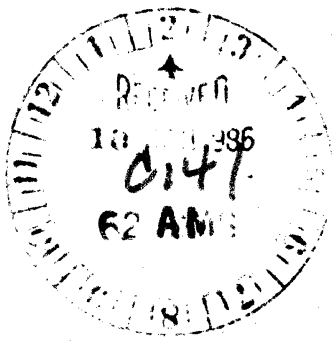


For Training Only

MPX-32

Release 1.5B

Reference Manual

Volume I

September 1982

Publication Order Number: 323-003661-000

GOULD

Electronics & Electrical Products

JUN83

This manual is supplied without representation or warranty of any kind. Gould Inc., S.E.L. Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

LIMITED RIGHTS LEGEND

for

PROPRIETARY INFORMATION

The information contained herein is proprietary to Gould S.E.L. and/or its vendors, and its use, disclosure or duplication is subject to the restrictions stated in the Gould S.E.L. license agreement Form No. 620-06(1/82) or the appropriate third-party sublicense agreement. The information is provided to government customers with limited rights as described in DAR 7-104.9A.

Copyright 1982
Gould Inc., S.E.L. Computer Systems Division
Printed in the U.S.A.

HISTORY

The MPX-32 Release 1.0 Reference Manual, Publication Order Number 323-001011-000, was printed June, 1979.

Publication Order Number 323-001011-100 (Revision 1, Release 1.3) was printed February, 1980.

Publication Order Number 323-001011-200 (Revision 2, Release 1.4) was printed July, 1980.

Publication Order Number 323-003661-000 (Revision 3, Release 1.5B) was printed September, 1982. The updated manual contains the following pages:

Title page
Copyright page
iii/iv through xxi/xxii
1-1 through 1-24
2-1 through 2-36
3-1 through 3-19/3-20
4-1 through 4-77/4-78
5-1 through 5-66
6-1 through 6-42
7-1 through 7-92
8-1 through 8-90
A-1 through A-7/A-8
B-1 through B-21/B-22
C-1 through C-39/C-40
D-1 and D-2
E-1 and E-2
F-1 through F-3/F-4
G-1 through G-3/G-4
GL-1 through GL-11/GL-12
IN-1 through IN-17/IN-18

CONTENTS

2. SYSTEM OVERVIEW

3. TASK STRUCTURE AND OPERATION

5. INTERACTIVE PROCESSING

6. BATCH PROCESSING

7. FILE AND DEVICE ALLOCATION AND I/O

8. SYSTEM SERVICES

ILLUSTRATIONS

TABLES

Documentation Conventions

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

lowercase letters

In command 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 command 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 commands, parameters, or instructions have been omitted. For example,

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

means one or more commands 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 command 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▲

A subscript delta specifies a required space. For example,

EDT > STO▲TSSPGM

means a space is required between O and T.

1. INTRODUCTION

The SYSTEMS Mapped Programming Executive (MPX-32) is a disc-oriented, multiprogramming operating system that supports concurrent execution of multiple tasks in interactive, batch, and real time environments. MPX-32 provides functionality in many areas, including memory management, terminal support, multiple batch streams, and intertask communication.

MPX-32 employs the SelMAP to fully support the 16MB address space of the 32 Series. Each task executes in a unique address space which may be expanded under task control up to 2MB of memory. An integrated CPU scheduler and a swap scheduler provide efficient use of main memory by balancing the in-core task set based on time-distribution factors, software priorities, and task state queues. The SelMAP is used to perform dynamic relocation of tasks during inswap.

Tasks operating under MPX-32 can be activated and/or resumed by hardware interrupts, system service requests, interactive commands, job control directives, or by the expiration of timers. Multiple copies of a task can be executed concurrently in interactive, batch, or real time environments. Through its various scheduling capabilities, MPX-32 provides the flexibility needed to adapt system operation to changing real time conditions.

The MPX-32 software package is composed of various software modules including the resident OS (IOCS, CPU and Swap schedulers, Resource Allocator, File System Executive, and reentrant system services), device and interrupt handlers, a Terminal Service Manager (TSM), a system generator (SYSGEN), and utilities such as a Text Editor, Debugger, and File Manager. The Macro Assembler (Model 1011) is also provided as part of the MPX-32 package. Figure 1-1 describes the system nucleus and utilities.

Memory-only MPX-32 (MPX-32/M) is a condensed version of the MPX-32 operating system that does not support any file structure capability. The system is designed for use in real-time environments where all tasks reside permanently in memory and limited operator communications is required. System installation and initial task loading/activation is accomplished via a System Distribution Tape (SDT) giving the user the option of loading a modified version of the operator communications task which allows limited communications with the system via the operator's console.

The Internal Processing Unit (IPU) is a second central processor designed to work with a Series 32 CPU to increase system throughput. The IPU is attached to the SelBUS like the first CPU and shares all memory (including the resident operating system area) with the first CPU. The IPU's function is to execute user's task level code in parallel with CPU operation. (The IPU is optional hardware and must be specified during SYSGEN for use on a system.)

To avoid contention between the IPU and CPU, there are some limitations on what the IPU can do:

- It cannot communicate with peripherals (perform I/O).
- It cannot process supervisor call (SVC) or call monitor (CALM) system services.
- It cannot execute interrupt control instructions.

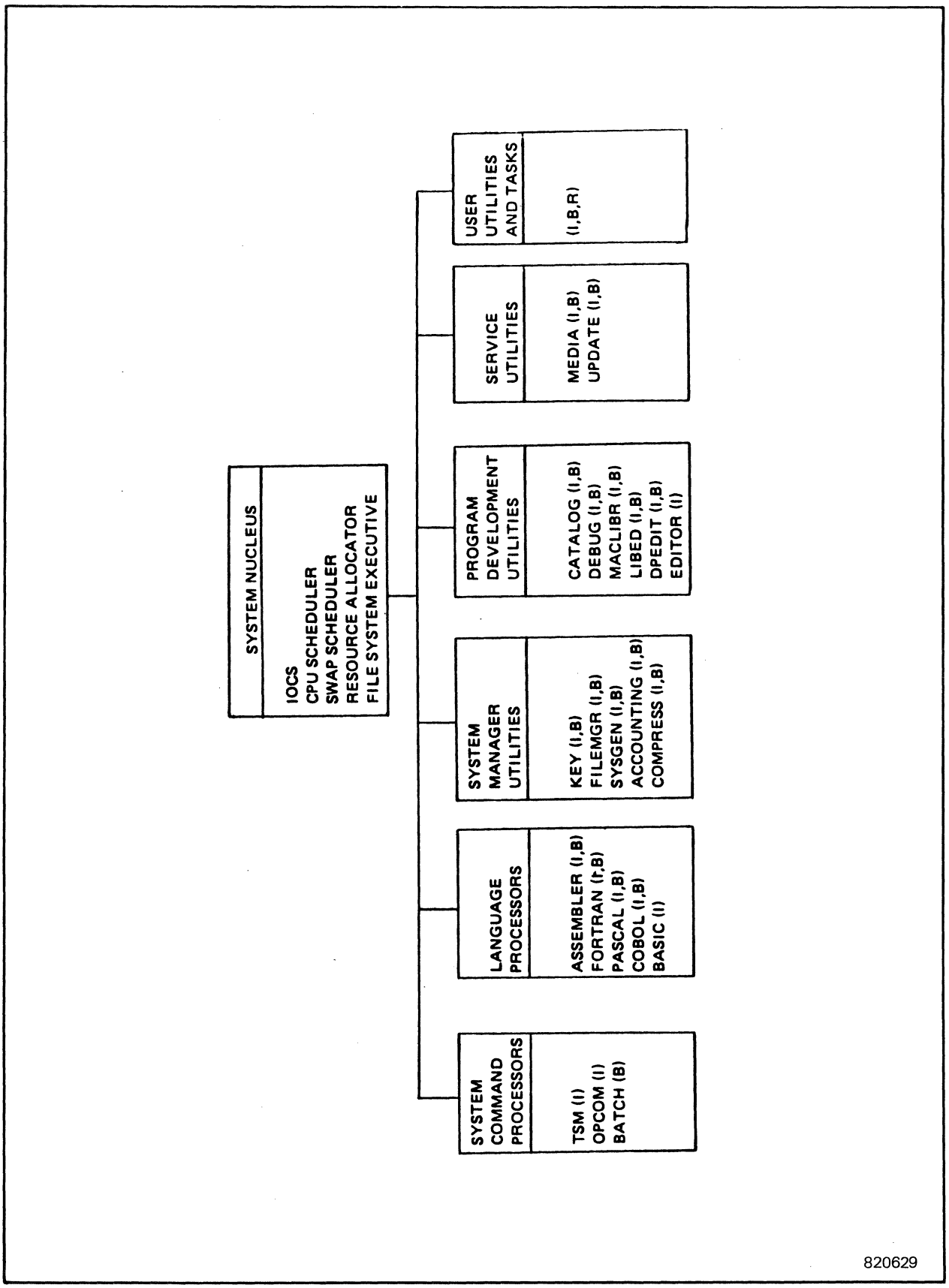


Figure 1-1. MPX-32 Processors and Utilities

Therefore, the IPU and CPU are designed to manage task execution transparently around the IPU limitations. For example, if the IPU is executing a task and encounters a service it cannot perform, a trap is sent to the CPU, the CPU takes over execution of the task at that point, and the task remains in the CPU until completion or it is reselected for IPU execution.

MPX-32 standard features include:

- Support for the full 16MB addressability of the 32 Series.
- Up to 255 tasks executing concurrently.
- 64 software priority levels, 10 of which are time distributed.
- Servicing of all standard peripheral devices including XIO controllers. I/O buffering for 32/5x series controllers.
- Standard handlers for interrupts and traps.
- Intertask communications, including send-receive.
- Intertask shared memory partitions, e.g., GLOBAL common and DATAPOOL.
- Dynamic allocation and deallocation of memory and peripherals.
- Multiple batchstreams including multiple spooled input and output queues.
- Wait and no-wait I/O capabilities including automatic blocking, buffering, and queueing.
- Terminal support for up to 64 devices, including device independent operation and an extensive repertoire of online commands.
- Automatic task reentrancy through separation of pure code and data areas.
- Reentrant system services available to all tasks.
- Several levels of system security including access restrictions based on task ownership.
- File management, assignment, and security.
- Up to 255 logical files (files or devices) opened concurrently per task.
- Project accounting capability.
- Transparent support of the IPU.

MPX-32 supports the 32/7x computers and the CONCEPT/32 computers, i.e., the SYSTEMS 32/27 and SYSTEMS 32/87.

The following is an outline identifying the major differences in operating MPX-32 on SYSTEMS 32/7x and CONCEPT/32 computers:

32/7x	CONCEPT/32
8KW map block	2KW map block
16 protection granules per map block	4 protection granules per map block
1 megabyte logical address space	2 megabyte logical address space
Supports E-class and F-class magnetic tape	Supports F-class magnetic tape or Floppy Disc
Uses TLC console	Uses IOP console
Supports E-class or D-class GPMC	Supports D-class GPMC
Supports E-class or F-class discs	Supports F-class discs
CALM or SVC callable	SVC callable only (CALM instructions are automatically converted to their equivalent SVC type)
Supports IPU	No IPU support
Bounding of data types is not enforced; results are indeterminate	Enforces (via traps) doubleword and 8 word bounding for doubleword and file instructions

1.1 System Description

SYSTEMS' 32 Series computers operating under MPX-32 use hardware and software priorities for scheduling and executing tasks. Figure 1-2 shows the various MPX-32 software elements and the hardware and software priority levels that are assigned to each.

1.1.1 Hardware Interrupts/Traps

A SYSTEMS 32/7x computer can contain up to 112 hardware priority interrupt and trap levels. The CONCEPT/32 computers support up to 96 hardware interrupts and traps. The exact number in a particular system is dependent on the user's requirements and the number of peripheral devices in the configuration.

The highest hardware priority levels in the system are reserved for the basic system integrity interrupts and traps. These include the Power Fail-Power Up traps and System Override interrupts and traps. Lower levels are used for the I/O transfer interrupts, Memory Parity Trap, Console Interrupt, and I/O service interrupts.

The next lower group of interrupts and traps are used for exceptional conditions, Call Monitor and Supervisor Call requests, and Real-Time Clock. The exceptional conditions include Non-Present Memory Trap, Undefined Instruction Trap, Privilege Violation Trap, and Arithmetic Exception Interrupt.

All lower hardware priority levels are used for external interrupts. User tasks can be connected directly or indirectly to the external interrupts.

1.1.2 Software Interrupt System

MPX-32 provides 64 software priority levels for controlling the user's application. All system scheduling is performed by priority. Users can assign multiple tasks to any priority level and thus achieve a high level of multiprogramming versatility. The software priority levels are used by the Resource Allocator for peripheral and memory allocation, by the I/O Supervisor for the queueing of I/O requests, and by MPX-32 whenever CPU control is allocated.

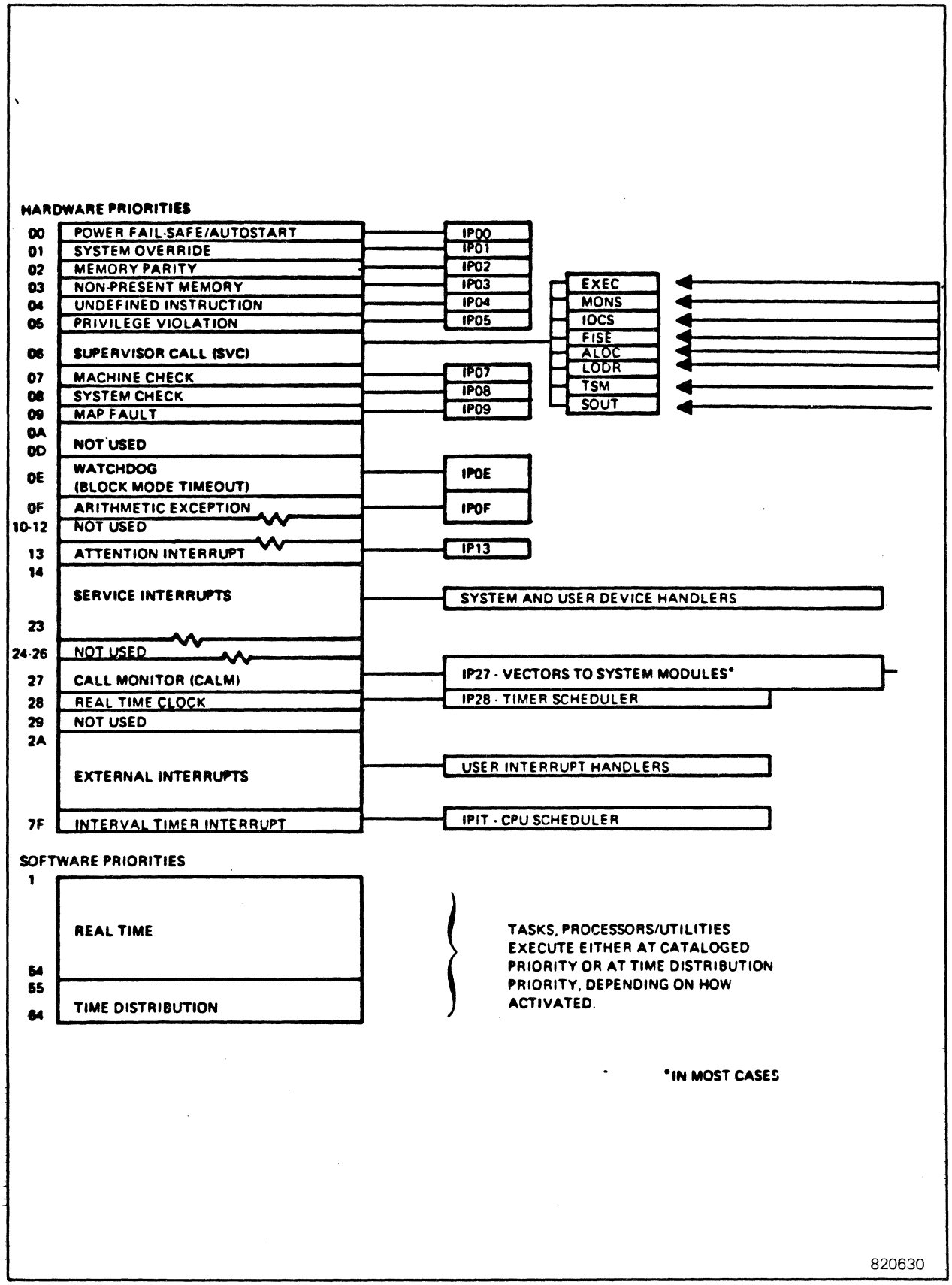


Figure 1-2 Hardware/Software Priorities

Table 1-1
CONCEPT/32 Trap Vectors

Relative Priority	Default Trap Vector Location (TVL)	Trap Condition
00	80	Power Fail Trap (Power Down)
01	84	Auto Start Trap (Power Up)
02	88	Memory Parity Trap
03	8C	Nonpresent Memory Trap
04	90	Undefined Instruction Trap
05	94	Privilege Violation Trap
06	98	Supervisor Call Trap
07	9C	Machine Check Trap
08	A0	System Check Trap
09	A4	MAP Fault Trap
0A	A8	Call Monitor Trap
0B	AC	Not Used (Reserved for IPU)
0C	B0	Address Specification Trap
0D	B4	Console Attention Trap
0E	B8	Privilege Mode Halt Trap
0F	BC	Arithmetic Exception Trap
10	C0	Cache Memory Parity Trap (32/87 only)

