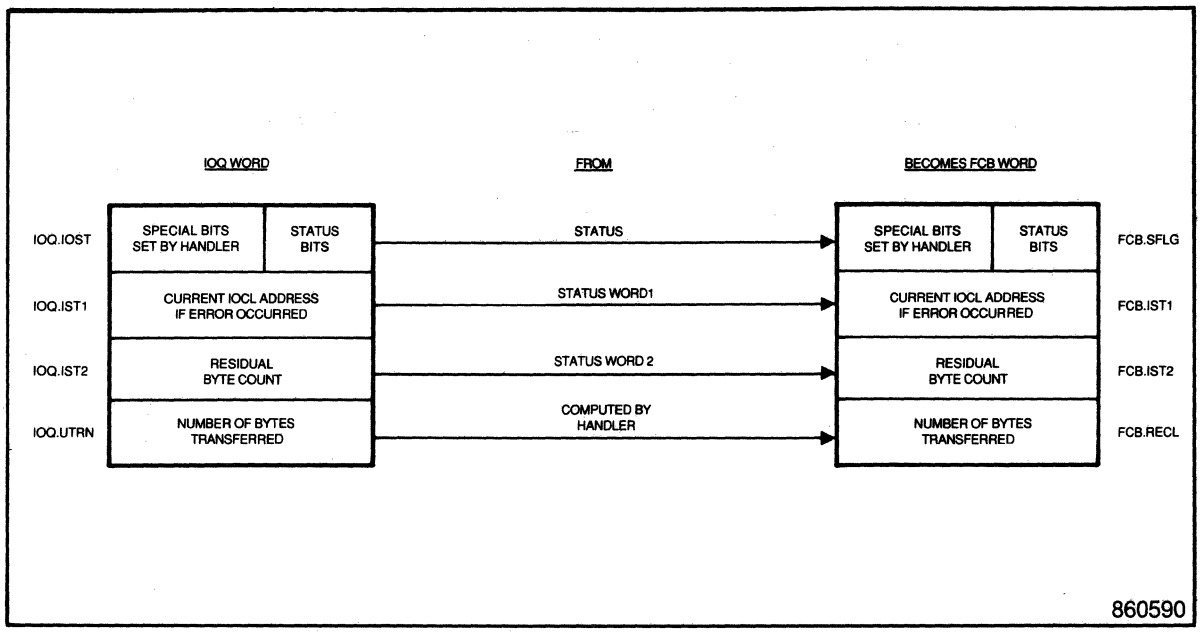
- SYSGEN creates only one Controller Definition Table (CDT) entry per GPMC (regardless of the number of device addresses). The CDT fields utilized are as follows:

| | |
|---|---|
| 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<br>Bit 1 (CDT.MUX) set to indicate a multiplexing controller. |
| CDT.SIHA | Points to the H.MUX0 HAT |
| CDT.UTn | Where n is a hexadecimal digit from 0 to F. The UTn 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:

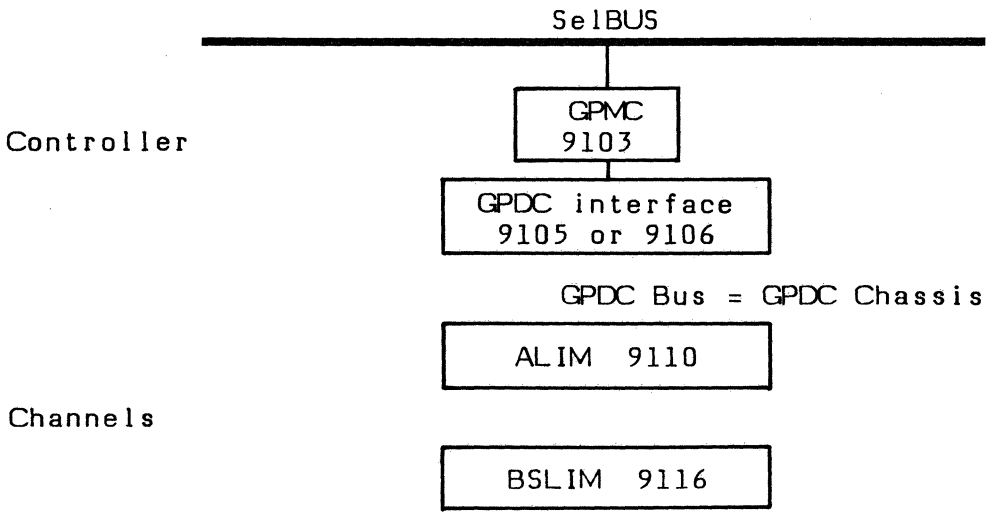



Customer written handlers must use the Device Context Area (DCA) because H.MUX0 and GPMC.SUB reference certain locations within the DCA.

GPMC supports the following devices:

| | |
|---|---|
| 9103 | Extended GPMC, Class D (16MB) |
| 9109 | Synchronous Line Interface Module (SLIM) |
| 9110 | Asynchronous Line Interface Module (ALIM) |
| 9112 | Paper Tape Reader/Punch Controller |
| 9116 | Binary Synchronous Line Interface Module (BLIM) |

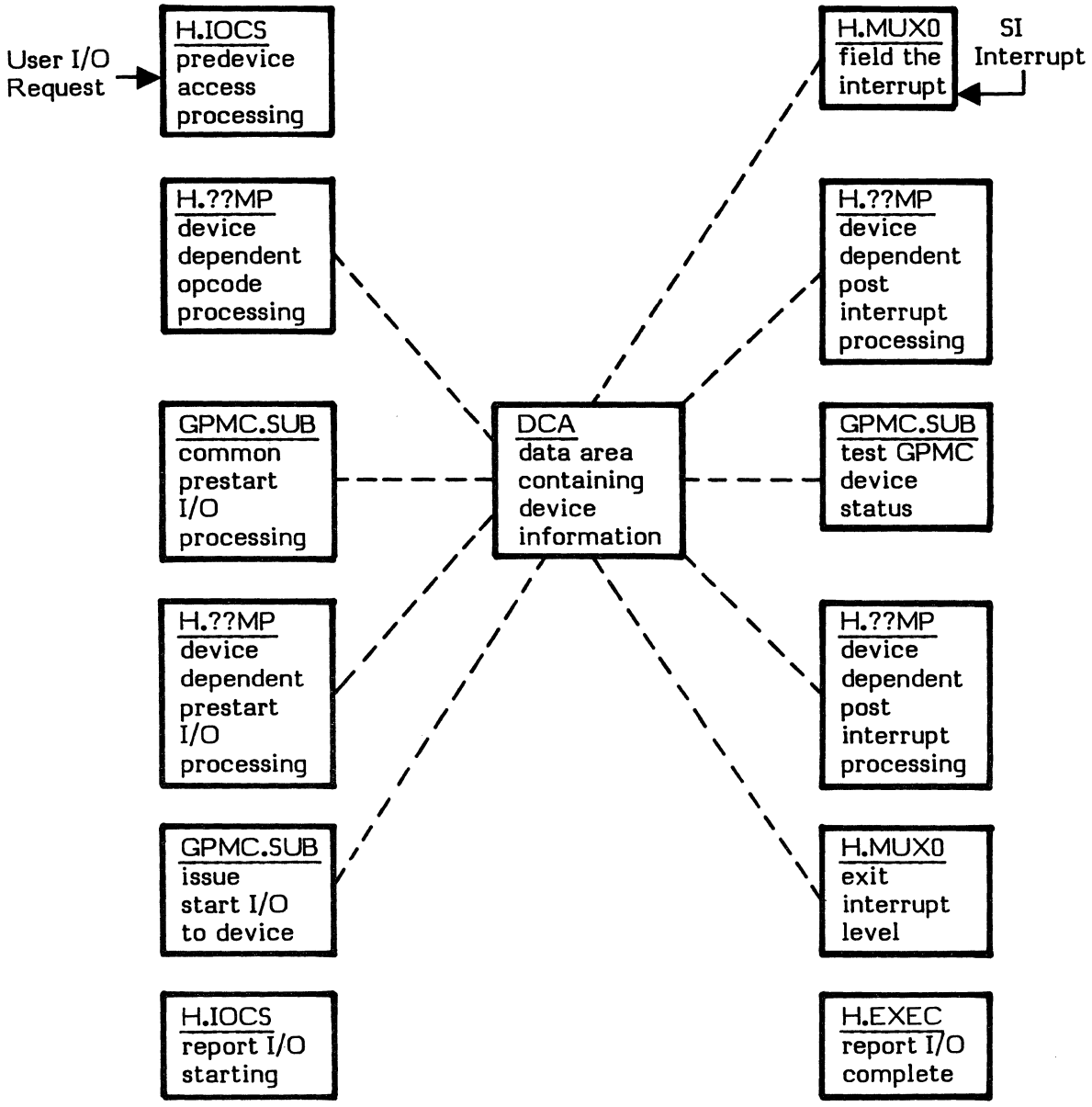
## 1.2 Hardware Structure



Refer to the GPMC Technical Manual, 325-329104/9103 for details of the GPMC operation. Refer to the appropriate technical manual part number 325-32xxxx, 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.



## SECTION 2 - 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 above.

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 two-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 the completion of a command issued to a device.

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:

1. Report I/O complete via the appropriate executive routine.
2. Restore the state of the machine to mapped, block all interrupts, and issue a deactivate request on the interrupt level being serviced.
3. 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 tally 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: | 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 tally of all spurious interrupts.

Entry Conditions

| | |
|---|---|
| Calling Sequence: | Entered from H.MUX0 |
| 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 is 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 is 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 with 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 |

Exit Conditions

| | |
|---|---|
| Return Sequence: | 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 re-entrant; therefore, only one copy is needed per system.

### 2.3.1 Subroutine S. GPMC0 - Report GPMC Status

| | | |
|---|---|---|
| Entry | R0 | Return address |
| | R2 | IOQE address |
| | R3 | Interrupt block address |
| Exit | R0-R4 | Unchanged |
| | R5,R6 | Destroyed |
| | R7 | Device status in right halfword |

### 2.3.2 Subroutine S.GPMC1 - I/O Initiation Logic

| | | |
|---|---|---|
| Entry | R2 | IOQ address |
| | R3 | Interrupt block address |
| | R6 | IOCL address |

### 2.3.3 Subroutine S.GPMC2 - Lost Interrupt Logic

| | | |
|---|---|---|
| Entry | R0 | IOCS return address |
| | R2 | Handler address (unused) |
| | R3 | UDT address |
| Exit (Direct to IOCS) | R0-R3 | Unchanged |
| | R4-R7 | Destroyed |

Refer to Section 2.1.4.

### 2.3.4 Subroutine S.GPMC3 - Operation 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 | R0 | H.??MP return address (not used) |
| | R1 | FCB address |
| | R2 | Opcode vector table address |
| | R3 | UAT address |
| Exit (Through vector table) | R0, R1 | Unchanged |
| | R3 | Interrupt block address |
| Other Registers | Indeterminate | |

### 2.3.5 Subroutine S.GPMC4 - Execute Channel Program 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:

- 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
- Store user specified time-out value in IOQ
- 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 Extended I/O Reads and Writes

This macro 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

# HIGH-SPEED DATA HANDLER (H.HSDG)

## SECTION 1 - OVERVIEW

### 1.1 General Information

The High-Speed Data (HSD) Handler is an optional software component of MPX-32 which provides general device support for user devices.

The handler design is based on the notion that the HSD hardware acts as a controller: it performs handshaking with the CPU and performs all SelBUS operations needed to fetch and store either data or status relating to requested operations. Therefore, references to the HSD imply controller functions which are generic to I/O operations in general. The device attached to the HSD is the user's addition. Little is known or presumed about its nature. The general assumption is that the device can source and/or synchronize data with the possibility of presenting device specific status on request.

For additional information, refer to the High Speed Data Interface Technical Manual.

# SECTION 2 - USAGE

## 2.1 Related Data Structures

The HSD handler provides a software interface between MPX-32 tasks and the HSD. The HSD, in turn, provides a hardware interface to a user device. The HSD is a D class I/O device that uses an Input/Output Command List (IOCL) located in the processor memory to provide the commands for the operation. The IOCL is made up of one or more Input/Output Command Blocks (IOCBs). For each I/O operation processed by the HSD handler, an IOCL is constructed. When the operation is possible, it is initiated by loading the address of the IOCL into the Transfer Interrupt (TI) location assigned to the device and executing the command device instruction START I/O for the HSD.

### 2.1.1 HSD I/O Command Block Structure

The HSD uses a four-word IOCB with the following format:

| Word | 0 7 | 8 15 | 16 23 | 24 31 |
|---|---|---|---|---|
| 0 | HSDCMD. See Note 1. | UDDCMD. See Note 2. | Transfer count. See Note 3. | |
| 1 | Data address or device command. See Note 4. | | | |
| 2 | Unused by hardware (software). See Note 5. | | | |
| 3 | HSD/device status word. See Note 6. | | | |

Notes:

1. HSDCMD is the HSD command and defines the operation of the controller with the following bit definitions:

| Bit | Meaning |
|---|---|
| 0 | When set indicates input transfer; reset for output. |
| 1 | Command transfer, word 1 of IOCB is sent to device. |
| 2 | Device status request, store into word 3 of IOCB. |
| 3 | Continue on error. |
| 4 | Interrupt when completed processing IOCB. |
| 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 and is passed by the HSD to the user device.