Table 1-2
CONCEPT/32 Interrupt Vectors

Relative Priority	Default Interrupt Vector Location (IVL)	Default IOCD Address	Default TCW Address	Interrupt Condition
00	100			External/Software Interrupt 0
01	104			External/Software Interrupt 1
02	108			External/Software Interrupt 2
03	10C			External/Software Interrupt 3
04	110	700	704	I/O Channel 0 Interrupt
05	114	708	70C	I/O Channel 1 Interrupt
06	118	710	714	I/O Channel 2 Interrupt
07	11C	718	71C	I/O Channel 3 Interrupt
08	120	720	724	I/O Channel 4 Interrupt
09	124	728	72C	I/O Channel 5 Interrupt
0A	128	730	734	I/O Channel 6 Interrupt
0B	12C	738	73C	I/O Channel 7 Interrupt
0C	130	740	744	I/O Channel 8 Interrupt
0D	134	748	74C	I/O Channel 9 Interrupt
0E	138	750	754	I/O Channel A Interrupt
0F	13C	758	75C	I/O Channel B Interrupt
10	140	760	764	I/O Channel C Interrupt
11	144	768	76C	I/O Channel D Interrupt
12	148	770	774	I/O Channel E Interrupt
13	14C	778	77C	I/O Channel F Interrupt
14	150			External/Software Interrrupts
15	154			
16	158			
17	15C			External/Software Interrupts
18	160			Real-Time Clock Interrupt
19	164			External/Software Interrupts
↕	↕			↕
5E	278			External/Software Interrupts
5F	27C			Interval Timer Interrupt

1.1.3 Task Priority Levels

Priorities 55-64 are time-sliced to provide for round-robin time distribution among tasks of the same priority. Priorities 1-54 are not time distributed. A task's cataloged priority will be altered based on its eligibility to run. For example a task's priority is boosted when an I/O operation is completed and restored after a minimal time quantum. Priority migration ensures maximum response to real time events.

1.1.4 Supervision and Allocation

CPU scheduling is maintained through a set of state queues including the priority state chains and such execution states as suspended, queued for memory, queued for peripheral, I/O wait, etc. Each CPU dispatch queue entry defines all scheduling attributes of a single task. The entry typically migrates among the state queues as the task's execution eligibility changes. These state chains are also used by the swap scheduler to select candidates for swapping.

The CPU scheduler is invoked whenever a scheduling event occurs. Scheduling events include:

- o Occurrence of an external interrupt
- o The completion of an I/O operation
- o The expiration of a timer
- o The deallocation of a resource
- o The completion of a system service

IPU scheduling is maintained through state queues consisting of biased tasks (C.RIPU) scheduled in addition to MPX-32 normal state queues for nonbiased tasks (SQRT thru SQ64). Biased tasks, if any, are prioritized among themselves and are scheduled for execution before any nonbiased tasks. Nonbiased tasks, if any, are also prioritized among themselves and are scheduled for execution after all biased tasks have been completed. If a nonbiased task waiting for execution has a higher priority level than a biased task also waiting for execution, the nonbiased task is still not executed until after all waiting biased tasks have been completed.

1.1.5 Memory Allocation

The unit of memory allocation is a map block, which is 8KW on the 32/7x computer and 2KW on the CONCEPT/32 computers. Memory is allocated to tasks as needed. All tasks are loaded discontiguously into a whole number of physical map blocks, utilizing the SelMAP to create their contiguous logical address space. No partial map blocks are allocated.

The MPX-32 memory allocation scheme allows the user tasks to dynamically expand and contract their address space via system service calls.

The unit of memory protection is called a protection granule and is 512W. Thus it is possible to have protected data areas within a map block.

1.1.5.1 Dynamic Allocation

Dynamic allocation and deallocation are performed via the allocate and deallocate system services. With these services, the user can dynamically allocate and deallocate any peripheral device, permanent and temporary disc files, or the System Listed Output (SLO) and System Binary Output files (SBO). By allocating peripheral devices dynamically, the user can ensure that each task will have exclusive use of a peripheral only during the time required to perform the task's I/O. Therefore, when peripherals are unallocated, other tasks can use them on an as-needed basis.

Because the allocation of system-wide peripheral devices that are requested dynamically cannot be guaranteed, the user task must be prepared to accept a denial return.

A task requesting additional memory is automatically queued until the memory can be allocated. For peripherals and file space, the caller can optionally queue for allocation or take alternative action.

1.1.6 File Management

In the MPX-32 operating environment, files are used in several ways. Permanent files are created for user programs, user data, and system programs. Temporary files provide system scratch storage, user scratch storage, and system output data storage for the system printer and card punch. Separation is maintained among files belonging to different users.

The file management system for MPX-32 consists of the resident File System Executive and the non-resident File Manager. Together they supervise all file space on the discs.

1.1.6.1 Permanent Files

Residing in disc storage, permanent files are defined as system or user files by entries in the System Master Directory (SMD) which specifies each file's name, device address, size, type and system flags. The permanency of these files stems from the fact that all SMD entries are stored on the disc and may be deleted only when specifically directed by the user or when the user initiates a cold start.

All permanent files are referenced by name, and any number of tasks may access any permanent file for both input and output. To locate the SMD entry for each file, MPX-32 employs a hashing technique which translates the characters in the file name to a specific location in the SMD.

Permanent files are classified as either fast or slow depending on the speed at which their SMD entries can be located. A fast permanent file is one whose entry can be located with one disc access. Slow permanent files are not necessarily characterized by a unique mapping of names in the SMD, and therefore, two or more disc accesses may occasionally be required to find each file's entry.

1.1.6.2 Temporary Files

Temporary files are files whose definitions are eliminated from the system upon completion of the task requiring the space. Temporary file space is allocated and deallocated by the File System Executive, which is responsible for maintaining space allocation maps for all available discs. Temporary files are typically used for system scratch storage and user scratch storage.

1.1.6.3 Random Access Files

Any disc file may be accessed randomly by record number through standard IOCS calls. The user sets a bit in a File Control Block (FCB) and specifies the relative disc block number (also in the FCB) to utilize this feature.

1.1.6.4 Disc File Protection

File protection mechanisms are available to prevent unauthorized access to and deletion of permanent files. Protection of individual files may be specified when the files are created, and is based on a one- to eight-character password. In addition, read and write access to a file may be restricted. User files can also be protected on a per user basis. If a key is associated with an ownername/username in the M.KEY file, it must be specified before any access to the user's files is permitted.

1.1.6.5 Dedicated System Files

To increase system thruput, and to minimize I/O delay time, IOCS supports disc buffered I/O in conjunction with the use of special system files. Four dedicated file codes exist in the system. One file code is for buffered system input (SYC), two file codes exist for buffered system output (SLO, SBO), and one file code is reserved for a System Object file (SGO). A user may assign a system file to a file code in the same manner that he would assign a device to a file code.

1.1.7 System Services

MPX-32 offers an extensive array of resident system service routines designed to perform frequently required operations with maximum efficiency. Using the CALM or the Supervisor Call instruction, tasks running in batch, interactive, or real time environments can call these routines.

All system service routines are reentrant. Thus, each service routine is always available to the currently active task.

The system service routines are provided as standard modular components of MPX-32. The "open-ended" design of the system, however, gives each user freedom to add any service routines required to tailor MPX-32 to his specific application.

1.1.8 Input/Output Operations

The Input/Output Control System (IOCS) provides I/O services that relieve the programmer of detailed chores. While keeping software overhead to an absolute minimum, IOCS receives and processes all I/O requests for both user and tasks. It performs all logical error checking and parameter validation. IOCS also logically processes all I/O operations and assigns I/O control to the appropriate device handler. The device handler, in turn, executes the I/O data exchange, processes service interrupts, and performs device testing.

Input/Output operations under MPX-32 include the following general capabilities: direct I/O, queued I/O requests, device independent I/O, device interchangeability, device reassignment, and disc-buffered (blocked) I/O.

1.1.8.1 Direct I/O

Should the user wish to acquire data at rates which prohibit the overhead of IOCS, he can issue I/O directly. Mechanisms are provided in IOCS to ensure that no conflict occurs with IOCS file operations. The interface facilities provided in IOCS for direct I/O enable a task to gain exclusive use of an I/O channel.

1.1.8.2 Device Independent I/O

Normal I/O operations in the system occur to and from user specified logical file codes. These file codes are assigned and reassigned to the physical device to which the I/O commands are ultimately routed.

1.1.8.3 Logical File Codes

The user logical file code consists of from one to three ASCII characters. For each file code defined and referenced by a user task, there is an entry in a File Assignment Table (FAT). The FAT entry describes the device controller channel and the device to which the file is assigned. In the case of a disc, which is a shared device, additional addressing information is provided for complete identification of the file. Each user task is allowed a maximum of 255 logical file assignments.

1.1.8.4 File Access

Both random and sequential file access is supported by IOCS. Random or sequential access is specified by the user. All files assigned to devices other than disc are considered sequential. A file assigned to disc may be referenced by both random and sequential transfers. A parameter specifying either read-only or read-write status must accompany the open request. Attempts to perform a write operation on a file specified as read-only, or attempts to circumvent disc file protection and security are aborted.

1.1.9 Communications Facilities

MPX-32 offers complete facilities for conducting communications between individual users, between internal system elements, between user tasks, and between the operator and the system. Users communicate with one another through sharing permanent files, Global Common and DATAPOOL partitions, and job status flags which can be set and interrogated by system service routines. Tasks communicate with one another via messages or run requests.

1.1.9.1 Intertask Messages

Tasks can establish message receivers for intertask communication. Messages are buffered by MPX-32 in memory pool until the receiving task is eligible to receive. The receiving task is interrupted asynchronously and optionally responds to the sender. The sender optionally waits for a reply or elects to be interrupted asynchronously by a response. Messages can be queued to an arbitrary depth.

1.1.9.2 Run Requests

A task can send a run request to any other task. A run request is similar to a message, except that, with a run request, the receiver may not yet be in execution. In such cases, the receiving task is activated before the message is queued. The receiving task can process run requests at any time.

1.1.9.3 Global Common

Global Common is an area of memory that many programs can access by using symbolic names to identify specific storage cells. In this respect, Global Common is comparable to local common. Unlike local common, however, access to Global Common is not restricted to programs within a single task. Rather, programs belonging to many independent tasks can freely access the same data and exchange control information within the Global Common area.

1.1.9.4 Datapool

Like Global Common, DATAPOOL is an area of memory that many tasks can access using symbolic references. In addition to providing all the advantages of Global Common, DATAPOOL provides a much wider range of structuring flexibility. For example, where Global Common symbolic references must follow the same order as the locations of the data in memory, symbolic references to the DATAPOOL may be entirely independent of the actual positioning of data within the memory area.

1.1.9.5 Internal Communications

Internal system elements communicate through temporary files, system queues, and the system communications region. The system communications region occupies approximately 2KW of lower memory. It contains information common to all system modules and processors.

1.1.10 Trap Processors

Trap processors are entered when any exceptional condition trap occurs. Certain traps indicate task errors, such as a reference to non-present memory, a privilege violation, or execution of an unimplemented instruction. These traps cause the violating task to be aborted. When the arithmetic exception trap occurs, the overflow condition is noted for use by the task in execution.

1.1.11 Timer Scheduler

The timer scheduler schedules events such as task activation, task resumption, flag setting and resetting, and interrupt activation on a timed basis.

1.2 System Command Processors

The Terminal Services Manager (TSM), the interactive OPCOM command processor, and the Job Control processor provide the user with access to MPX-32 interactive, batch, and real time processing environments.

1.2.1 Terminal Services Manager (TSM)

The MPX-32 Terminal Services Manager (TSM) provides interactive, timeshared access to the MPX-32 system for terminals connected either through TLC, ADS, ALIM, or 8-Line Asynchronous controllers. It is an integral part of the MPX-32 operating system and allows the terminal user to:

- o log on to MPX-32
- o access any MPX-32 processor

- o run tasks designed to run in any MPX-32 environment (batch, real time, or interactive) in the interactive environment
- o access other environments in the system, e.g., to activate a task at its base (normally real time) priority in the real time environment or submit a job to run in the batch environment
- o return to the interactive environment on exit from another processor
- o log off MPX-32

1.2.2 Operator Communications (OPCOM)

MPX-32 provides a comprehensive set of commands that can be used to interrogate the system and tune it for optimum response to changing conditions.

The commands allow the user to perform the following functions:

- o List the status of all queues, tasks, and I/O controllers
- o Control spooled print and punch output
- o Hold and continue execution of tasks
- o Activate and abort tasks
- o Connect tasks to interrupts
- o Establish resident and non-resident tasks
- o Set and interrogate time-of-day clock
- o Create and delete timer scheduler queue entries
- o Delete allocation queue entries
- o Enable, disable, and trigger hardware interrupts
- o Reserve devices, release them, and place them offline or online
- o Change the assignment of the system input device, the SGO file, and the destination of the SLO and SBO spooled output files
- o Initiate the reading of the batch stream
- o Issue system debugging commands

1.2.3 Batch Processing

Batch processing consists of spooling batch jobs to disc, interpreting job control statements, and directing listed and punched spooled output to destination files and devices. Multiple jobs are processed concurrently within limits established by SYSGEN and the availability of computer resources. Tasks comprising batch processing compete with each other and with nonbatch tasks for computer resources under standard MPX-32 allocation algorithms.

Each job is spooled to a separate System Control (SYC) disc file prior to processing. Jobs may be spooled to SYC files from card, magnetic tape, and paper tape peripheral devices, and from blocked, temporary and permanent disc files. The OPCOM BATCH command may be used to initiate spooling from peripheral devices and permanent files. The Submit Job From Disc File system service (M.CDJS) is used by TSM and EDITOR and can be invoked by a user task to initiate spooling from permanent and temporary disc files.

Job sequence numbers reflect the order that jobs are entered and uniquely identify each job and its tasks.

Upon job completion, a job's spooled listed and punched output is automatically routed to usable peripheral devices if no particular device(s) or permanent file(s) are specified for the job. Usable devices for automatic selection are specified via SYSGEN and OPCOM commands. Spooled output destination devices include line printer, card punches, magnetic tape, and paper tape. Spooled output is selected for processing based on the software priority of jobs and, within a given priority, on the order in which jobs complete processing.

1.3 Program Development Utilities

1.3.1 Task Cataloging (CATALOG)

By exercising the facilities of the Cataloger, users create permanent load modules that will execute as tasks on the MPX-32 system. During cataloging, relocatable object modules produced by the Assembler or compilers are loaded and linked internally and externally to library subroutines. The linked body of code thus produced is then sent to a selected permanent file in relocatable or absolute format. In addition, the Cataloger places a preamble on this file. This preamble contains a summary of the resources required by the task, such as memory, permanent files, and peripheral devices and defines special task characteristics (shared, resident, etc.). Once created, a task is known to the system by the name of the permanent file in which it resides. The task can then be activated, saved, restored, or otherwise operated on by specifying its name in the appropriate job control statement, system service call, or terminal directive.

1.3.1.1 Privilege

Whether a task is privileged/unprivileged can be defined along with the task. (The ability to specify privileged operation for a task can be restricted by ownername.)

By specifying whether tasks are privileged or unprivileged, users can control system security. Tasks designated to run privileged are free to execute any instruction in the instruction repertoire. They also have read/write access to all memory locations.

1.3.1.2 Overlays

For efficient use of memory, the Cataloger provides the user with facilities for dividing large programs into overlays. Both the main program segment (the root), and the overlay segments, can be cataloged in relocatable format. Individual overlays can be cataloged separately, permitting the user to modify or replace any overlay without disturbing any of the others. Flexible symbol linkage is provided between the root and its associated overlays and between individual overlays of various levels.