3. Transfer count is the count in words to transfer when the operation is not a command transfer or transfer in channel.

4. Data address is the physical address of data to be read or written, if data transfer operations, or the address of the IOCB to process next if a transfer in channel.

   Device command is a 32-bit value sent to the device if HSDCMD bit one is set.

5. Unused word is made available for use by the software.

6. HSD/device status word is used to store operation status whenever an interrupt on end-of-block is requested or to store external device status returned if HSDCMD bit two is set.

### 2.1.2 IOCB Classes

The HSD I/O Command List (IOCL) is composed of one or more I/O Command Blocks (IOCBs). Each IOCB is one of the following general classes depending on the bits set in HSDCMD:

- Device Command Transfer
- Device Status Transfer
- Transfer in Channel
- Data Transfer Request
- Data Chain Descriptor

#### 2.1.2.1 Device Command Transfer

Device commands are operations with the command transfer bit set in the HSDCMD byte. The second word of the IOCB is sent to the device. This is generally used to provide device addressing or commanding. This can be followed by a transfer request.

#### 2.1.2.2 Device Status Transfer

Under normal conditions, the status posted in the IOCB is controller status; for example, generic information about the transfer. If specific information about the device is desired, it must be requested by constructing an IOCB with bits two and four set in HSDCMD. This causes the HSD to request the specific device status, and store it in the IOCB word three. By convention, bit zero of the returned status in word three is set to a one to flag device status as opposed to controller (HSD) status.

#### 2.1.2.3 Transfer In Channel

This command is used to instruct the HSD to continue processing the IOCL at the address specified in the second word of the IOCB. This is a branch in the I/O command list. A TIC is generally used to link discontiguous IOCLs together to form one logically contiguous command set or it is used to cause the device to re-execute the I/O command list. In this case, the interrupt on end-of-block (word three of IOCB) is usually set to inform the software that the device has restarted the I/O command list.

#### 2.1.2.4 Data Transfer Request

All IOCBs which do not have the command transfer, device status request, or transfer in channel bits set are data transfer requests. A write request is indicated by resetting bit zero of HSDCMD.

#### 2.1.2.5 Data Chain Descriptor

This form of IOCB only occurs following a data transfer or another data chain descriptor IOCB. It has the effect of specifying a continuation of the current transfer operation. It performs a scatter write or gather read and accounts for the discontinuities in the logical-to-physical mapping of tasks in MPX-32.

### 2.2 HSD Request Processing

The handler accepts requests in two general formats. One looks very much like other MPX-32 I/O requests where the buffer address and transfer count, as well as an optional device control word, is included in the File Control Block (FCB). This is referred to as FCB format.

The other format permits the user to construct an I/O Command List (IOCL) and to have the FCB reference this list of commands to define the desired operation. This is referred to as START I/O format.

The START I/O format is indicated by I/O function EXCPM (Execute Channel Program) and the 15 remaining function codes are FCB format.

In START I/O format, there are two variations depending on whether the IOCL provided is logical or physical. A physical IOCL is a final device I/O command list ready to be processed by the device. All addresses are physical and discontinuities in the logical-to-physical mapping have been resolved by breaking the request into physically contiguous segments with data chaining. A logical IOCL is an I/O command list representing the desired operation into the logical address space of the task. It must be transformed to a physical IOCL. The addresses must be changed to physical, and logical address discontinuities must be accounted for by breaking the transfer into a series of data chained transfers.

For tasks which perform the same operation repeatedly, the overhead of translation may not be acceptable. The physical IOCL can be generated once and used on each subsequent request.

Tasks using a single physical IOCL must be privileged and nonswappable.

Physical IOCL is indicated by the task setting the Data Format Inhibit bit in the FCB general control flags byte.

### 2.2.1 FCB Format Request

This form of request provides an interface to the user device compatable with standard devices and provides the flexibility to control almost any device. The FCB interface is designed to permit a device command and/or a data transfer to be initiated as the result of a user request. The standard MPX-32 FCB used to request I/O contains fields to define buffer address, byte count, and device address. See Volume I, Chapter 5 of the MPX-32 Reference Manual for related FCB information.

For requests which involve data transfer, the buffer address and byte count describe the user buffer for the operation. The device address field in the FCB is used to supply the data to send to the device for device command transfers.

MPX-32 supports two variant forms of the FCB, normal and extended. Both forms are supported. However, the normal form has limited transfer count and device address fields which restrict functionality.

MPX-32 supports 16 I/O operation codes. The next section defines the opcode number, its name, its corresponding IOCS entry point number for reference, its use by the HSD handler, and special considerations.

#### 2.2.1.1 I/O Operation Codes

| Opcode | Name | EP | Use | Notes |
|---|---|---|---|---|
| 0 | OPEN | 1 | Device/handler init | Only called once until close |
| 1 | RWND | 2 | Device control function | (Rewind device) |
| 2 | READ | 3 | Input data transfer | May cause device command status |
| 3 | WRITE | 4 | Output data transfer | May cause device command status |
| 4 | WEOF | 5 | Device control function | (Write end-of-file) |
| 5 | EXCPM | 10 | STARTIO format request | Not used for FCB format |
| 6 | ADVR | 7 | Device control function | (Advance record) |
| 7 | ADVF | 8 | Device control function | (Advance file) |
| 8 | BKSR | 9 | Device control function | (Backspace record) |
| 9 | BKSF | 19 | Device control function | (Backspace file) |
| 10 | UPSP | 20 | Device control function | (Up space) |
| 11 | ERPT | 21 | Device control function | (Erase or punch trailer) |
| 12 | EJCT | 22 | Device control function | (Eject) |
| 13 | CLOSE | 13 | Device/handler reset | |
| 14 | RSVP | 24 | Device control function | (Reserve port) |
| 15 | RLSP | 27 | Device control function | (Release port) |

The following are I/O operation errors that can be returned in extended status word 2 of the FCB:

| Value | Explanation |
|---|---|
| 1 | Request made with non-expanded FCB |
| 2 | In FCB format and the transfer count was zero |
| 3 | In FCB format and the byte transfer count was not a word multiple of four 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 double word 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 | Either the format or 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 the operating system or below DSECT |

#### 2.2.1.2 Device Open

This function is intended for user device-specific processing as required. The standard handler will contain a null routine that can be expanded by the user as needed.