1.3.2 Debugger (DEBUG)

The MPX-32 Debugger is a command-oriented processor used to debug a single, cataloged user task. It can be accessed with a DEBUG command in TSM, with a $DEBUG statement in batch, by coding a M.DEBUG service call within the cataloged task, or by using the Break key after a task has been activated via TSM, in which case TSM provides the option of calling M.DEBUG.

DEBUG commands allow the user to:

- o trace task execution
- o set debugging traps within the task
- o display and/or alter contents of the task's logical address space, general purpose registers, etc.
- o watch for privileged task entry into the operating system or other areas of memory not usually accessed even by a privileged task
- o perform other operations that facilitate task debugging

1.3.3 Macro Library Editor (MACLIBR)

With the Macro Library Editor, macros that are used frequently can be placed in a macro library where they will be available for use by the Macro Assembler. During execution, the Macro Library Editor simply transfers the macros from the Source Input File to the Macro Library File. The macros entered into the library are listed on an output file.

1.3.4 Subroutine Library Editor (LIBED)

The Subroutine Library Editor provides facilities for creating and modifying the System Subroutine Library and any number of user subroutine libraries. The user is provided with a listing of directives, module names, external definitions, the quantity of library and directory space remaining on the disc, and the modules that were specified for deletion but were not located in the library.

1.3.5 DATAPOOL Editor (DPEDIT)

The DATAPOOL Editor provides the ability to create and maintain dictionaries for access to static or dynamic DATAPOOL common memory partitions.

1.3.6 Text Editor (EDIT)

The MPX-32 Text Editor provides a comprehensive set of commands for building and editing text files, merging files or parts of files into one file space, copying existing text from one location to another, and in general for performing editing functions familiar to users of interactive systems.

EDIT is typically used to create source files and to build job control files and general text files. A job file built in the Editor can be copied directly into the batchstream using the Editor BATCH command.

1.4 Service Utilities

1.4.1 Source Update (UPDATE)

The Source Update processor provides facilities for revising source files. It permits the user to enter new files as well as to update existing files by adding, replacing, and deleting source statements. Input can be in either standard or compressed format, and either format may be selected for the output file. The user can also elect to have Source Update produce a listing of the control stream as it generates the output file.

1.4.2 Media Conversion Processor (MEDIA)

The Media Conversion Processor performs utility functions ranging from card duplication to merging multiple media inputs into single or multiple media outputs. It provides media editing, media-to-media conversion, code conversion, media copying, and media verification. Rather than restricting the user to a fixed set of functions, the Media Conversion Processor is controlled by a language of directives very similar to FORTRAN which the user employs to program his utility functions.

1.5 System Manager Utilities

1.5.1 The M.KEY Editor (KEY)

KEY is a utility used to build an M.KEY file for the MPX-32 system. The M.KEY file specifies valid owner names/user names on the system and optionally sets, for each owner name/user name:

- a key to restrict access to the owner name during logon and to restrict access to the user name when accessing files
- OPCOM indicators restricting the owner's use of OPCOM commands
- an indicator that prevents the owner from cataloging "privileged' tasks (tasks that use privileged system services or privileged variations of unprivileged system services)
- an indicator that prevents the owner from activating tasks cataloged as privileged
- default tab settings
- default alphanumeric project names/numbers for accounting purposes

After KEY has been run, only those owners/users established in the M.KEY file are allowed to log on to the system and access files.

1.5.2 The File Manager (FILEMGR)

The MPX-32 File Manager is used to create or delete permanent disc file space, create or delete special GLOBAL partitions and/or a DATAPOOL partition (one that can be dynamically allocated in memory when required by tasks). A primary use is to provide system and user permanent file backup.

1.5.3 MPX-32 System Startup, Generation, and Installation (SYSGEN)

Under MPX-32, the user can install a SYSTEMS-supplied starter system by booting from the master System Distribution Tape (SDT). Using the starter system, which is fully operational, a user-configuration of the system can be generated via the SYSGEN utility (running either interactively or in batch). An online RESTART command is available to test user-configured systems. When a system is checked out, the user can create his own SDT using the FILEMGR SDT command.

When SYSGEN runs, system tables are constructed and linked to the resident system modules, handlers, and user-supplied resident modules and handlers as specified via SYSGEN directives. A resident system image is formed and subsequently written to a dynamically acquired permanent disc file. Concurrent with this process, a listing of directives is built and a load map of the system is generated. The load map can be saved on a system symbol table file specified by the user with the SYMTAB directive and used subsequently in patching the system.

A System Debugger can also be configured in the resident system image to assist in patching or debugging resident system code, including user interrupt and I/O handlers.

1.6 Libraries

Subroutine Libraries - Subroutine Libraries can be utilized to simplify the development of applications. Subroutines can be added, modified, or deleted. This permits one routine to be changed withough having to reassemble or recompile all of the subroutines needed for a task. Only the task must be recataloged.

Subroutines on a subroutine library can be used by programs written in various languages, including Assembly language. They are accessed as object modules when a task is cataloged. The subroutine library and directory for MPX-32 are called MPXLIB and MPXDIR, respectively. User subroutine libraries can be built and modified via the LIBED utility.

System Macro Libraries - Three macro libraries are supplied as part of the MPX-32 system. They are used only with programs written in Assembly language. The first, M.MPXMAC, should be accessed when code that uses MPX-32 system services is assembled. The second, M.MACLIB, is used when code contains RTM monitor service calls. These macro libraries provide macros containing equates for MPX-32 communication region variables. The third, M.RTMMAC, is used when developing an application on a MPX-32 system that will be run on a RTM system.

The user can expand, contract, or modify a macro library by using the MACLIBR utility.

Other - The SYSTEMS Scientific Subroutine Library is optionally available. It contains math and statistical routines for scientific and engineering applications. A user group library is also available. It is provided by and for SYSTEMS users.

1.7 Minimum Hardware Configuration for MPX-32

Hardware requirements for MPX-32 operation on a SYSTEMS 32/7x computer are as follows:

Memory-Only	Disc-Based
64KW Memory	128KW Memory
Magnetic Tape (Class E or F)	Magnetic Tape (Class E or F)
TLC Console	TLC Console
	32 MByte Extended I/O Cartridge Disc
	Line Printer

Hardware requirements for MPX-32 operation on a CONCEPT/32 computer are as follows:

Memory-Only	Disc-Based
64KW Memory	128KW Memory
Magnetic Tape (Class F) or Floppy Disc	Magnetic Tape (Class F) or IOP Floppy Disc
IOP Console	IOP Console
	32 MByte Extended I/O Cartridge Disc

The minimum configuration must also include the prerequisites that are required to support the items listed, i.e., controllers, formatters, etc.

Hardware supported by MPX-32 is listed in Table 1-3. Where appropriate, the code used to access a device is shown in parentheses. The code indicating the appropriate device, e.g., 'TY' for a terminal on an ALIM, is used when accessing devices connected via a communications link.

Table 1-3
MPX-32 Device Support

Model Number	Description
2005	Internal Processing Unit (IPU) (32/7x only)
7410	Analog Digital Interface (ADI)
8000	I/O Processor
8030	Line Printer/Floppy Disc Controller (32/27 only) (LP)
8050	XIO High Speed Tape Processor (HSTP) (XIO)
8055	Disc Processor II
8110	32mb Cartridge Module Disc
8130	80mb Disc Processor Subsystem
8140	300mb Disc Processor Subsystem
8170	Floppy Disc with Controller (FL)
8210	High Speed Tape Processor Subsystem 75 ips 9-Track 1600/6250 bpi (M9)
8211	High Speed Tape Processor Subsystem 125 ips 9-Track 1600/6250 bpi (M9)
8212	High Speed Tape Processor Subsystem 125 ips 9-Track 800/1600/6250 bpi (M9)
8310	Band Printer (300 lpm) (64 character) (LP)
8311	Band Printer (600 lpm) (64 character) (LP)
8313	Band Printer with VFU (300 lpm) (64 character) (LP)
8314	Band Printer with VFU (600 lpm) (64 character) (LP)
8510	8 Line Asynchronous Communication Controller
9005	Terminal Line Printer/Card Reader Controller (TLC) (32/7x only)
9009	10mb Cartridge Disc Controller (32/7x only)*
9010	Moving Head Disc Controller (32/7x only) (DM)
9013	Magnetic Tape Controller (Copper) (32/7x only) (MT)
9014	Fixed Head Disc Controller (32/7x only) (DF)*
9020	Low Speed Tape Processor (LSTP) (XIO)
9024	Disc Processor I (XIO)*
9103	Extended (Class D) General Purpose Multiplexer Controller (GPMC)
9104	General Purpose Multiplexer Controller (GPMC) (32/7x only)
9109	Synchronous Line Interface Module (SLIM)

*This product is no longer available but remains supported by MPX-32 in existing installations.

Table 1-3 (Cont.)

Model Number	Definition
9110	Asynchronous Line Interface Module (ALIM)
9116	Binary Synchronous Line Interface Module (BLIM)
9122	Asynchronous Data Set Interface (ADS) (32/7x only)
9131	High Speed Data Interface II (HSD II)
9132	High Speed Data Interface I (HSD I)*
9201	KSR Teletypewriter (10 cps) (CT or TY)*
9202	Teletypewriter (30 cps) (CT or TY)
9203	Alphanumeric CRT (95 character) (CT or TY)
9210	Card Reader (300 cpm) (CR)
9211	Card Reader (1000 cpm) (CR)
9223	Matrix Printer (340 cps) (LP)
9225	Line Printer (300 lpm) (64 character) (LP)*
9226	Line Printer (600 lpm) (64 character) (LP)*
9237	Line Printer (900 lpm) (64 character) (LP)
9245	Line Printer (260 lpm) (96 character) (LP)*
9246	Line Printer (436 lpm) (96 character) (LP)*
9247	Line Printer (600 lpm) (96 character) (LP)
9260	Paper Tape Reader (PT)*
9264	Paper Tape Punch/Spooler (120 cps) (PT)
9337	4 MB Fixed Head Disc Drive (DF)
9448-32	32 MB Cartridge Module Disc (XIO) (DM)
9460	Paper Tape Reader with Controller (300 cps) (PT)
9462	Paper Tape Reader/Spooler (300 cps) (PT)
9508	10 MB Master Cartridge Disc Drive (DM)
9520	80 MB Master Moving Head Disc (DM)
9521	40 MB Master Moving Head Disc (DM)
9523	300 MB Master Moving Head Disc (DM)
9561	45 ips Master Magnetic Tape Unit 9-Track (M9)

*This product is no longer available but remains supported by MPX-32 in existing installations.

Table 1-3 (Cont.)

Model Number	Definition
9563	45 ips Master Magnetic Tape Unit 9-Track (M9)
9567	Low Speed Tape Processor Subsystem 45 ips 9-Track 800 bpi (M9)
9568	Low Speed Tape Processor Subsystem 45 ips 9-Track 800/1600 bpi (M9)
9571	Low Speed Tape Processor Subsystem 75 ips 9-Track 800/1600 bpi (M9)
9577	75 ips Master Magnetic Tape Unit 9-Track (M9)
9733-5	5 MB Fixed Head Disc (XIO) (DF)

2. SYSTEM OVERVIEW

The MPX-32 operating system consists of a system nucleus and a number of associated processors and utilities. The nucleus provides overall supervision of tasks and resources as described in this section.

Note: The term 'task' is used throughout MPX-32 to describe a body of code which is scheduled for CPU time as a single entity. A load module is a task, in loadable form, stored on disc.

2.1 Task Activation Sequencing (M.ACTV, M.PTSK)

The MPX-32 Allocator performs task activation in two phases.

2.1.1 Phase 1 Activation

When a task is activated on the MPX-32 system, either via the M.ACTV service or the M.PTSK service, the MPX-32 Allocator runs on behalf of the task that issues the service call (the activating task). In many cases, the activating 'task', is TSM, OPCOM, or Job Control. Running at the priority of the activating task, the Allocator constructs a rudimentary Task Service Area (TSA) for the new task in the task's address space and a rudimentary Dispatch Queue Entry (DQE) in the communications region. Data in the prototypes include: a task number (or 'taskno'), parameters passed with the task (M.PTSK), the Load Module Information Table (LMIT), and other basic data that define the task.

Initially, the DQE for the task is unlinked from the list of free DQE's maintained by the CPU scheduler and linked to the preactivation state queue (PREA). (See Section 2.2.4.) After the prototype TSA and DQE are constructed, the DQE is unlinked from the PREA state queue and linked to the appropriate ready-to-run queue. A context is set up in the prototype TSA so that the Allocator can gain control for the second phase of activation as soon as the new task becomes the highest priority ready-to-run task on the system. There are several cases where task activation does not continue at the end of phase 1:

activation via a run request for a single-copied task that is already active

timer activation requests (M.SETT)

RTM-compatible activation on an interrupt (CALM X'66' or M.CONN)

In the first case, the CPU scheduler may link the run request to an existing DQE (see Section 2.4.5). In the last two cases, the task remains in the preactivation state queue until the timer expires or the interrupt fires. At that point, such tasks are linked to the appropriate ready-to-run queue as described previously.

2.1.2 Phase 2 Activation

In this phase, the Allocator operates on behalf of the new task, and runs at the new task's specified priority. It reads in the Resource Requirements Summary (RRS) from the load module file, merges them with static assignments, and validates the results. Resources are allocated. The task's DQE may be linked and unlinked to various state queues as it moves through stages of device and memory allocation.

If any parameters, assignments, or other task resource requirements specified in the load module or by Job Control or TSM assignments are invalid, the Allocator aborts the task during this phase and the task exits as described in Section 2.8.

When the new task has allocated all resources required for execution, it is loaded into memory, relocated, and the Allocator transfers control to the task at its cataloged transfer address.

There are two exceptions to the control transfer at the end of phase 2. The first is a task that has been initiated via the OPCOM ESTABLISH command. This task is linked into the suspended state queue (SUSP) instead of going into execution. The purpose is to provide a capability within the MPX-32 structure equivalent to the capability in RTM (Real Time Monitor) for activating a task that resides permanently in memory (a resident task). (In MPX-32, resident means 'locked in memory'.) When an activating request occurs for a task that has been established (a timer expires, an interrupt fires, or the task is resumed), the task is fully ready to execute and is brought into execution with just a context switch. If the task has been cataloged as resident, no inswap is required.

The second exception is a task that has been activated with the MPX-32 Debugger attached (TSM or job control DEBUG taskname command). Instead of transferring control to the task, the Allocator first loads and then transfers control to the Debugger.

2.1.3 Task Service Area (TSA)

The Task Service Area (TSA) is a section of memory associated with each active task. The size of each task's TSA is fixed for the duration of the task's execution. However, the sizes of TSA's among tasks is variable and is dependent on the amount of space reserved for I/O activity.

As depicted in the following figure, the number of blocking buffers, File Assignment Table (FAT) entries and the File Pointer Table (FPT) entries is variable among tasks. For all tasks, the first buffer and FAT and FPT entry are reserved for MPX-32 use; i.e., they are present in every TSA.

The pushdown stack area in the TSA provides reentrancy in calls to system modules. At each call to a system module entry point, T.REGP is incremented to the next 32-word pushdown level where the contents of the general purpose registers and Program Status Doubleword (PSD) are saved. Within this 32-word level, 22 words are available for scratchpad storage by the module entry point being called. T.REGP is decremented to the previous pushdown level upon return to the entry point caller. Upon context switch away from a task, the next pushdown level is used to preserve the contents of the task's registers and PSD. Ten words are used at the context switch level.