#### 2.2.1.3 Device Close

This is the complementary function to device open and will also consist of a null routine.

#### 2.2.1.4 Device Control Functions

All device control functions dispatch to a common routine in the device driver. An internal table, which could be written at open time, is used to set the UDDCMD field of the IOCB. The device address field from the FCB is used as the device command word (IOCB word one) and is sent to the device.

Users have the option of adding specific processing to the standard handler on a per operation code basis. The first FCB special flag bit, bit eight of word two of the FCB, indicates if device-specific status should be requested after the device command transfer. This results in a second IOCB being added to the IOCL. The device status is returned in extended I/O status word two of the FCB. This requires that the request be made using the expanded FCB.

#### 2.2.1.5 Data Transfer Initiate Requests

Input and output requests are processed in the same fashion. The only difference is the direction bit in the HSDCMD. The following special flag bits control the operation:

- Request device status after transfer (FCB.SCFG0) - This bit indicates that an IOCB should be added to the IOCL to retrieve device-specific status after the data transfer has completed.

- Send device command prior to data transfer (FCB.SCFG1) - This bit indicates that an IOCB should prefix the data transfer to transmit a device command word to the device. If the FCB is the expanded form, the value sent is the 32-bit expanded random access address. Otherwise, the value sent is the least significant 20 bits of word 2 of the IOCB, the random access address field.

- Disable time out on this request (FCB.SCFG2) - This flag indicates that the operation will take an indeterminable period of time and the handler should wait an indefinite period of time for the I/O to complete. This generally only applies to read operations.

- Set UDDCMD from least significant byte of word two (FCB.SCFG3) - This bit indicates that the UDDCMD byte in the data transfer operation should be set from the least significant byte of the random access address field of the FCB. This provides the ability to pass additional control information to the device without modifying the device driver.

### 2.2.2 STARTIO Format Requests

The STARTIO format allows tasks to utilize all hardware capabilities available from the device/HSD interface.

A STARTIO format request is indicated by the operation code EXCPM. The data format inhibit flag bit indicates request with physical IOCL. For physical IOCL requests, the handler simply queues the request for processing. For logical requests, the IOCL must be made physical. This involves three transformations:

- All addresses in data transfer IOCBs must be converted to physical.
- Each time a transfer crosses a physical boundary (2KW), the request must be broken at the boundary and data chaining must be specified to account for the physical discontinuity of the task address space.
- Transfer in Channel request addresses must be updated to reflect the final address of the target IOCB. This must account for expansions in the IOCL due to intervening data transfers that have crossed map blocks.

The HSD device handler in MPX-32 supports two additional functions as part of the start I/O protocol. They are:

- Subtract one and branch nonzero IOCB (SOBNZ).

- Asynchronous status presentation.

These functions are initiated when the HSD handler receives a non-terminal interrupt. An interrupt when Transfer Interrupt (TI) status does not indicate end-of-line (EOL). This indicates that the interrupt on the end-of-block bit was set on an IOCB that was not last in the list. The IOCB could be a TIC or a command with the command chaining bit set.

#### 2.2.2.1 Subtract One and Branch Nonzero

This function is a special version of the Transfer in Channel IOCB. It provides the ability for the task to specify that a series of IOCBs may be executed a specific number of times.

| Word | 0 7 8 | 15 16 | 23 24 31 |
|---|---|---|---|
| 0 | HSDCMD | UDDCMD | Not used |
| 1 | Address of next IOCB | | |
| 2 | Initial count | | Current count |
| 3 | Normal next IOCB address | | |

The HSDCMD specifies Transfer In Channel (TIC) and interrupt on end of block. When processing starts, the initial count and current count values are equal to the number minus one of times to execute the IOCB loop. The contents of word one and three are equal of the address of the IOCB to execute in the loop.

The HSD interrupts each time it processes the TIC. The handler detects the interrupt to be a SOBNZ function and subtracts one from the current count value. When the count equals zero, word one is modified to contain the IOCB following the SOBNZ. On the next pass, the interrupt indicates that processing has passed out of the loop. The handler resets word one to the loop IOCB from word three and resets the current count from the left halfword of word two.

#### 2.2.2.2 Asynchronous Status Presentation and Notification

Asynchronous status presentation and notification is a software analog to the hardware capability to receive an interrupt and HSD status on the completion of each block, or the ability to explicitly request status from the device at the completion of any operation.

- Asynchronous status presentation - When a logical IOCL is converted to physical and asynchronous status is required, the address of the logical IOCB is saved in word two of the physical IOCB. When the HSD handler receives a nonterminating interrupt, the IOCB indicated by the TI location indicates that status should be posted in the user space by copying word three of the physical IOCB to word three of the logical IOCB and by adding to the right halfword of word two of the logical IOCB to indicate that the status changed.

- Asynchronous notification - This is 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.

Whenever the HSD updates status in the task's logical address space, it checks to see if the previously requested notification has been delivered. If it has, the request for notification interrupt is requeued to the Task Interrupt Dispatch Queue. If it has not been delivered, nothing further is done. Whenever the task receives the interrupt, it must check for all possible status changes until it finds one that has not changed. On the next interrupt, the task can be assured that the status will have changed since the hardware executes the IOCL in a well-defined order. The task should know where to look for status changes.

### 2.3 HSD I/O Request Processing Details

When an I/O request is made by a task in MPX-32, IOCS performs some initial validation. If validation is successful, IOCS transfers control to the associated device

driver at entry point five with the address of the user's FCB as input. Byte zero of the FCB is set to contain the requested operation code in the range of 0 to 15. The handler performs an indexed jump based on this value. In the standard driver, this will result in codes 0 to 13 going to the FCB format request routine, 14 to the logical IOCL routine, and 15 to the physical IOCL routine.

### 2.3.1 FCB Format Request Processing

Processing of the FCB request is done in two phases. First, the required size of the IOCL is computed. This is added to the size of the I/O Queue Entry (IOQ) and the IOQ is allocated from the system's memory pool. The I/O queue portion is then initialized by the IOCS routine INIT.IOQ. The IOCL is then constructed according to the tables based on the task's operation code. The request is queued to the CDT and initiated in its turn. When it completes, the final status is sent into the FCB and the request is terminated.