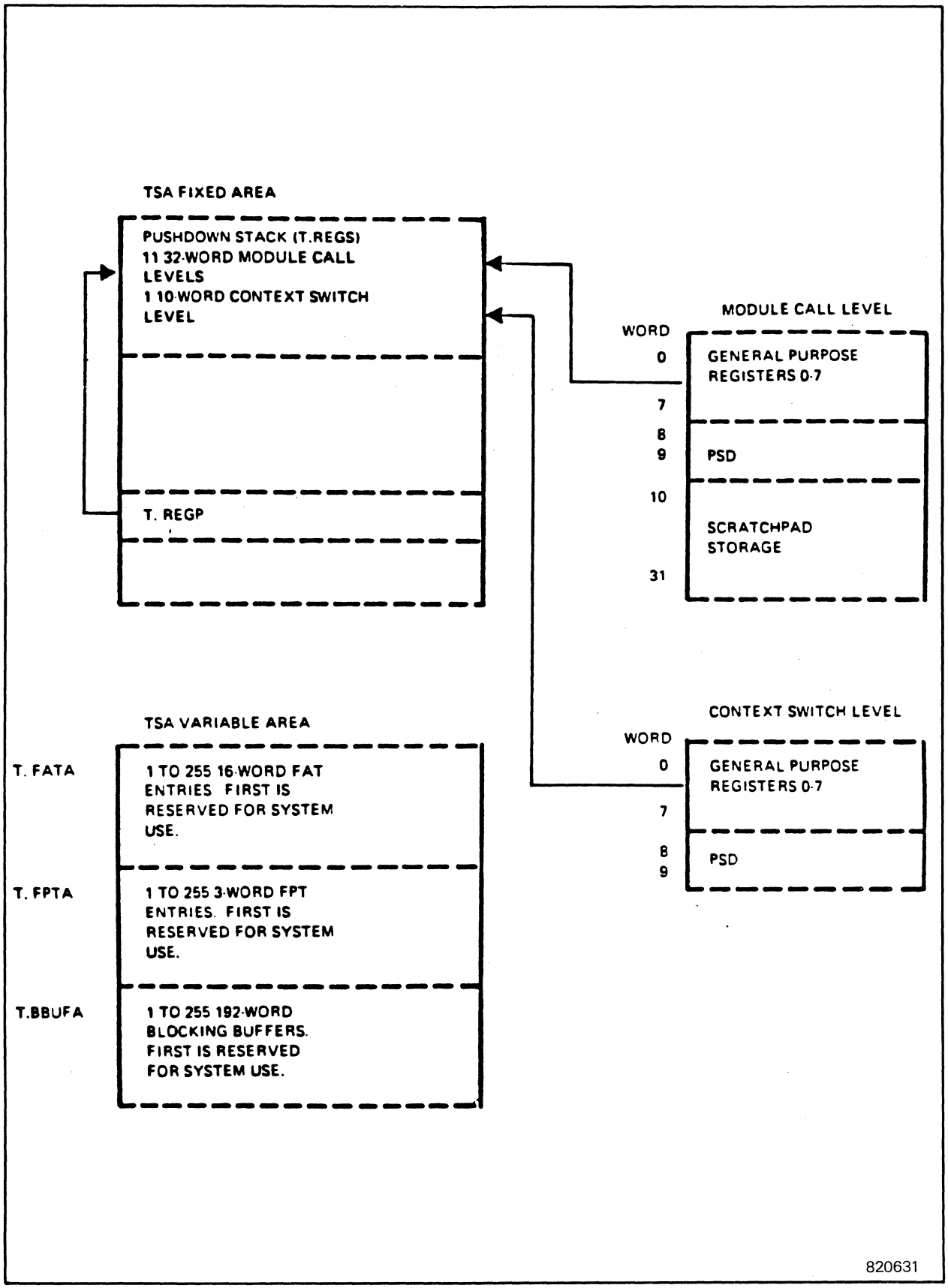


Figure 2-1
Task Service Area (TSA) Structure

2.2 MPX-32 CPU Scheduling

The MPX-32 CPU scheduler is responsible for allocating CPU execution time to active tasks. Tasks are allocated CPU time based on execution priority and execution eligibility. Execution priority is specified when a task enters (is cataloged into) the system. Execution eligibility is determined by the task's readiness to run.

2.2.1 Execution Priorities

The MPX-32 system provides 64 levels of execution priority. These priority levels are divided into two major categories. Real-time tasks operate in the priority range 1-54. Time distribution tasks operate in the priority range 55-64.

2.2.2 Real Time Priority Levels (1-54)

Scheduling of real time tasks in MPX-32 occurs on a strict priority basis. The system does not impose time-slice, priority migration, or any other scheduling algorithm which will interfere with the execution priority of a real time task. Execution of an active real time task at its specified priority level is inhibited only when it is ineligible for execution (not ready to run). Execution of a real time task may, of course, always be preempted by a higher priority real time task that is ready to run.

2.2.3 Time Distribution Priority Levels (55-64)

For tasks executing at priority levels 55-64, MPX-32 provides a full range of priority migration, situational priority increment, and time quantum control.

2.2.3.1 Priority Migration

The specified execution priority of a time distribution task is used as the task's base execution priority. Each time distribution task's current execution priority is determined by the base priority level as adjusted by any situational priority increment. The current execution priority is further adjusted by increasing the priority (by one level) whenever execution is preempted by a higher priority time distribution task, and decreasing the priority whenever the task gains CPU control. The highest priority achievable by a time distribution task is priority level 55. The lowest priority is clamped at the task's base execution level.

2.2.3.2 Situational Priority Increments

Time distribution tasks are given situational priority increments in order to increase responsiveness. The effect of situational priority increments is to give execution preference to tasks that are ready to run after having been in a natural wait state. A task that is CPU bound will migrate toward its base execution priority. Situational priority increments are invoked when a task is unlinked from a wait state list, and relinked to the ready to run list.

Situation	Priority Increment
Terminal input wait complete	Base level + 2
I/O wait complete	Base level + 2
Message (send) wait complete	Base level + 2
Run request (send) complete	Base level + 2
Memory (inswap) wait complete	Base level + 3

2.2.3.3 Time Quantum Controls

The MPX-32 system allows for the specification of two time quantum values at SYSGEN. If these values are not specified, system default values are used. The two quantum values are provided for scheduling control of time distribution tasks. The first quantum value (stage 1) indicates the minimum amount of CPU execution time guaranteed to a task before preemption by a higher priority time distribution task. The stage 1 quantum value is also used as a swap inhibit quantum after inswap. The second quantum value represents the task's full time quantum. The difference between the first and second quantum values defines the execution period called quantum stage 2. During quantum stage 2, a task may be preempted and/or outswapped by any higher priority task. When a task's full time quantum has expired, it is relinked to the bottom of the priority list, at its base execution priority.

Time quantum accumulation is the accumulated sum of actual execution times used by this task. A task's quantum accumulation value is reset when the task voluntarily relinquishes CPU control (e.g., suspends, performs wait I/O, etc.).

2.2.4 State Chain Management

The current state of a task (e.g., ready to run, waiting for I/O, etc.) is reflected by the linkage of the dispatch queue entry (DQE) associated with the task into the appropriate state chain. Linkage is established via string forward and string backward addresses and a state queue index in each DQE. The string forward address for a given DQE points to the closest lower priority DQE and the string backward address points to the closest higher priority DQE in a given state. The index points to a state chain head cell, which contains the link forward/backward addresses from the DQE at the top (highest priority task) of the state chain. At a given time, from any one DQE or from a head cell, an entire state chain queue can be examined by moving either backward or forward through the DQE linkages.

The state queues are divided into two major categories: ready to run and waiting. The ready-to-run category is subdivided by priority, with a single queue for the real time priorities and a separate queue for each of the time distribution priority levels. The "waiting" category is subdivided according to the resource or event required to make the task eligible for execution.

STATE INDEX	LABEL	MEANING	
0	FREE	DQE is available (in free list)	
1	PREA	Task activation in progress	
2	CURR	Task is executing	
3	SQRT	Task is ready to run (priority level 1-54)	Ready to run queues
4	SQ55	Task is ready to run (priority level 55)	
5	SQ56	Task is ready to run (priority level 56)	
6	SQ57	Task is ready to run (priority level 57)	
7	SQ58	Task is ready to run (priority level 58)	
8	SQ59	Task is ready to run (priority level 59)	
9	SQ60	Task is ready to run (priority level 60)	
10	SQ61	Task is ready to run (priority level 61)	
11	SQ62	Task is ready to run (priority level 62)	
12	SQ63	Task is ready to run (priority level 63)	
13	SQ64	Task is ready to run (priority level 64)	
14	SWTI	Task is waiting for terminal input	Operation wait queues
15	SWIO	Task is waiting for I/O	
16	SWSM	Task is waiting for message complete	
17	SWSR	Task is waiting for run request complete	
18	SWLO	Task is waiting for low speed output	
19	SUSP	Task is waiting for: 1) Timer expiration, or 2) Resume request, or 3) Message request interrupt.	Execution wait queues
20	RUNW	Task is waiting for: 1) Timer expiration,or 2) Run request.	
21	HOLD	Task is waiting for a continue request	

Table 2-1 MPX-32 State Queues

STATE INDEX	LABEL	MEANING	
22	ANYW	Task is waiting for: 1) Timer expiration, or 2) No-wait I/O complete, or 3) No-wait message complete, or 4) No-wait run request complete, or 5) Message request interrupt, or 6) Break interrupt	Execution wait queues (continued)
23	SWDC	Task is waiting for disc space allocation	Resource wait queues
24	SWDV	Task is waiting for device allocation	
25	SWFI	Task is waiting for file system (H.FISE)	
26	MRQ	Task is waiting for memory allocation	
27	SWMP	Task is waiting for memory pool allocation	
28	SWGQ	Task is waiting in general wait queue	
29	CIPU	Current IPU task	IPU state queues*
30	RIPU	Requesting IPU execution	

Table 2-1 MPX-32 State Queues

* See the MPX-32 Technical Manual, Chapter 9 for further details.

2.3 Internal Processing Unit (IPU) Scheduling

The IPU is a user transparent device managed by the MPX-32 operating system. The IPU is scheduled as an additional resource to offload the CPU and improve system thruput in a multi-tasking enviroment. Scheduling of tasks for IPU execution is controlled by the IPU scheduler (H.CPU) based on a task's eligibility to run. Tasks may be biased for execution in the CPU or IPU thru cataloged or runtime options. However, if no biasing is in effect, a task is directed to the first available processor. Biased tasks have processing priority over nonbiased tasks.

2.3.1 Options

IPU options available are as follows:

IPUBIAS — When set, tasks that are IPU compatible will be run by the IPU. At any point during execution where compatibility ceases, the CPU is trapped and the task is transferred to the CPU for execution.

CPUONLY — When set, the IPU is ignored and the task is executed by the CPU.

Default: Tasks are executed by the first available processor.

2.3.2 Biased Task Prioritization

If the IPU scheduler finds more than one biased task waiting for processing, they are placed in a ready to run state (C.RIPU), in priority order among themselves, and are eligible for swapping while waiting.

2.3.3 Nonbiased Task Prioritization

If the IPU scheduler finds more than one nonbiased task waiting for processing (any task in ready state queues SQRT thru SQ64). they are placed in priority order among themselves and scheduled for processing after processing of all biased tasks that may be waiting. Nonbiased tasks never get processed in the IPU before biased tasks, even though a nonbiased task may have a higher priority level than a waiting biased task.

2.3.4 IPU Task Selection and Execution

When the IPU task scheduler has found a task, it checks for IPU eligibility. For a task to be eligible for IPU execution, the following conditions must be present:

No pending task interrupts
No system action requests, e.g. aborts
Not CPU biased
Current execution address outside of resident O.S.

If a task fails any one of these tests, it is ineligible for IPU execution (i.e., ignored) and the task scheduler proceeds to select the next task, if any.

If a task has been selected and is determined eligible for IPU processing, it is linked to the current IPU task queue (C.CIPU), a start IPU (SIPU) is executed, the IPU executive (H.IPU) fields the trap, performs memory management on behalf of the task, and transfers control to the task, i.e., executes the task.

If there are no IPU compatible tasks found, the IPU remains idle.

2.3.5 CPU Execution of IPU Tasks

Unbiased tasks require CPU execution for code sequences requiring OS execution. Unbiased tasks are also free to execute task level code in the CPU.

IPU biased tasks will be executed by the CPU for only those code sequences requiring OS execution. When the PSD points back into the task, its CPU execution is terminated immediately and the task is linked to the IPU request queue (C.RIPU). If the IPU is running and this new task has a higher priority than the task the IPU is executing, the executing task is 'bumped' and replaced with the new task. If the IPU is running and the new task has a lower priority than the task currently under execution, the new task is placed in the IPU ready to run queue (C.RIPU).

2.3.6 IPU Accounting

When the IPU and its interval timer handler are specified during SYSGEN, and the IPU is used for task execution, the following message will be displayed at EOJ and when logging off a terminal:

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

where xx is a decimal number.

2.4 MPX-32 Task Interrupt Scheduling

In addition to the 64 levels of execution priority available, the MPX-32 scheduler provides a software interrupt facility within the individual task environment.

2.4.1 Task Interrupt Levels

Individual tasks operating in the MPX-32 environment may be organized to take advantage of task unique software interrupt levels. Each task in the MPX-32 system may have six levels of software interrupt, sometimes referred to as pseudo interrupts:

Level Priority	Description
0	Reserved for operating system use
1	DEBUG
2	Break
3	End Action
4	Message
5	Normal Execution - Run Request

2.4.1.1 Task Interrupt Receivers

An individual task is allowed to issue system service calls to establish interrupt receiver addresses for both break and message interrupts. The DEBUG interrupt level is used by the system to process tasks running in DEBUG mode. The end action interrupt level is used for system post processing of no-wait I/O, message, or run requests. It is also used for executing end action routines specified by the user task. The normal execution level is used for run request processing and general base level task execution.

2.4.1.2 Scheduling

Task interrupt processing is gated by the MPX-32 CPU scheduler (see Figure 1-1) during system service processing. If a task interrupt request occurs while the task is executing in a system service, the scheduler defers the interrupt until the service returns to the user task execution area. If service calls are nested, the scheduler defers the task interrupt until the last service executes and returns to the user task execution area. The user may defer task interrupts through calls to synchronize task interrupts (M.SYNCH) or disable message interrupts (M.DSMI).

2.4.1.3 System Service Calls from Task Interrupt Levels

A task may utilize the complete set of system services from any task interrupt level. It is prohibited, however, from making a wait-for-any call (M.ANYW, M.EAWAIT) from task interrupt levels.

2.4.1.4 Task Interrupt Context Storage

When a task interrupt occurs, the CPU scheduler automatically stores the interrupted context into the TSA pushdown stack. This context is automatically restored when the task exits from the active interrupt level.

2.4.1.5 Task Interrupt Level Gating

When a task interrupt occurs, the level is marked active. Additional interrupt requests for that level are queued until the level active status is reset by the appropriate system service call. When the level active status is reset, any queued request is processed.

2.4.2 User Break Interrupt Receivers (M.BRK and M.BRKXIT)

A task may enable the break interrupt level by calling the M.BRK service to establish a break interrupt receiver address. The level becomes active as a result of a break interrupt request generated either from a hardware break or from a M.INT service call which specified this task. When the break level is active, end action, message, and normal execution processing is inhibited. The level active status is reset by calling the M.BRKXIT service to exit from the pseudo interrupt (break) level.

2.4.3 User End Action Receivers (M.XMEA, M.XREA, M.XIEA)

When a task issues a no-wait I/O, message, or run request, a user task end action routine address may optionally be specified. If specified, the routine will be entered at the end action priority level from the appropriate system post processing routine. When the end action level is active, processing at the message or normal execution level is inhibited. The level active status is reset by calling the appropriate end action service:

End Action Type	End Action Exit Service
I/O	SVC 1,X'2C'
Send Message	M.XMEA
Send Run Request	M.XREA

2.4.4 User Message Receivers (M.RCVR, M.GMSGP/M.XMSGR)

A task may enable the message interrupt level by calling the M.RCVR system service to establish a message interrupt receiver address. The level becomes active as the result of a message send request specifying this task as the destination task.

When the message level is active, normal execution processing is inhibited. The task's receiver may optionally call a service M.GMSGP to store the message in a user receiver buffer. After appropriate processing, the message interrupt level may be reset by calling the M.XMSGR system service to exit from the message interrupt receiver.

2.4.5 User Run Receivers (M.GRUNP/M.XRUNR)

User run receivers execute at the normal task execution (base) level. The cataloged transfer address is used as the run receiver execution address. The run receiver mechanism is provided by the system to allow queued requests for task execution with optional parameter passing.

When a run request is issued via the M.XRUNR service, if the task is single-copied, the load module name is used to identify the task to be executed. If the task is multicopied and a run request is being issued to a task which is waiting for a run request (i.e., in the RUNW state chain), the task number must be specified. If a run request is sent to a multicopied task by load module name, another copy of that load module will be activated.

The task receiving the run request may optionally call a service M.GRUNP to store the run parameters in a user receiver buffer. After appropriate processing, the run receiver task may exit by calling the M.XRUNR system service. Any queued run requests are then processed.

2.4.6 User Abort Receivers (M.SUAR)

User abort receivers execute at the normal task execution (base) level. The user task may optionally establish an abort receiver by calling the M.SUAR service.

If an abort condition is encountered during task operation, control is transferred to the task's abort receiver. Before entry, any active software interrupt level is reset, all outstanding operations or resource waits are completed, and all no-wait requests are processed. End action routines associated with no-wait requests which complete while the abort is outstanding are not executed. Status bits reflecting this are posted in the appropriate FCB's. Any files opened or resources allocated at the time the abort condition is encountered remain opened and/or allocated when the abort receiver is executed.