#### 2.3.1.1 FCB Request IOCL Size Computation

The IOCL for FCB format request has from one to three sections in it. They are any combination of:

- Device Command Transfer IOCB
- Data Transfer IOCL
- Device Status Request IOCB

The device command and status IOCBs are always one word. Space is allocated if the related function is required. The data transfer IOCL is one IOCB to initiate the request and one additional IOCB for each map block crossed by the transfer. The total size is the sum of the required pieces.

#### 2.3.1.2 FCB Request IOCL Construction

The following procedure defines the construction of the IOCL:

1. Set IOCBPTR to first IOCB.

2. If device command transfer is required, then:
   - Set HSDCMD to CMDXFER
   - Set device address from FCB to word one of IOCB

   If data or device status transfer is required, then:
   - Set command chain in HSDCMD
   - Advance IOCBPTR
   - ENDIF

   ENDIF

   or

   If data transfer, then:
   - Set HSDCMD to read
   - If transfer is write, then set output bit in HSDCMD

W: set XFERADR in IOCB to transfer address
. Set XFERCNT in IOCB to number of bytes within map block
. Add XFERCNT to transfer address
. Subtract XFERCNT from transfer count

If transfer count is greater than zero, then:
. Set data chain in HSDCMD
. Advance IOCBPTR
. Go to W
. ENDIF

3. If device status request, then:
   . Set command chain in HSDCMD
   . Advance IOCBPTR
   . ENDIF

4. If device status transfer, then:
   . Set HSDCMD to XFRDEVSTAT

5. Set interrupt on end-of-block in HSDCMD (for last IOCB).

### 2.3.2 Logical IOCL STARTIO Format Request

Users of this form of request will generate a logical IOCL. This will contain some number of IOCBs that define their operation. The IOCL will perform any one of the following operations:

. Command device

. Transfer in channel

. Data transfer

. Data chain

The logical IOCL must be converted to a physical IOCL since the device deals in terms of physical addresses which can not be generated by a nonprivileged user. Also, transfers which cross map block boundaries must be separated using data chaining to allow for physical memory discontinuities. The entire IOCL is copied to the system memory pool to perform the conversion. This insures that the physical IOCL is physically contiguous.

### 2.3.3 Conversion of Logical to Physical IOCL

The conversion process varies for each type of IOCB in the logical IOCL. Command device IOCBs need no conversion other than the move to the physical IOCL.

Transfer in Channel (TIC) IOCBs must be moved and the physical address of the target IOCB must be found. This results in a two-pass conversion of the IOCL. During pass one, the TICs are simply copied to the physical IOCL and word three of each IOCB in the physical IOCL is loaded with the logical address of its counterpart in the logical IOCL. This serves as a logical to physical address translation table. At the end of pass one, when the IOCL has been expanded to its maximum size, pass two is initiated. Pass two searches the physical IOCL for TICs and then searches the physical IOCL for the IOCL which corresponds to the logical IOCB referenced in the TIC. The address is updated.

Data transfer and data-chained entries are handled the same way as TICs. Each time the requested transfer crosses a map block, the IOCB is divided and data chaining is specified. This results in the physical IOCL becoming larger than its logical counterpart, which necessitates the use of the translation table for conversion. (Same as previously described for TICs.)

The size of the physical IOCL is equal to the size of the logical IOCL plus the additional space required for additional data chain entries needed. Therefore, the size is computed by adding the number of TIC, device command, and device status IOCBs plus the number needed to map each data transfer initiate or data chain IOCB.

### 2.3.4 Physical IOCL Processing

Service of this request is a subset of the logical IOCL processing because the IOCL needs no conversion. Tasks must be privileged and resident to use this form of the request. An I/O queue and a notification packet is allocated for processing this request. When asynchronous status is presented by the device, the right halfword of word two of the IOCB is bumped and a task interrupt is requested.

## 2.4 Common Request Handling

For all requests, an I/O queue is allocated and set to describe the request. This includes the IOCL address to account for the variation in the location of the IOCL, depending on the request type.

## 2.5 Product Relationships

The HSD handler becomes an integral part of the MPX-32 system if it is included at SYSGEN time. The handler requires MPX-32 Release 1.2 and above software to operate.

## 2.6 Device Considerations

Development of the generic HSD handler has uncovered several characteristics of the HSD which would require a significant increase in the overhead of using the device if programmed around. They are as follows:

- The device will not interrupt on end-of-list unless explicitly told to do so. It is the software's responsibility to insure that the IOE bit is set if an IOCB does not have command or data chaining set to insure that an interrupt is generated when the device goes idle.

- In the case of nonpresent memory or FIFO overflow when the IOCB has continue on error set, the device will post status and only interrupt if the interrupt on end-of-block bit is set. The device should interrupt when it posts error status.

  When the HSD is operated in mode two, only nonpresent memory errors are affected by continue on error. Therefore, it is recommended that mode two be used and continue on error be avoided. Nonpresent memory errors are serious and their cause should be determined and eliminated.

- 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 will stop processing the IOCL. The device will not interrupt unless the IOE bit was set in the IOCB currently being processed or the device is operating in mode two. Interrupt on end-of-block for an IOCB of this nature is not the normal case. This behavior varies depending on whether it is the last data chaining IOCB. Also, if command chain and no interrupt on end-of-block is set in the IOCB where device end-of-block is posted, no residual byte count is given. Therefore, if there is no interrupt, no residual byte count is posted.

  This problem does not exist in mode two. Therefore, it is recommended that the device be operated in mode two to avoid problems.

- When external mode is active and the software issues a CD start I/O, the device rejects the command with a privilege violation but no indication is given to the software. The operation will time out, resulting in a halt I/O being issued that kills the currently running external operation.

  This problem only becomes apparent when the device is operated in a combination of normal (internal) and external control mode. This is probably rare; however, there is no definite solution. One approach is to have the user device force an error at the end of each external mode transfer to cause an interrupt each time. If the handler had started a transfer, it would know to restart the operation.

  Note: Any device time out must be set long enough to insure that the external transfer can complete to prevent an abortion in the device time out routine.

- If the interrupt bit is set on a TIC, no SI status is posted, the interrupt is delivered, and the TIC address is lost. This causes the device behavior to be unpredictable.

  Interrupts on TIC IOCBs must be avoided. When the software needs to know that the HSD has executed a TIC, it should set the interrupt on end-of-block on the previous IOCB. By the time the software finds out what is going on, the HSD will have long since executed the TIC and will be off processing the IOCL at the point indicated by the transfer in channel IOCB. The speed of the HSD guarantees this. Also, if the TIC is being used to effect a channel program loop and manages to complete the IOCB preceding the TIC before the software has deactivated the interrupt, the HSD will stall and wait until the level becomes inactive. Therefore, if the software wants to perform a loop counting operation and modify the branch address, it can do so with repetitive behavior as long as the interrupt service routines run with the interrupt level active until after the adjustment or a decision is made to not adjust the TIC destination address.