The TSA stack is clean. The context at the time the abort condition is encountered is stored in T.CONTXT. When the abort receiver is entered, R5 reflects task interrupt status at the time the abort condition was encountered:

Bit	Meaning When Set
16-18	N/A
19	User Break Interrupt Active
20	End Action Interrupt Active
21	Message Interrupt Active
22-31	N/A

The standard exit service described in Section 2.8.1 is used to exit from a task's abort receiver. If another abort condition is encountered while a task is executing an abort receiver, the task is deleted.

A privileged task may re-establish its abort receiver through the M.SUAR service. An unprivileged task is not allowed to re-establish its abort receiver after an abort condition has been encountered. An attempt to do so will result in a task delete.

2.4.7 Task Interrupt Services Summary

Table 2-2 summarizes the services described in this section including required parameter blocks. For a detail description of the parameter blocks for run and message requests, see Section 3.4.

Table 2-2 Task Interrupt Operation/Services Summary

	TASK INTERRUPT PRIORITY	Level 5		Level 4		Level 3		Level 2
	TASK INTERRUPT FUNCTIONS	Run Requests	Abort Requests	Message Requests	End Action - Run	End Action - Msg	End Action - I/O	Break Requests
	Issue Request	M.SRUNR	M.BORT OPCOM Abort	M.SMSGR OPCOM Send	MPX	MPX	MPX	Hardware Break OPCOM Break M.INT
Sending Task Functions	Send Block	PSB	N/A	PSB	N/A	N/A	N/A	N/A
	Wait for Completion (wait)	PSB	N/A	PSB	N/A	N/A	N/A	N/A
	Establish End Action Receiver	PSB	N/A	PSB	N/A	N/A	N/A	N/A
	Wait for Completion (no-wait)	M.ANYW	N/A	M.ANYW	N/A	N/A	N/A	N/A
	Call-Back Information	PSB	N/A	PSB	N/A	N/A	N/A	N/A
Receiving Task Functions	Establish Receiver	N/A	M.SUAR	M.RCVR	PSB	PSB	FCB	M.BRK
	Get Parameters	M.GRUNP	N/A	M.GMSGP	PSB	PSB	R1 Points to FCB	N/A
	Receive Block	PRB	N/A	PRB	PSB	PSB	FCB	TY's UDT or Contents of T.BREAK
	Exit Receiver	M.EXIT M.XRUNR	M.EXIT	M.XMSGR	M.XREA	M.XMEA	SVC 1,X'2C'	M.BRKXIT M.XBRKR
	Exit Block	RXB	N/A	RXB	N/A	N/A	N/A	N/A
	Wait for Next Request	RXB (if M.XRUNR)	N/A	M.SUSP M.ANYW	N/A	N/A	N/A	M.ANYW
	Disable Interrupt Level	N/A	N/A	M.DSMI	N/A	N/A	N/A	N/A
	Enable Interrupt Level	N/A	N/A	M.ENMI	N/A	N/A	N/A	N/A

2.5 CPU Dispatch Queue Area

The CPU Dispatch Queue is a variable length table built at SYSGEN and contains a maximum of 255 Dispatch Queue Entries (DQE's). Free DQE entries are linked into the C.FREE head cell in the standard linked list format. When a task is activated, a DQE is obtained from the free list and is used to contain all of the memory-resident information necessary to describe the task to the system.

For example, the task sequence number, owner name, load module name, TSA address, priority, and current state chain pointers are kept in the DQE, as are abort codes, message and run receiver queue addresses, etc.

Additional (swappable) information is maintained in the Task Service Area (TSA). While a task is active, its DQE is linked to one of the various ready-to-run or wait state chains provided by the CPU scheduler to describe the task's current status. When a task exits, its DQE is again linked to the free list.

2.6 I/O Scheduling

I/O scheduling is designed to provide efficient service to I/O bound tasks while keeping the CPU busy with compute-bound tasks. This allows the fullest possible utilization of both the CPU and I/O devices.

A task that has been waiting for I/O to complete (SWTI or SWIO) is changed to an executable state at a priority slightly higher than a similar compute-bound task when the I/O completes as described in Section 2.2.3.2. At that time, the CPU scheduler interrupts the execution of the compute-bound task so that the I/O-bound task can execute. The I/O-bound user requires comparably little CPU time before initiating another I/O request and returning to the SWTI or SWIO state. The compute-bound task then resumes execution. The CPU scheduler automatically adapts to tasks that alternate between bursts of computing and bursts of I/O.

2.7 Swap Scheduling

The Swap Scheduler task processes entries in the Memory Request Queue (MRQ). It provides memory allocation and swap scheduling as appropriate, to service individual requests for memory.

2.7.1 Structure

The Swap Scheduler is a resident, privileged task which resides in low memory. It is mapped into the address space of every task in the system. It has its own minimal TSA and DQE, and executes at the priority of the highest priority task in the Memory Request Queue.

The Swapper remains suspended until resumed by the executive in response to a swap scheduler event.

2.7.2 Entry Conditions

The Swap scheduler task is normally suspended. It is relinked to the ready-to-run queue by the executive in response to a system service calling the executive to report a swap scheduler event. There are four basic types of swap scheduler events.

2.7.2.1 Dynamic Expansion of Address Space (M.GE/M.GD)

Whenever there is insufficient memory to satisfy a dynamic memory request on behalf of a task, the task is linked into the memory request queue and the Swap scheduler is resumed.

Memory is allocated in 2KW increments on a CONCEPT/32 and in 8KW increments on a SYSTEMS 32/7x computer.

2.7.2.2 Deallocation of Memory (M.FE/M.FD)

Whenever a task deallocates some or all of its memory, and the memory request queue is not empty, the Swap scheduler is resumed.

2.7.2.3 Request for Inswap

Whenever a currently outswapped task becomes eligible for execution, it is linked into the memory request queue. The Swap scheduler is resumed to process the inswap request.

2.7.2.4 Change in Task Status

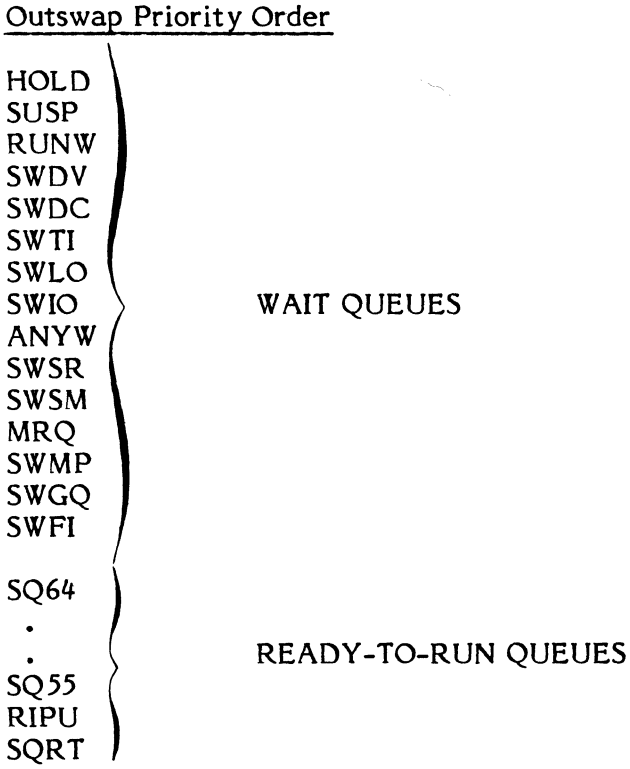
Whenever a task which had been previously ineligible for swapping becomes eligible, the Swap scheduler is resumed. Such status changes include the completion of an unbuffered I/O operation, the release of a lock-in-memory flag, or the expiration of a stage one time quantum.

2.7.3 Exit Conditions

The Swap scheduler signals the executive when it cannot process any more outstanding requests, or when the memory request queue is empty. The Swap scheduler is unlinked from the ready-to-run queue and placed in a special wait-for-memory-event state.

2.7.4 Selection of Inswap and Outswap Candidates

The Swap scheduler initially attempts to allocate the memory required for the highest priority task in the memory request queue. If there is insufficient free memory, the Swap scheduler examines the state queues on a priority basis (as indicated next), searching for the memory class and number of map blocks required.



The DQE address of the first outswap candidate satisfying any of the current memory request is determined, and the task is then outswapped completely before the Swap scheduler reexamines the memory request queue.

Whenever sufficient memory is available for inswap, the inswap process is initiated. The Swap scheduler continues to process entries in the memory request queue until the queue is empty or until it cannot find an available outswap candidate for a task requesting memory. Both outswap and inswap are serial processes which go to completion before the memory request queue is reexamined. Dynamic memory requests are similar to inswap requests, except that there is no associated disc file to read. Some tasks in the memory request queue may be queued for both inswap and a dynamic request. In such cases, both requests must be satisfied before the inswap process can proceed.

2.7.5 Outswap Process

Outswap is a demand process which is initiated in response to an inswap or dynamic memory request. The TSA of the outswap candidate is mapped into the Swap scheduler and used to construct a new address space which represents the swappable map blocks in a logically contiguous format. Then the swap file is allocated and opened by the Swap scheduler. For F class swap devices, a single write request is given to IOCS. Command and data chains are built in the handler to perform the specified transfer. For E class swap devices, the swap scheduler issues multiple 4KW writes to IOCS from its internal buffer until all swappable map blocks have been written. The Swap scheduler issues no-wait I/O requests, and performs its own double buffering of H and S map blocks through its dedicated E class mapblock.

Once output is complete, the memory is deallocated, and the memory request queue is reexamined to find the highest priority candidate for inswap.

2.7.6 Inswap Process

Once sufficient memory is available, the Swap scheduler allocates the memory required by the highest priority task in the memory request queue. If the request is simply a dynamic one, the Swap scheduler adjusts the TSA of the requestor to reflect the newly allocated memory, and the CPU scheduler is informed.

If the request requires an inswap, the Swap scheduler allocates and opens the Swap file and reads the swapped image into the newly allocated memory. For F class swap devices, a single READ request is given to IOCS. Command and data chains are built in the handler to perform the specified transfer. For E class swap devices, the swap scheduler issues multiple 4KW reads to IOCS until all swapped map blocks have been read. The Swap scheduler issues no-wait I/O requests, and performs its own double buffering of H and S map blocks through its dedicated E class map block.

Once inswap is complete, the Swap scheduler cleans up its map and reexamines the memory request queue for the next inswap candidate.

2.8 Task Termination Sequencing

Three types of task termination are provided by the MPX-32 executive: exit, abort, and delete task execution.

2.8.1 Exit Task (M.EXIT)

The exit task service is called by a task that wishes to terminate its execution in a normal fashion. The sequence of system processing on task exit is described in Table 2-3.

2.8.2 Abort Task (M.BORT)

The abort task service is called by a task that wants to terminate its execution in an abnormal fashion. It may also be initiated by the system when a task encounters a system trap condition (e.g., undefined instruction, privilege violation, or non-present memory); or by a system service because of a parameter validation error. This service may also be asynchronously initiated by another task of the same ownername or by the OPCOM ABORT command. The sequence of system processing on task abort is described in Table 2-3.

2.8.3 Delete Task (M.DELTSK)

The delete task service is called by the system on behalf of a task that encounters a second abort condition when processing an initial abort request. This service may also be initiated asynchronously by another task of the same ownername or by the OPCOM KILL command. The sequence of system processing on task delete is reflected in Table 2-3.

Table 2-3 Task Termination Sequencing
(EXIT, ABORT, and DELETE)

Task Has	System Action		
	Task Exit	Task Abort	Task Delete
Outstanding I/O	Defers processing until any outstanding I/O is complete. Aborts task if encounters user no-wait I/O end action routine.	Same as exit, except inhibits execution of user no-wait I/O end-action routines. Task abort is reflected in appropriate FCB(s).	Terminates all outstanding I/O.
Outstanding Messages in Receiver Queue	Unlinks all outstanding messages; posts complete with abnormal status.	Same as exit.	Same as exit.
Outstanding No-wait Run Requests with Call Back	Defers exit processing until all requests are complete. Inhibits execution of end action routines. End action not processed status is reflected in run request parameter block.	Defers abort processing until all requests are complete. Inhibits execution of end-action routines. Task abort status is reflected in run request parameter block.	Call backs are ignored.
Run Requests in Receiver Queue	Terminates the current run request and posts appropriate status in run request parameter block. Then activates a new copy of the task for next run request in queue, if any.	Same as exit.	Same as exit.

End Table 2-3 Task Termination Sequencing
(EXIT, ABORT, and DELETE)

Task Has	System Action		
	Task Exit	Task Abort	Task Delete
Task Abort Receiver	Not processed.	Transfers control to task after other steps taken above. Files are not closed; devices and memory are not deallocated. (Remaining abort processing by system is discontinued.)	Not processed.
Files Open	Closes all open files automatically. Preserves integrity of both user and system files.	Same as exit.	Does not close files automatically; preserves integrity of system critical files. User files are left as is.
Devices/ Memory Allocated	Deallocated automatically.	Same as exit.	Same as exit.

2.9 Resource Management

Resource management refers to the allocation of peripherals, file space, and memory. With MPX-32, allocation may be static or dynamic, and resources may be shared or unshared.

2.9.1 General

2.9.1.1 Static Allocation

Static allocation refers to those resources which are allocated to a task at activation. Static assignments are specified via ASSIGN and ALLOCATE directives. These directives may be specified when the program is cataloged (see Volume 2, Chapter 2). Alternatively, Job Control or TSM commands (Volume 1, Chapters 5 and 6) may be used to override cataloged assignments or provide additional static assignments. If a statically assigned resource is temporarily unavailable, the requesting task is automatically queued for allocation. Task loading is deferred until all static resources can be allocated.

2.9.1.2 Dynamic Allocation and Deallocation (M.ALOC,M.DALC)

Once task activation is complete, additional resources may be allocated or deallocated via system service calls M.ALOC and M.DALC (see Volume 1, Chapter 7) for files and peripherals and M.GE/M.FE and M.GD/M.FD for memory. This process is called dynamic allocation. A task requesting additional memory is automatically queued until the memory can be allocated. For peripherals and file space, the caller may optionally queue for allocation or try again later. System table space for dynamically allocated files must be specified via the FILES statement provided in the Cataloger.

2.9.1.3 Shared versus Unshared Resources

Shared resources may be allocated concurrently to more than one task. Unshared resources are restricted to one task at a time. Permanent disc files and common memory partitions are always shared resources. Peripherals may be SYSGENed as shared or unshared.

2.9.1.4 Device Allocation (M.PDEV)

Peripherals may be allocated generically by device type, or specifically by device type, channel and subaddress as described in Volume 1, Chapter 7. When allocating generically, unshared peripherals are allocated before shared peripherals. The physical device inquiry system service (M.PDEV) may be used to determine specific attributes of a device.

Certain devices have special allocation considerations:

(a) Discs are always considered shared devices, although temporary file space is unshared. (The generic type code DC may be used to specify any type of disc; DM refers to moving head discs; and DF to fixed head discs.)

(b) Magnetic tape devices may be generically refered to as MT. M9 refers to nine-track magnetic tapes only and M7 refers to seven-track only. Parity, density, etc., are I/O considerations specified as control flags in the FCB. Tapes may have optional volume number specification. For multi-volume reels, both the reel name and volume number are written onto the tape and verified during input operations. For tapes without volume specification (volume=0) the reel ID is an optional parameter which is not written to the tape. (See also Chapter 7.)

(c) The system console has a dual nature. Under the device type CT it is unshared and used as a TSM terminal. However, it is shared with respect to system messages e.g., tape mount requests, real-time aborts, and device status errors. These messages are written via the M.TYPE system service. The ATTENTION button on the front panel functions as the Break key for this device.

(d) TSM terminals must be SYSGENed as unshared devices. Allocation to these terminals is provided by TSM functions.

2.9.1.5 Task-Synchronized Access to Common Resources

MPX-32 provides the structure for tasks to voluntarily synchronize access to a common resource such as a disc file, a sharable device, a common data area, a shared/included procedure area, or any other physical resource.

The capability provided by MPX-32 is a general resourcemark mechanism. Each task using a 'marked' resource must:

use the M.RSML and M.RSMU (Resourcemark Lock/Unlock) services to synchronize access to a resourcemark with other tasks

make the association of a particular resourcemark with an actual resource

What MPX-32 provides is a table of resourcemarks that are currently in use, a mechanism for queuing tasks for each mark, and automatic unlock on a resourcemark when a task terminates (aborts, exits, or is deleted), if the task has not unlocked the resourcemark on its own.

A resourcemark is simply a numeric value from 1 to 64. Values 1-32 are for SYSTEMS' internal use, values 33-64 are available for customer use. The default size of 64 can be increased by using the SYSGEN RMTSIZE directive. However, MPX-32 does not enforce resource access restrictions, i.e., the system does not associate a particular resource with a particular resourcemark. Thus, if several tasks use synchronization service calls to gate access to a resourcemark and another task does not, the 'outside' task will gain the resource just as if no restrictions were active for it.

Tasks synchronizing use of resources are responsible for using resourcemarks that uniquely identify resources across the system. MPX-32 ensures only that a specified mark is within the legal numeric range.

To use resource marking, each cooperating task:

uses M.RSML to lock the resourcemark

performs the access which requires synchronization

uses M.RSMU to unlock the resourcemark and release the highest priority task queued for the resourcemark

The task has several options available if the resourcemark is locked when it issues the M.RSML call. As specified in the call, it can:

obtain an immediate denial return and go on

wait until it can gain ownership of the lock

wait until it can gain ownership or until a specified number of timer units have expired, whichever occurs first

If a single task uses more than one resourcemark, i.e., if it is synchronizing access to more than one resource, the user must exercise care to avoid deadlock situations, e.g., Task A in wait for a lock owned by Task B while Task B is in turn waiting for a lock owned by Task A.

A task using more than one resourcemark can avoid deadlocks by unlocking all locked resourcemarks if it cannot succeed in locking any one of them. The task then waits for the critical unlock to occur before reattempting locks on all the other resourcemarks in the set.

Sample Resourcemark Use by a Task

```
        PROGRAM T4
        M.REQS
        LIST NGLIST
        ENDM
T4      EQU
        M.RSML 33,0          LOCK RSM,INDEF.WAIT,NORM SWAP
        M.WRIT ABC           WRITE TO CRITICAL FILE
        M.RSMU 33            UNLOCK RSM
        M.EXIT
ABC     DATAW C'ABC'         LFC SETUP
        GEN 12/B 80,20/B(SBUF)
        REZ 6W
SBUF    RES 80B
```

2.9.1.6 File Gating

File gating mechanisms implemented in MPX-32 are an extension of the general resourcemarking capability described in the previous section, and are supported by the MPX-32 Executive in cooperation with the File System Executive and the Allocator.

File gating is used to preserve the integrity of critical disc files, and to avoid noncompatible operations on the same file by two or more different tasks. For example, an exclusive lock is used by the operating system to prevent one task from deleting any permanent file while another task is writing to the same file.

MPX-32 supports both synchronized and exclusive file gating. Synchronization gating services enable a task to coordinate concurrent access to a file that it shares with other tasks. Exclusive gating services enable a task to gain sole allocation of a file, as though it were an unshared resource (exclusive lock). File gating services are available to both privileged and unprivileged tasks.

As with resourcemarking, use of file gating services is entirely voluntary, i.e., if a synchronized file gating service is used by a task, a task that is not cooperating in the gating can access a file as if the lock had not been imposed. However, if an exclusive lock has been imposed, no other task can allocate a file until the file is unlocked. Exclusive lock is thus binding on noncooperating as well as cooperating tasks.

Services which support file gating are:

M.FXLS	Set Exclusive File Lock
M.FXLR	Release Exclusive File Lock
M.FSLS	Set Synchronization File Lock
M.FSLR	Release Synchronization File Lock

2.9.1.6.1 Gating Mechanism and Support Structures

MPX-32 maintains a memory resident File Lock Table (FLT), which contains an entry showing the UDT index and starting sector for each permanent disc file that is currently allocated, whether it is locked or not. The FLT entry also includes a count of tasks that are currently queued for the lock and the task number of the task which currently owns the lock.

When a task requests a lock and the file is not yet locked, the task's number is entered in the FLT and execution continues. If a task requests a lock and the file is already locked, the task can obtain an immediate denial and continue, or go into a general wait state (SWGQ) until the file is unlocked. As in resourcemarking, a requesting task has the additional option of setting a watchdog timer for the maximum time it will wait to obtain a lock.

Figure 2-2 illustrates task interface with the locking mechanisms used by MPX-32 for synchronized locks. The same basic mechanism is used for exclusive locks, except that on an exclusive lock, the requesting task must be the only task that has allocated the file.

If a task locks a file for either synchronized or exclusive gating and does not unlock it, the lock is automatically removed when the task deallocates the file or exits. If a task is aborted, the lock is released.

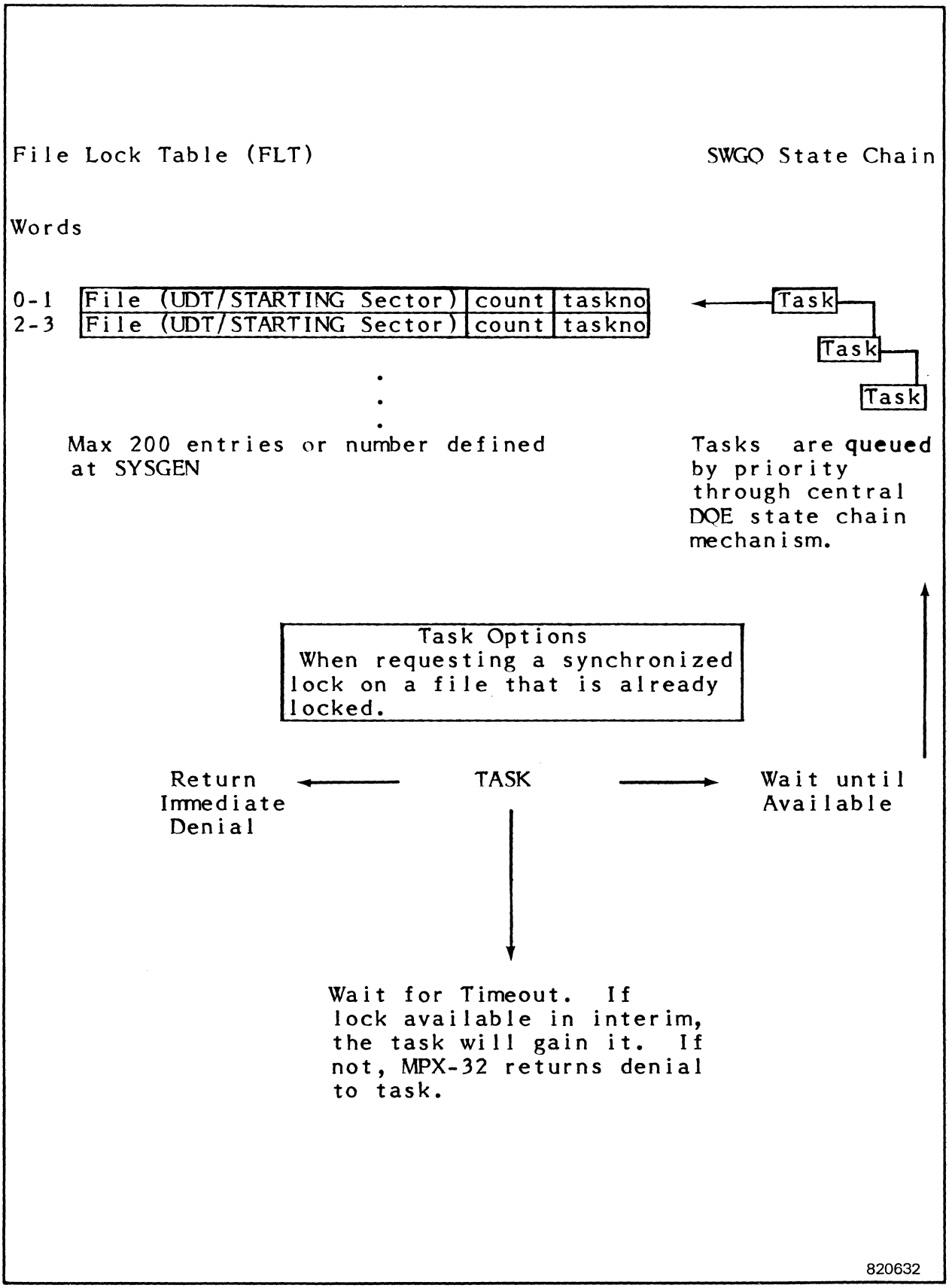


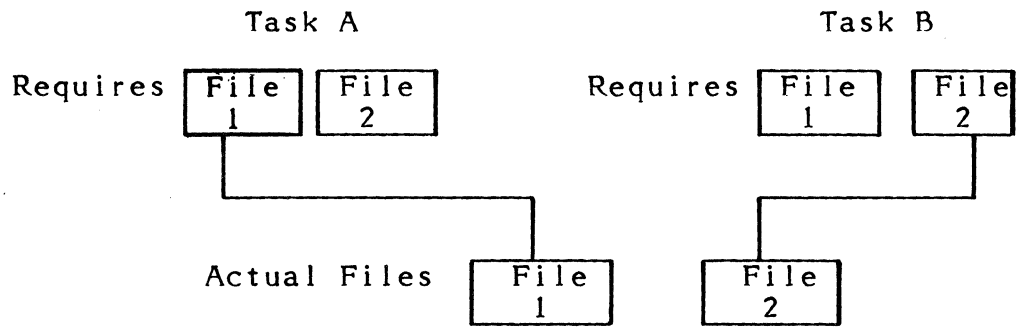
Figure 2-2 File Lock Overview

2.9.1.6.2 Task Implementation

Locks are performed only on files which are already allocated to a task. The task uses a logical file code (rather than a file name) to identify the file lock. This enables the task to lock by critical operation rather than specific name.

2.9.1.6.3 Avoiding Deadlocks

If a task must lock more than one file at a time, there is potential deadlock with other tasks as illustrated below.



Task A needs both File 1 and 2 simultaneously, as does Task B. Task A gets File 1, but before it can get File 2, Task B gets File 2. Task A goes into a wait for File 2. Task B is in wait for File 1. Since neither task can get both the files it requires simultaneously, processing does not continue. To further complicate the deadlock, any other task that queues for either file will wait indefinitely.

The way to ensure against deadlocks is for a task that needs to lock more than one file simultaneously to perform the locks only when all such files are available. This can be accomplished via the denial or timed denial return options available with MPX-32 file lock services.

2.9.2 Operating System Memory Allocation

MPX-32 occupies the lower portions of each task's address space, thereby allowing MPX-32 to run "mapped" with each task.

MPX-32 maintains lists of available memory for allocation to tasks and for I/O that requires intermediate buffering.

2.9.2.1 I/O Buffer and I/O Queues

The system buffer pool is an area of 'E' class memory which is contiguous to the resident system and has a size specified at SYSGEN by the POOL directive. The entire buffer pool is write-protected from the unprivileged task and is intended for use exclusively by system services. The buffer pool is mapped into the address space of each task. System buffer space is allocated in 2W increments. The maximum size of any entry is 192 words. The entries fall into one of three categories:

(1) I/O Queues - approximately 26 words

These are allocated when IOCS queues a request and deallocated when post I/O processing is complete.

(2) Message or Run Request Buffers - up to 192 words

These are allocated by the M.SMSGR or M.SRUNR services and deallocated when receiver processing is complete.

(3) 'E' Class I/O Buffers - up to 192 words

These are allocated and deallocated when I/O queues for I/O to E class devices, but only when actually necessary.

None of these buffers are allocated permanently.

For tasks which require I/O buffers for 'E' class devices greater than 192 words, IOCS automatically allocates one 'E' class memory map block. This buffer is deallocated with the I/O queue when I/O is complete.

2.9.2.2 Blocking Buffers for Blocked I/O

File assignments for permanent files (ASSIGN1's), and devices (ASSIGN3's), optionally specify that a file is blocked or unblocked. (The default is blocked.) If blocked, blocking buffers for the files are allocated at load time immediately above the TSA. The Catalog BUFFERS directive may be used to provide additional blocking buffer space for dynamically allocated, blocked files. (See Volume 2, Chapter 2.)

2.9.2.3 Task Service Area (TSA)

This area comprises the first part of a task's address space located immediately above the operating system. See Figure 2-1 in Section 2.1.3 and Figure 2-3 in Section 2.8.4.

2.9.3 Memory Class Requirements

When a task is cataloged, the user specifies the class of memory required at run time. User tasks are divided into two types based upon their memory class requirements. Tasks that require direct I/O to Class 'E' devices and must run at the fastest memory speed available are type E and are allocated space within Class 'E' memory. These user tasks are required to use the Cataloger ENVIRONMENT directive with a parameter 'E' to indicate the necessity of Class 'E' memory.

Tasks which are not "memory speed critical" can request various types of memory allocation; these requirements are also specified for MPX-32 via a Cataloger ENVIRONMENT directive parameter:

Parameter	Result
S	Execution delayed until Class 'S', 'H', or 'E' available. (Default)
H	Execution delayed until Class 'H' or 'E' available.
E	Execution delayed until Class 'E' available.

In situations where a 32 Series system has no memory installed of the class the user task requests, the first available lower class is allocated to that task.

The absence of an ENVIRONMENT parameter for memory class is dealt with by assuming that tasks are to be loaded into any memory class available.

2.9.4 Memory Allocation for Tasks

The unit of memory allocation is called a map block, and is 8KW on the 32/7x computer and 2KW on the CONCEPT/32. All user tasks are discontiguously loaded into a whole number of physical map blocks, utilizing the SEL map itself to create their contiguous logical address space. No partial map blocks are allocated.

This scheme allows user tasks to dynamically expand and contract their address space by using the M.GE/M.FE and M.GD/M.FD service calls described in Section 8. Figure 2-3 illustrates the logical address space of a task.

Memory Protection

The unit of memory protection is called a protection granule, and is 512W. Thus it is possible to protect a task's TSA even though it is in the same map block as the data section (DSECT).

2.9.4.1 Static Memory Allocation

The Cataloger determines the size, in protection granules, of a cataloged load module. The cataloger directive, ALLOCATE, may be used to specify additional bytes of memory. At activation time, the size of the TSA is determined and rounded up to a number of protection granules. This value is added to the cataloged requirement to determine task size. Additionally Job Control or TSM ALLOCATE directives may be used to specify the total task size of the DSECT. The final sum is rounded up to a map block increment.

2.9.4.1.1 Static vs Dynamic Memory Partitions

Characteristics	Static	Dynamic
Logical addresses	Fixed at SYSGEN	Fixed by File Manager CREATEM
Physical addresses	Fixed at SYSGEN	Variable
Allocation unit	512 words	8K words (32/7x) 2K words (CONCEPT/32)
Time of allocation	SYSGEN	Run time via M.SHARE
Time of deallocation	Never	When allocation count=0
Inclusion	Automatic via Activation	Run time via M.INCL
Exclusion	Automatic via Exit or M.EXCL	Automatic via Exit or M.EXCL
Owner names or task numbers	None	Established by M.SHARE caller
Swapping	Never swapped	Swappable when user count=0

2.9.4.1.2 Memory Partition Applications

Characteristics	Global	DATAPOOL	Extended Common	CSECTs
Cataloger resolves references	Yes	Yes	No	Yes
Compiler resolves references thru extended bases	No	No	Yes	N/A
Must be logically below 128KW	Yes	Yes	No	Yes
Variables are order dependent	Yes	No	Yes	N/A
Static	Yes	Yes	Yes	No
Dynamic	Yes	Yes	Yes	Yes