# Memory Disc Handler (H.MDXIO)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

# MEMORY DISC HANDLER (H.MDXIO)

## SECTION 1 - OVERVIEW

### 1.1 General Information

The Memory Disc Handler (H.MDXIO) is a software component of MPX-32 that controls a memory disc. H.MDXIO also translates disc 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 disc. No-wait I/O is supported for memory discs, as are all functions previously available to users of disc files. Unlike other discs, memory disc 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 discs 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. See Sections 2.1.17 and 2.1.18.

Warning: Memory disc cannot be used as a physical cache disc.

### 1.2 Hardware/Software Relationship

The memory disc 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 discs. XIO.SUB is the XIO common subroutine package the disc 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 discs can be configured on channel 00 on even only subchannels (starting on subchannel 00). However, a memory disc cannot be configured on the same channels and subaddress as the null device. This reduces the number of memory discs to seven for most systems.

## 1.3 Size of Memory Discs

The maximum size of a memory disc is limited by the physical memory that is available and the memory requirements of system tasks.

The minimum size of the memory disc is limited by the fixed amount of space occupied on a disc after formatting. To calculate the required size of a memory disc, add the following items:

- 17 KB that are filled with the number of directories and resource descriptors.
- The total size of required data on discs, like directories and temporary files.
- 768 bytes for each resource descriptor specified by the MAXRES and MAXROOT parameters of J.VFMT.

# SECTION 2 - USAGE

## 2.1 Dual-port Memory Disc Support

Dual porting allows two CPUs to share a single memory disc through the Multiprocessor Shared Memory System (MSMS). A dual-ported memory disc is only located in MSMS memory because the memory disc semaphore (bit 0 of the memory disc) must be cached to the other CPU. Memory Bus Controller (MBC) shared memory does not cache bit settings.

In order to maintain disc and system integrity, mechanisms must exist to prevent both CPUs from accessing the memory disc 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.

### Implicit Device Reservation

Implicit device reservation is processed by the memory disc dual-port semaphore. When dual-port access to the memory disc is required, the processor attempts to set the semaphore. When successful, the memory disc I/O is performed.

### Explicit Device Reservation

Explicit device reservation makes a memory disc 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 disc 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 disc environment, one of the systems may fail while the shared memory disc is reserved. If this happens, the shared memory disc can be accessed by the opposing processor through the J.UNLOCK system task. See Chapter 3 of the Technical Manual, Volume I.

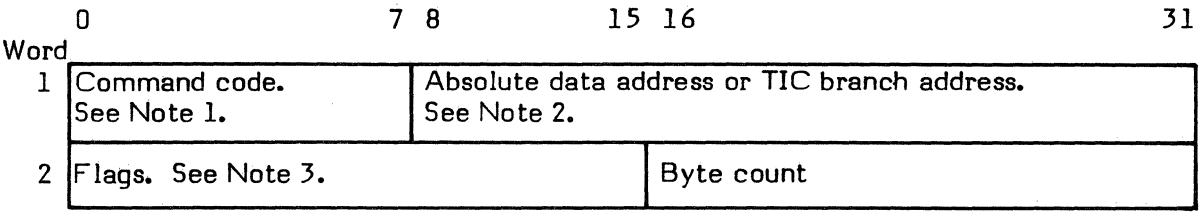
## 2.4 Maximum Byte Transfer and IOCD Generation

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 | Postprogram 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 Transfer in Channel (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 Write Data (WD) command is a user write request that transfers data to the disc from the address specified in the IOCD. WD is used for a user write request.

### 2.5.3 Read Data (RD)

The Read Data (RD) command transfers data from the disc to the address specified in the IOCD. RD is used in the user read request.

### 2.5.4 Read Track Label (RTL)

When the Read Track Label (RTL) command is specified, the H.MDXIO handler returns a pseudotrack label with the following information changed in the track label buffer:

| Bytes | Contents |
|---|---|
| 12-15 | Number of sectors on the memory disc |
| 27 | Number of sectors per track (20) |
| 28 | Number of heads (1) |

### 2.5.5 Reserve (RES)

The Reserve (RES) command reserves a memory disc 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 disc 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 Release (REL) command releases a reserved memory disc from the reserving CPU. The release is not issued if more than one task has the memory disc reserved. REL is user callable through the M.RELP service routine and is associated with dual-port memory disc operations. Execute channel programs must never include a release command and should use the M.RELP service routine when memory disc release is desired.

### 2.5.7 Rezero (XEZ)

The Rezero (XEZ) command is a recalibration request to the memory disc 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 disc 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 disc 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 Volume I Chapter 2 for a description of the common section.

| Word | Hex byte | 0 15 16 31 |
|---|---|---|
| 36<br>37 | 90 | Rezero IOCD used for error retry (DCA.REZO) |
| 38<br>39 | 98 | TIC IOCD used with rezero (DCA.TIC) |
| 40<br>41 | A0 | Load mode IOCD prototype (DCA.LMOD) |
| 42<br>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<br>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 disc (DCA.OFFS) |
| 53 | D4 | Size of memory disc in bytes (DCA.MSIZ) |
| 54 | D8 | Flags for memory disc (DCA.MDFL) |
| 55 | DC | Address of Shared Memory Table (SMT) for memory disc (DCA.SMTA) |
| 56 | E0 | Sector or cylinder for disc (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.

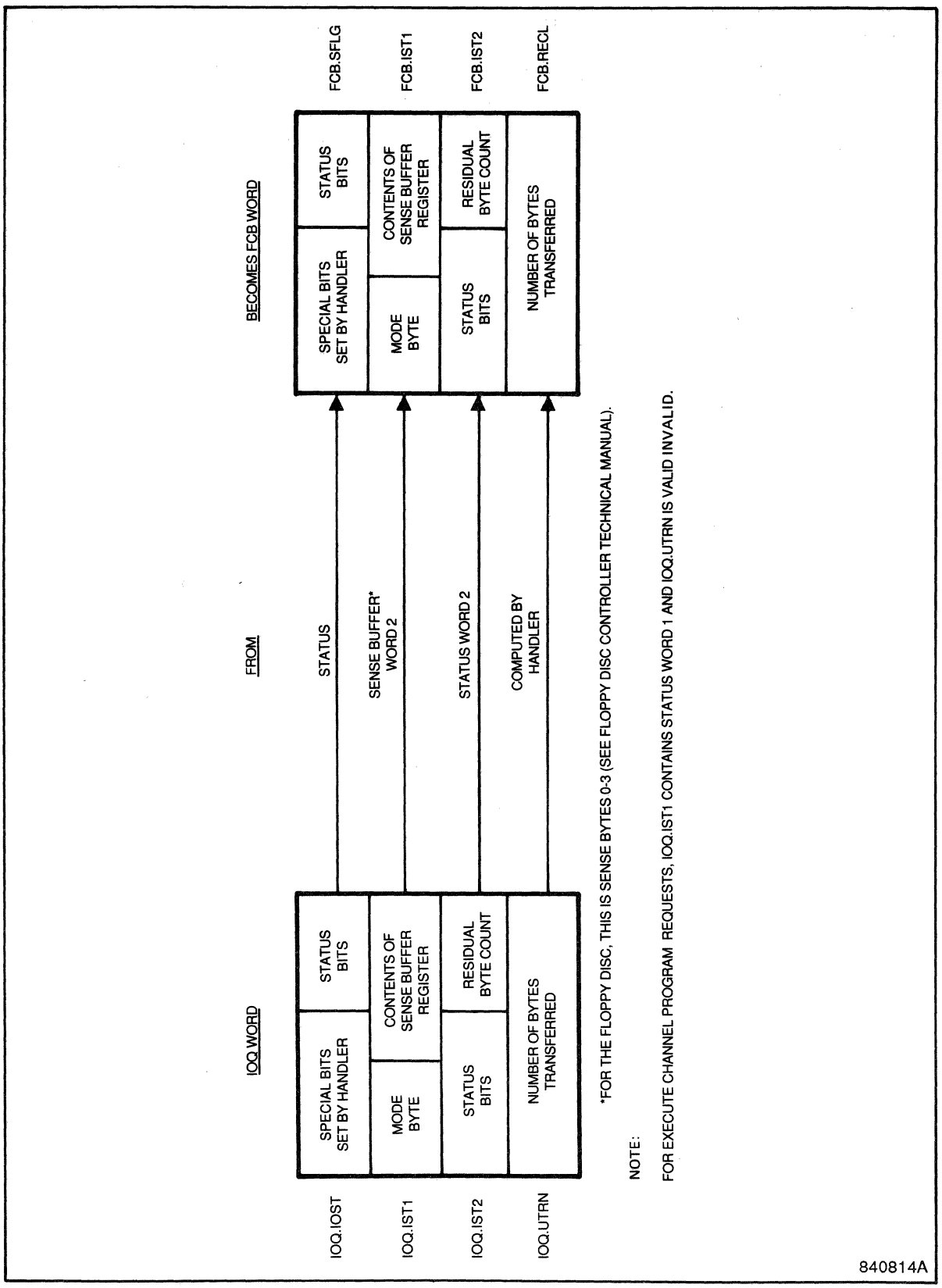



**Figure 2-1. Status Returned to User's FCB**

## 2.7 Error Processing for Execute Channel Program Requests

Requests for I/O outside the memory disc 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 discs 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 disc 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 eight memory discs on channel zero.

2. This DEVICE directive assigns a 1024KB memory disc to subaddress 02. The handler is H.MDXIO and is system re-entrant (one copy per system).

3. This DEVICE directive assigns a 1024KB dual-ported memory disc to subaddress 04 starting at decimal map block 256. The handler is H.MDXIO.