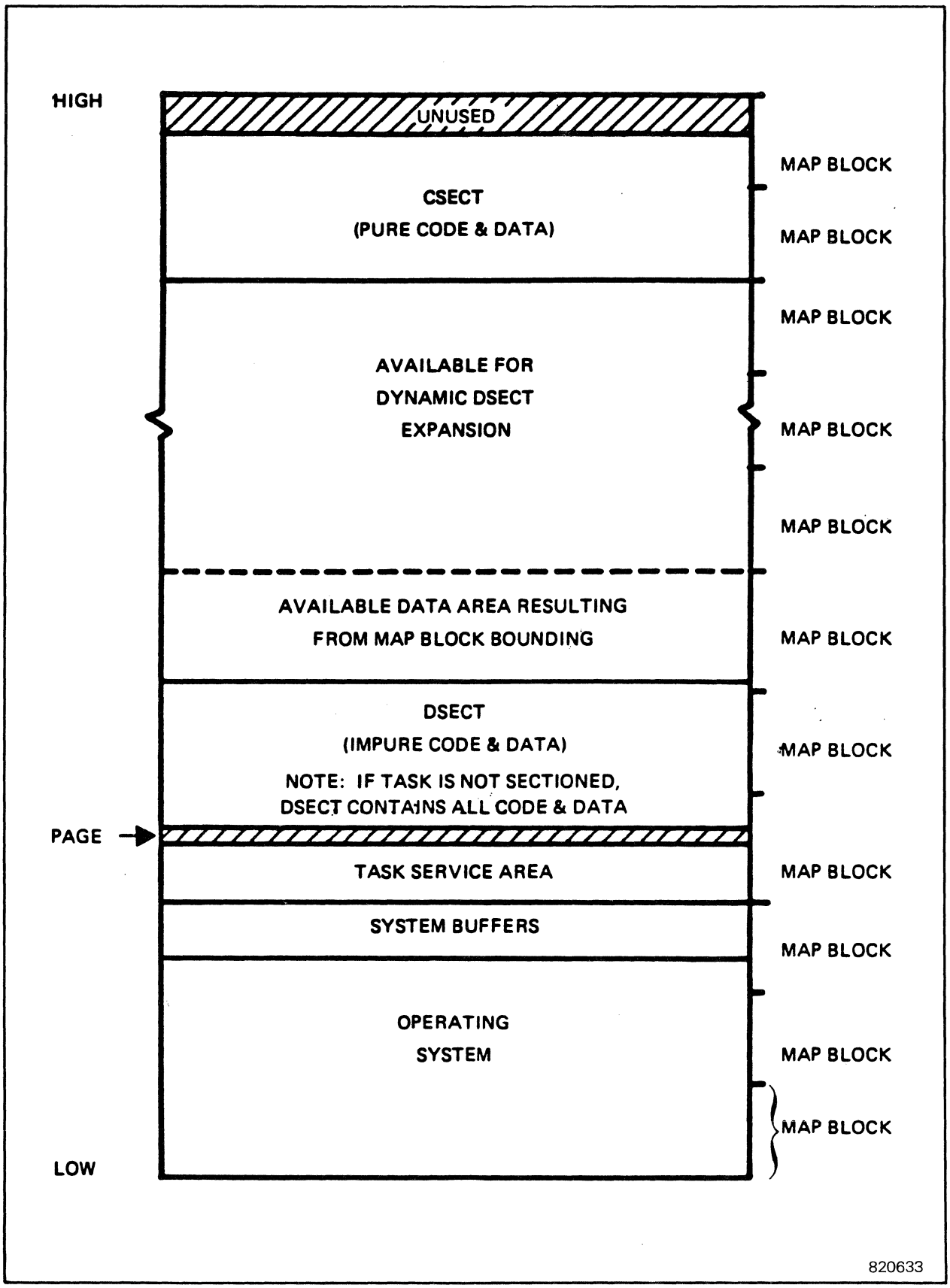


Figure 2-3. Sample Task Address Space

2.9.4.2 Dynamic Address Space Expansion/Contraction (M.GE,M.FE,M.GD,M.FD)

A task can expand/contract both its execution and extended data space via system services. The M.GE service appends a map block to the user's execution space starting at the top of his DSECT. M.GE can be used more than once to obtain additional map blocks, as long as they are available in the task's logical address space. The M.FE service frees the most recently obtained map block, i.e., it works in the opposite order of M.GE. The M.GD service appends a map block to the user's extended indexed data space, starting from 128KW. (Like M.GE, it can be used more than once.) The M.FD service frees the most recently obtained map block from the extended indexed data space, i.e., it works in the opposite order of M.GD.

2.9.4.3 Extended Indexed Data Space

MPX-32 provides limited support for logical addresses above 128KW. The following restrictions apply to the use of this address space:

- Instructions cannot be executed in this logical space.
- The user must reference this space via index registers. Negative offsets are invalid in the word address field of any instruction as long as the indexed addressing mode is active.
- Some system services cannot access data in this logical space. Full I/O support is provided via the expanded FCB specification.
- No memory protection is provided for this logical space on the 32/7x.
- No data initialization facilities are provided for this logical space.
- The user must dynamically request this logical space to be mapped via the M.GD, M.SHARE or M.INCL system services.
- GLOBAL and DATAPOOL are not supported in extended data space.

2.9.4.4 Intertask Shared GLOBAL and DATAPOOL Memory (M.SHARE,M.INCL,M.EXCL)

Intertask shared memory is provided under MPX-32 through GLOBAL and DATAPOOL memory partitions. As in RTM, there are up to one hundred Global regions (GLOBAL00 - GLOBAL99) plus a DATAPOOL partition available to the user.

Global and DATAPOOL partitions can be defined either via SYSGEN or through the CREATEM command in the File Manager.

Partitions created at SYSGEN are considered permanently allocated; they are assigned both physical and logical memory attributes which apply to any task that references the partitions. This type of allocation is called static allocation. The static Global and Datapool partitions are defined in integral numbers of protection granules.

Both GLOBAL and DATAPOOL partitions are located in an integral number of physical map blocks starting on a map block boundary and ending logically at the top of the user's execution space (128KW). Areas are mapped into user space when required. See Sections 2.9.4.1.1 and 2.9.4.1.2.

Write protection is available to prevent the user from storing into a common area to which he does not have write access.

Alternatively, the user can define dynamic shared memory partitions via the File Manager. There are several key distinctions between statically and dynamically allocated common:

- Statically allocated common is fixed in physical memory even when no task is sharing it through its map. Dynamically allocated common is deallocated when its allocation count equals zero.

- Statically allocated common is allocated in increments of 512 words, while dynamically allocated common is allocated in map block increments (8KW on a 32/7x computer and 2KW on a CONCEPT/32).

- Statically allocated common is invoked on a system-wide basis. Dynamically allocated common is based on a subsystem concept, where a single task issues an M.SHARE request and other tasks request that task's common to be included via the M.INCL system service. A particular common partition, e.g., DATAPOOL, can be defined concurrently in several such subsystems. However, each subsystem has a physically unique partition.

- Dynamically allocated common can be excluded from a task via the M.EXCL system service. The user can elect to subsequently include another dynamically allocated common area via M.INCL. Statically allocated partitions are not supported.

- All logical references to common, whether statically or dynamically allocated, are resolved by the Cataloger. This is possible because the logical address of a system common partition is fixed when the partition is defined.

- Unlike RTM, load modules from one MPX-32 configuration are compatible with another configuration, even if GLOBAL or DATAPOOL are allocated different physical addresses. The only compatability requirement is that both systems employ the same logical conventions.

Figure 2-4 illustrates a relatively complex view of the relationship between logical address spaces and statically allocated and dynamically allocated common partitions. The figure also introduces the allocation considerations for shared procedures, which are described in the section which follows.

In brief,

- Tasks A, B, and C all reference a static DATAPOOL partition.
- Task A has used an M.SHARE service for dynamic GLOBAL10 and Task B has used M.INCL for GLOBAL10. Thus GL10 is mapped at the same location in each logical address space.
- Tasks A and B use M.INCL for GL02, and Task C has used M.SHARE for GL02, to use for intertask communication. Thus GL02 is mapped into the same location in each logical address space.
- Task D shares no memory or code with other tasks. Map blocks 7 and up are available to the task.
- Tasks A and B are shared. They have CSECT mapped at the same location in each logical address space.

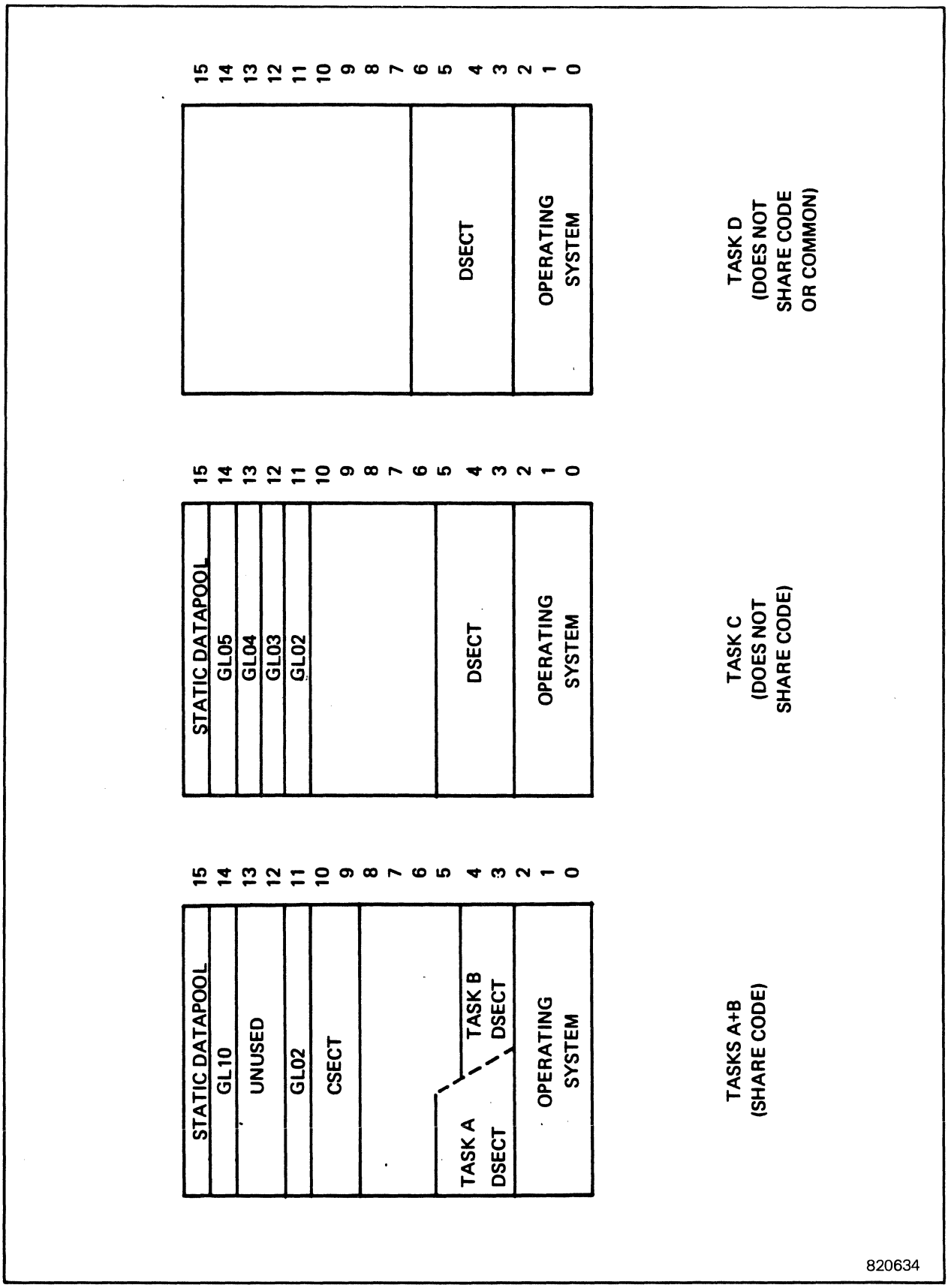


Figure 2-4 Sample Allocation of Common Memory Partitions and Common Code

2.9.4.5 Shared Procedures

MPX-32 supports shared procedures. A Catalog parameter, SHARED, is used to specify a shared procedure, which must consist of a CSECT (pure code and data) and a DSECT (any impure data). When a shared procedure is activated for the first time, the system loader reads both the shared procedure section and the impure data section into memory. The shared procedure section is mapped as if the task had issued an M.INCL service request. Every subsequent activation of the shared procedure causes only the impure data section of the shared procedure to be loaded since the procedure section is already in memory, has not been altered, and can be mapped as in an M.INCL service request.

Shared procedures, like shared memory areas, have an allocation count associated with them to prevent premature deallocation of the memory they occupy and a user count to control the swapping of these partitions.

Shared procedure space is allocated in the highest available logical address (below common partitions, if any). (See Figure 2-4.)

2.10 MPX-32 Faults/Traps and Miscellaneous Interrupts

MPX-32 provides interrupt and trap processors for all standard interrupts and traps. A list of these interrupts with associated information is shown in Table 2-4.

Processing for trap levels 03, 04, 05, 09 is dependent upon the location of the instruction causing the trap. A system crash (M.KILL; not OPCOM KILL) results if the offending instruction is issued from a location within the MPX-32 system area. If the instruction is issued from a location within a task area, the task is aborted.

When a system crash occurs as a result of a trap handler entry, the CPU halts with the registers containing the following information:

R0=PSD Word 0 (when trap generated)
R1=PSD Word 1 (when trap generated)
R2=Real address of instruction causing trap
R3=Instruction causing trap
R4=CPU status word (from trap handler)
R5=Crash code:
MP01=X'4D503031' (See H.IP02 Codes)
NM01=X'4E4D3031' (Non-Present Memory - H.IP03)
UI01=X'55493031' (Undefined Instruction - H.IP04)
PV01=X'50563031' (Privilege Violation - H.IP05)
MC01=X'4D433031' (Machine Check - H.IP07)
SC01=X'53433031' (System Check - H.IP08)
MF01=X'4D463031' (Map Fault - H.IP09)
CP01=X'42543031' (Cache Parity - H.IP10) 32/87 only
BT01=X'42543031' (Block Mode Timeout - H.IP0E)
HT01=X'48543031' (Privileged Halt Trap - H.IPHT) CONCEPT/32 only
R6=Real address of register save block
R7=C'TRAP'=X'54524150'

Relative Priority	Logical Priority	Dedicated IVL Loc.	Description	System Action: PSD in OS Area	SYSTEM Action: PSD In Task Area	Crash/Abort Code
00	00	0F4	Power Fail Safe Trap	Halt	Halt	N/A
01	01	0FC	System Override Trap	Halt	Halt	N/A
02/12	12	0E8	Memory Parity Trap	M.KILL	M.KILL	MP01
03/24	24	190	Non-Present Memory Trap	M.KILL	Abort Task	NM01
04/25	25	194	Undefined Instruction Trap	M.KILL	Abort Task	UI01
05/26	26	198	Privilege Violation Trap	M.KILL	Abort Task	PV01
06		180	Supervisor Call Trap	Process SVC	Process SVC	See Note 1
07		184	Machine Check Trap	M.KILL	M.KILL	MC01
08		188	System Check Trap	M.KILL	M.KILL	SC01
09		18C	Map Fault Trap	M.KILL	Abort Task	MF01
0E		0E4	Block Mode Timeout Trap	M.KILL	M.KILL	BT01
0F/29	29	1A4	Arithmetic Exception Trap	N/A (Not Enabled)	Record Exception in TSA	N/A
13	13	0EC	Attention Interrupt	Process Int.	Process Int.	N/A
27	27	19C	Call Monitor (CALM) Interrupt	Process CALM	Process CALM	See Note 2
28	28	1A0	Real-Time Clock Interrupt	Process Int.	Process Int.	N/A

Note 1: Illegal SVC Task Abort Codes

- SV01 Abort of unprivileged task using M.CALL
- SV02 Invalid SVC # abort
- SV03 Abort of unprivileged task attempting use of a "privileged-only" service
- SV04 Invalid SVC type abort
- SV05 Abort of unprivileged task attempting M.RTRN

Note 2: Illegal CALM Task Abort Codes

- CM01 CALM instruction not located
- CM02 Expected CALM instruction does not have CALM (X'30') opcode
- CM03 Invalid CALM number

3. TASK STRUCTURE AND OPERATION

3.1 Task Identification

Under MPX-32, the user can communicate with and control tasks either by task name or task number (unless a task is multicopied, in which case, the task number is required). The task name is the cataloged name of the load module file containing the task. The task number is assigned when the task is activated and is a sequential 24-bit number concatenated with an 8-bit DQE index. Task numbers are unique for each task in the system.

Each task is also associated with an owner. Valid owner names are specified in the M.KEY file, if it exists; otherwise, all owner names are valid. An owner can have any number of tasks with the same or different task names active on the system at any point in time.

In addition to the task numbers, each batch job is assigned a unique sequence number when the job is entered in the batchstream.

Active tasks can be listed by:

Task number

Owner name

Task Load Module Name

Batch sequence number (if Batch)

Execution Class (Real-time, Online, Batch)

Pseudonym used by MPX-32 to further identify the task, e.g., by the terminal it is running on

or a combination of the above.

The system provides the OPCOM LIST commands and the system service M.ID for listing any active task by specifying a unique combination of these attributes.

3.2 Task Structures

A task is structured to meet a user's particular requirements by defining the contents of a unique address space. A unique address space is a mapped logical address space which contains up to 128KW of code and data and may also contain up to 128KW of extended indexing data on a 32/7x computer. On a CONCEPT/32, a unique address space is mapped logical address space which contains up to 128KW of code and data and may also

contain up to 384KW of extended indexing data. This unique address space is established when the task is activated and may contain:

(a) A non-shared task

(b) A task which shares procedure (code and pure data) with another task

(c) A task which shares memory (common storage or user defined use) with another task

Shared memory considerations are described in Chapter 2.