# SECTION 3 - 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.

# XIO Common Subroutine Package & Device Handlers (H.XIOS)

## MPX-32 Technical Manual

## Volume II

## CONTENTS

## Figure

Figure



## Table

Table

# XIO COMMON SUBROUTINE PACKAGE AND DEVICE HANDLERS (H.XIOS)

## SECTION 1 - OVERVIEW

### 1.1 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.

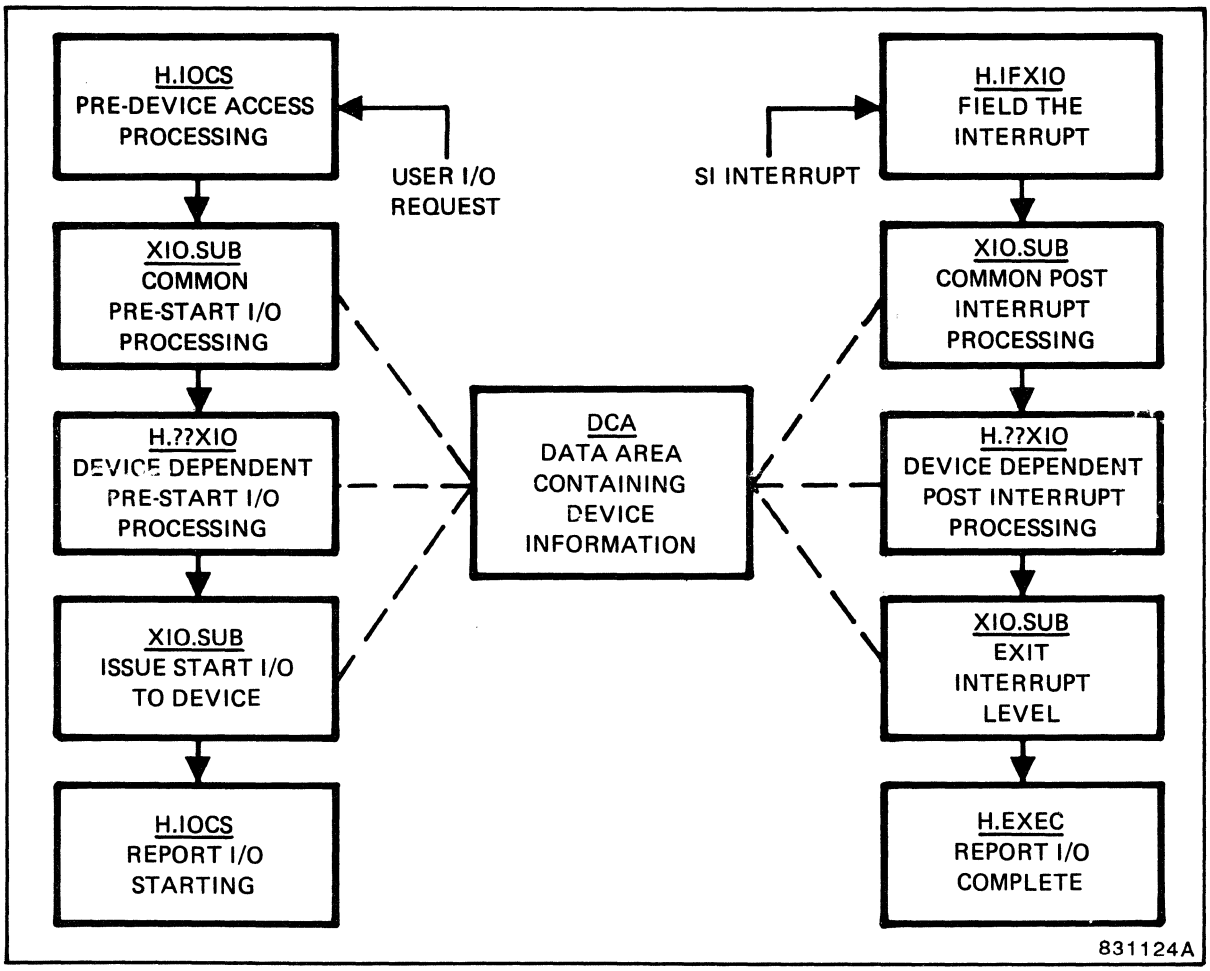



**Figure 1-1. Simplified Software Block Diagram**

# SECTION 2 - COMMON XIO SUBROUTINES PACKAGE (XIO.SUB)

## 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

### 2.2.1 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 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 |
| Register: | R7 | IOQ address | |

### 2.2.2 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 | |

### 2.2.3 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 |

## 2.3 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 Section 2.3.4 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

### 2.3.1 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 | Is 0 if entered from the XIO interrupt fielder (H.IFXIO) |
| | (or) | |
| | | Is the UDT address of the device on whose behalf the SIO or HIO was issued when the status stored response was generated. |

Exit Conditions

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 |

### 2.3.2 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 Section 2.3. |

### 2.3.3 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 Section 2.3. |

### 2.3.4 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. |

## 2.4 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 | |

## 2.5 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 | |

## 2.6 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.

# SECTION 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 so, all calls to this entry point are funneled to common processing. See Section 2.2 for details of the IQ.XIO subroutine.

### 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 | |

### 3.4.1 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

### 3.4.2 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

### 3.4.3 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.

### 3.4.4 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

### 3.4.5 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

### 3.4.6 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.

## 3.5 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. See the LI.XIO subroutine for details.

## 3.6 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 |

## 3.7 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 | |

### 3.8 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 | |

# SECTION 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 sections which follow.

```
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 | |

# APPENDIX A

## 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 |

| Macro name | Description | Reference |
|---|---|---|
| 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 |
| 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 |
| M.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 |

| Macro name | Description | Reference |
|---|---|---|
| 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 |
| 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 |

| Macro name | Description | Reference |
|---|---|---|
| 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 |
| 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 |

| Macro name | Description | Reference |
|---|---|---|
| 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; H.ALOC,8 |
| 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 |
| 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.HOLD and M_HOLD | Program Hold Request | H.REXS,25; H.MONS,25 |

| Macro name | Description | Reference |
|---|---|---|
| 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 |
| M.LOG | Permanent File Log | H.FISE,15; H.MONS,33 |
| M.LOGR and M_LOGR | Log Resource or Directory | H.VOMM,10 |

| Macro name | Description | Reference |
|---|---|---|
| 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<br>Load and Execute Overlay Segment | H.REXS,13; H.MONS,13<br>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_OPTIONWORD | Task Option Word Inquiry | H.REXS,24; H.MONS,24 |
| 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 |
| 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 |

| Macro name | Description | Reference |
|---|---|---|
| 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 |
| M.RSML and M_RSML | Resourcemark Lock | H.REXS,62 |

| Macro name | Description | Reference |
|---|---|---|
| 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.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 |

| Macro name | Description | Reference |
|---|---|---|
| M.SUSP and M_SUSP | Suspend Task Execution | H.REXS,17; H.MONS,17 |
| 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 |

| Macro name | Description | Reference |
|---|---|---|
| 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 |
| 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 |

**Gould Inc., Computer Systems Division**
6901 W. Sunrise Blvd.
P. O. Box 409148
Fort Lauderdale, FL 33340-9148
Telephone (305) 587-2900

GOULD
*Electronics*

## Users Group Membership Application

USER ORGANIZATION: _____

REPRESENTATIVE(S): _____

_____

_____

ADDRESS: _____

_____

TELEX NUMBER: _____ PHONE NUMBER: _____

NUMBER AND TYPE OF GOULD CSD COMPUTERS: _____

_____

OPERATING SYSTEM AND REV. LEVEL: _____

_____

APPLICATIONS (Please Indicate)

1. EDP
   - A. Inventory Control
   - B. Engineering & Production Data Control
   - C. Large Machine Off-Load
   - D. Remote Batch Terminal
   - E. Other

2. Communications
   - A. Telephone System Monitoring
   - B. Front End Processors
   - C. Message Switching
   - D. Other

3. Design & Drafting
   - A. Electrical
   - B. Mechanical
   - C. Architectural
   - D. Cartography
   - E. Image Processing
   - F. Other

4. Industrial Automation
   - A. Continuous Process Control Op.
   - B. Production Scheduling & Control
   - C. Process Planning
   - D. Numerical Control
   - E. Other

5. Laboratory and Computational
   - A. Seismic
   - B. Scientific Calculation
   - C. Experiment Monitoring
   - D. Mathematical Modeling
   - E. Signal Processing
   - F. Other

6. Energy Monitoring & Control
   - A. Power Generation
   - B. Power Distribution
   - C. Environmental Control
   - D. Meter Monitoring
   - E. Other

7. Simulation
   - A. Flight Simulators
   - B. Power Plant Simulators
   - C. Electronic Warfare
   - D. Other

8. Other

Please return to:

Users Group Representative

Date: _____

## Gould Inc., Computer Systems Division Users Group. . .

The purpose of the Gould CSD Users Group is to help create better User/User and User/Gould CSD communications.

There is no fee to join the Users Group. Simply complete the Membership Application on the reverse side and mail to the Users Group Representative. You will automatically receive Users Group Newsletters, Referral Guide and other pertinent Users Group activity information.

Fold and Staple for Mailing

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

**BUSINESS REPLY MAIL**

FIRST-CLASS MAIL PERMIT NO. 947 FT. LAUDERDALE, FL

POSTAGE WILL BE PAID BY ADDRESSEE

**GOULD INC., COMPUTER SYSTEMS DIVISION**
ATTENTION: USERS GROUP REPRESENTATIVE
6901 W. SUNRISE BLVD.
P. O. BOX 409148
FT. LAUDERDALE FL 33340-9970

Fold and Staple for Mailing

(Detach Here)

GOULD
*Electronics*