3.2.1 Non-Shared Tasks

This type of address space contains a single task including its Task Service Area (TSA), its code section (CSECT - write protected memory containing code and pure data), and its data section (DSECT - read/write memory containing impure data). (Note: Tasks which are not sectioned have only a DSECT, which contains the code and all data.)

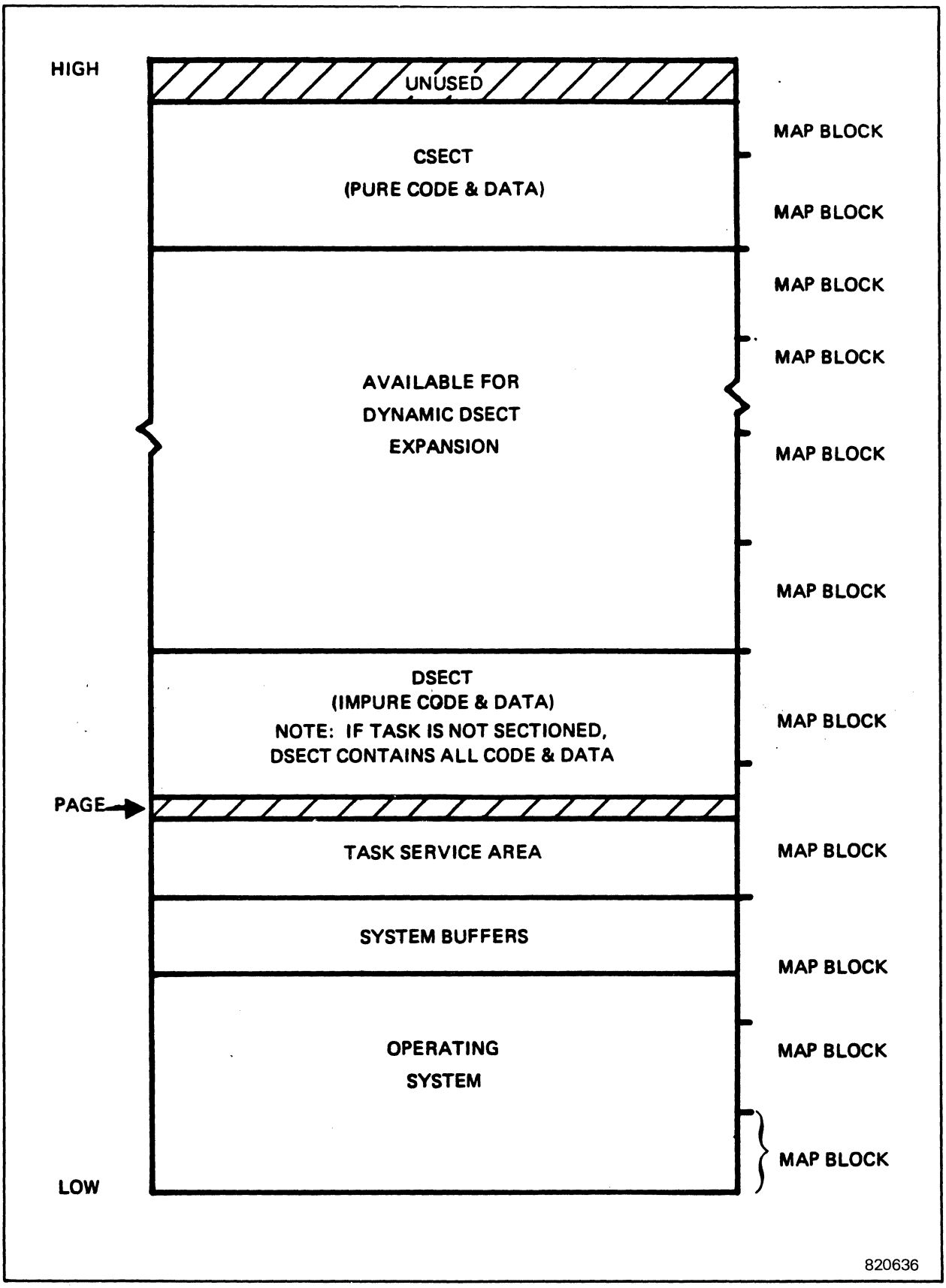


Figure 3-1. Non-Shared Task Environment Address Space

3.2.2 Shared Tasks

When a task is cataloged, the user can specify that a program section is to be shared. A program section (CSECT) consists of code and pure data. This section is write protected and mapped into each requesting user's logical address space. A separate data section (DSECT) is mapped into each logical address space, as illustrated in Figure 3-2.

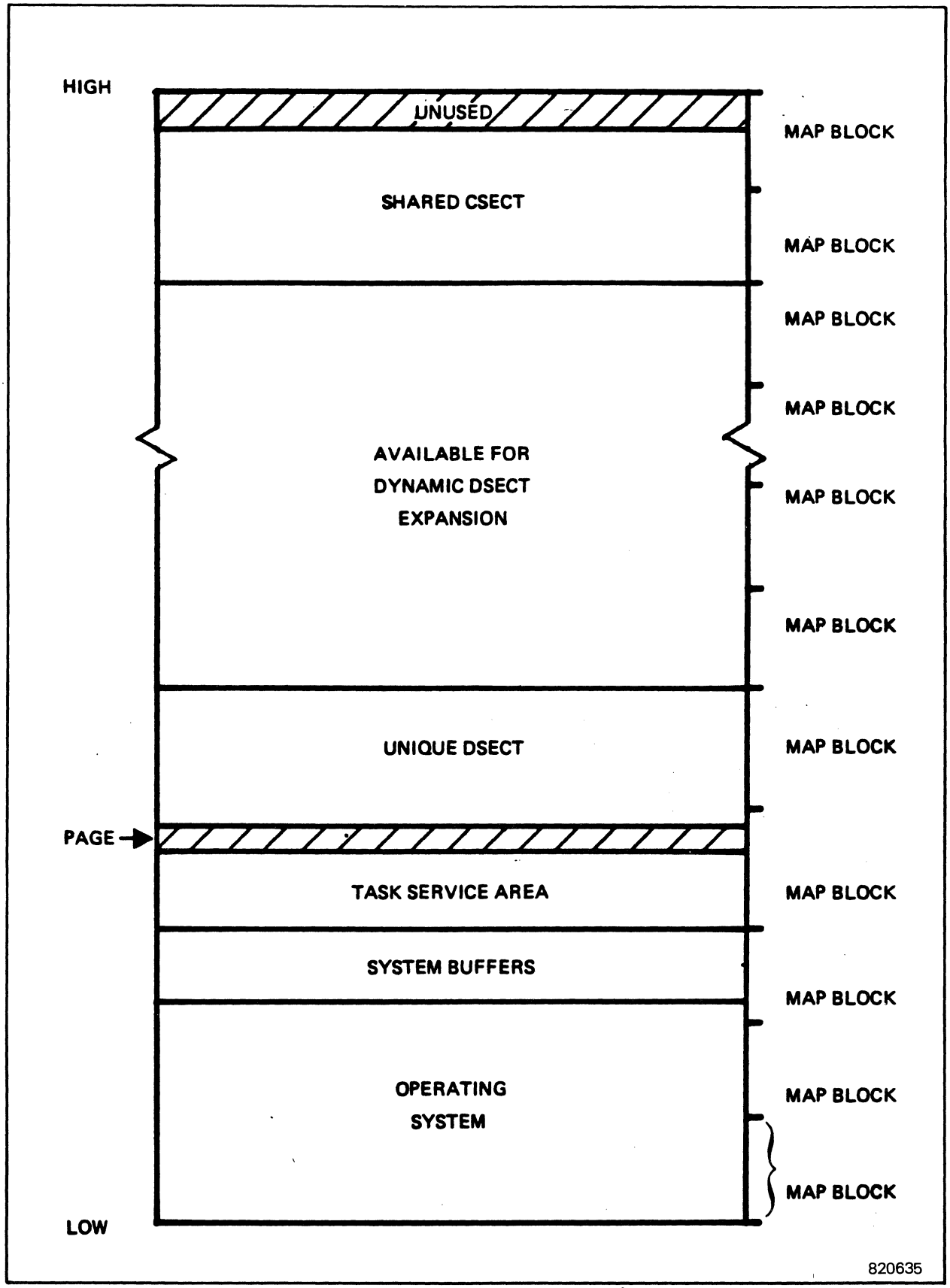


Figure 3-2 Shared Task Address Space

3.2.3 Multicopied Tasks

An owner or several owners can have tasks with the same name (and the same load module) active concurrently. This is accomplished by cataloging the task as multicopy. The task name is not sufficient to communicate with multicopy tasks; the task number must be used.

3.2.4 Unique Tasks

Although only one copy of a task that is unique can be active on the system at a given time, the MPX-32 run request mechanism can be used to queue run requests to the task, so that as soon as one user stops executing, another can begin. (See Intertask Communication and Activation, Section 3.4.)

3.3 Task Execution

Tasks are introduced to the system by a request to activate the cataloged load module by name. Activation can be requested in several different ways.

Batch tasks are activated via the $EXECUTE Job Control statement.

Online tasks are activated via the RUN command under TSM.

Real time tasks can be activated by the M.ACTV or M.PTSK system services. The requestor uses the M.PTSK service to rename the task or to specify additional or alternate resource requirements for the task. Real time tasks can also be activated by the job control statement $ACTIVATE or the OPCOM command, ACTIVATE.

Real time tasks can also be activated via timers or interrupts.

3.4 Intertask Communication

MPX-32 provides both message and run request send/receive processing. Run request services allow a task to queue an execution request (with optional parameter pass) for another task. Message services allow a task to send a message to another active task. The services provided for use by the destination tasks are called "receiving task services". Those provided for tasks which issue the requests are called "sending task services". Message and run request services use the software interrupt scheduling structure described in Section 2.4.

3.4.1 Receiving Task Services

3.4.1.1 Establishing Message Receivers (M.RCVR)

In order to receive messages sent from other tasks, a task must be active and have a message receiver established. A message receiver is established by calling the system service M.RCVR, and providing the receiver routine address as an argument with the call.

3.4.1.2 Establishing Run Receivers

Any valid task may be a run receiver. Although a set of special run receiver services are provided, in the most simple case, they need not be used. The run receiver mechanism is provided by the system to allow queued requests for task execution, with optional parameter passing. The cataloged transfer address is used as the run receiver execution address. The task load module name is used to identify the task to be executed unless the run request is being sent to a multicopy task in the waiting for run request state, in which case the task number should be used to identify the particular copy of that task. If a load module name is specified and a task of that load module name is currently active, but is not a single-copied task, the load module is activated (multi-copied) to process this request. When a single-copied task exits, any queued run requests are executed. If a run request is issued for a task that is not currently active, the task is activated automatically.

3.4.1.3 Execution of Message Receiver Programs

When a task is active and has a message receiver established, it can receive messages sent from other tasks. A message sent to this task causes a software (task) interrupt entry to the established message receiver.

3.4.1.4 Execution of Run Receiver Programs

When a valid task is executed as a result of a run request sent by another task, it is entered at its cataloged transfer address. A run receiver executes at the normal task execution (base) level.

3.4.1.5 Obtaining Message Parameters (M.GMSGP)

When the message receiver is entered, R1 contains the address of the message queue entry in memory pool. The task may optionally retrieve the message directly from memory pool, or the task may call a receiver service (M.GMSGP) to store the message into the designated receiver buffer. If the M.GMSGP service is utilized, the task must present the address of a five-word Parameter Receive Block (PRB) as an argument with the call.

3.4.1.6 Obtaining the Run Request Parameters (M.GRUNP)

When the run receiver is entered, R1 contains the address of the run request queue entry in memory pool. The task may optionally retrieve the run request parameters directly from memory pool, or the task may call a receiver service (M.GRUNP) to store the run request parameters into the designated receiver buffer. If the M.GRUNP service is utilized, the task must present the address of a three-word Parameter Receive Block (PRB) as an argument with the call.

3.4.1.7 Exiting the Message Receiver (M.XMSGR)

When processing of the message is complete, the message interrupt level must be exited by calling the M.XMSGR service. When M.XMSGR is called, the address of a two-word Receiver Exit Block (RXB) must be provided. The RXB contains the address of the return parameter buffer, and the number of bytes (if any) to be returned to the sending task. The RXB will also contain a return status byte to be stored in the Parameter Send Block (PSB) of the sending task. After message exit processing is complete, the message receiver queue for this task is examined for any additional messages to process. If none exists, a return to the base level interrupted context is performed.

3.4.1.8 Exiting the Run Receiver Task (M.EXIT or M.XRUNR)

When run request processing is complete, the task may use either the standard exit call (M.EXIT), or the special run receiver exit service (M.XRUNR).

If the standard exit service (M.EXIT) is used to exit the run receiver task, no user status or parameters are returned. Only completion status is posted (in the scheduler status word) of the Parameter Send Block (PSB) in the sending task. After completion processing for the run request is accomplished, the run receiver queue for this task is examined, and any queued run request causes the task to be re-executed. If the run receiver queue for this task is empty, a standard exit is performed.

If the special exit (M.XRUNR) is used to exit the run receiver task, the address of a two-word Receiver Exit Block (RXB) must be provided as an argument with the call. The RXB contains the address of the return parameter buffer, and the number of bytes (if any) to be returned to the sending task. The RXB also contains a return status byte to be stored in the Parameter Send Block (PSB) of the sending task. After completion processing for the run request is accomplished, the exit control options in the RXB are examined. If the "wait" exit option is used, the run receiver queue for this task is examined for any additional run requests to be processed. If none exist, the task is put into a wait-state, waiting for the receipt of new run requests. Execution of the task does not resume until such a request is received. If the "terminate" exit option is used, any queued run requests are processed. If the run receiver is empty, however, a standard exit is performed.

3.4.1.9 Waiting for the Next Request (M.SUSP, M.ANYW or M.EAWAIT)

In addition to the wait options described under the heading "Exiting the Run Receiver Task", a task can use M.SUSP, M.ANYW, or M.EAWAIT services. When operating at the base execution level, a task that has established a message receiver can use the M.SUSP service call to enter a wait-state until the next message is received.

A task may also make use of the special M.ANYW service from the base software level. The M.ANYW service is similar to M.SUSP. The difference is that whereas the M.SUSP wait-state is ended only upon receipt of a message interrupt, timer expiration, or resume, the M.ANYW wait-state is ended upon receipt of any message, end action, or break software interrupt.

M.EAWAIT is similar to M.ANYW except that if no requests are outstanding, an immediate return is made to the caller.

3.4.2 Sending Task Services

3.4.2.1 Message Send Service (M.SMSGR)

A task may send a message to another active task, providing the destination task has established a message receiver. The sending task must identify the destination task by task number. When the send message service (M.SMSGR) is called, the address of a Parameter Send Block (PSB) must be provided as an argument. The PSB specifies the message to be sent, whether or not any parameters are to be returned, and the address of a user end action routine. User status can be returned by the destination task. The operating system also returns completion status in the PSB. No-wait and no-call-back control options are also provided.

3.4.2.2 Send Run Request Service (M.SRUNR)

A task may send a run request to any active or inactive task, identifying the task by load module name. When the run request service (M.SRUNR) is called, the address of a Parameter Send Block (PSB) must be provided as an argument. The PSB format allows for the specification of the run request parameters to be sent, any parameters to be returned, scheduler and user status, as well as the address of a user end action routine. No-wait and no-call-back control options are also provided.

3.4.2.3 Waiting for Message Completion

A message may be sent in either the wait or no-wait mode. If the wait mode is used, execution of the sending task is deferred until processing of the message by the destination task is complete. If the no-wait mode is used, execution of the sending task continues as soon as the request has been queued. The operation in progress bit in the scheduler status field of the PSB may be examined to determine completion. A sending task may issue a series of no-wait mode messages followed by a call to the M.ANYW or M.EAWAIT system wait service. This allows a task to wait for the completion of any no-wait messages previously sent. The completion of such a message will cause resumption at the point after the M.ANYW or M.EAWAIT call.

3.4.2.4 Waiting for Run Request Completion

Waiting for a run request completion follows the same form and has the same options as waiting for message completion.

3.4.2.5 Message End Action Processing (M.XMEA)

User specified end action routines associated with no-wait message send requests are entered at the end action software interrupt level when the requested message processing is complete. Status and return parameters will have been posted as appropriate. When end action processing is complete, the M.XMEA service must be called to exit the end action software interrupt level.

3.4.2.6 Run Request End Action Processing (M.XREA)

Run request end action processing follows the same form and has the same options as message end action processing. The only difference is that the M.XREA service is used instead of M.XMEA.

3.4.3 Parameter Blocks

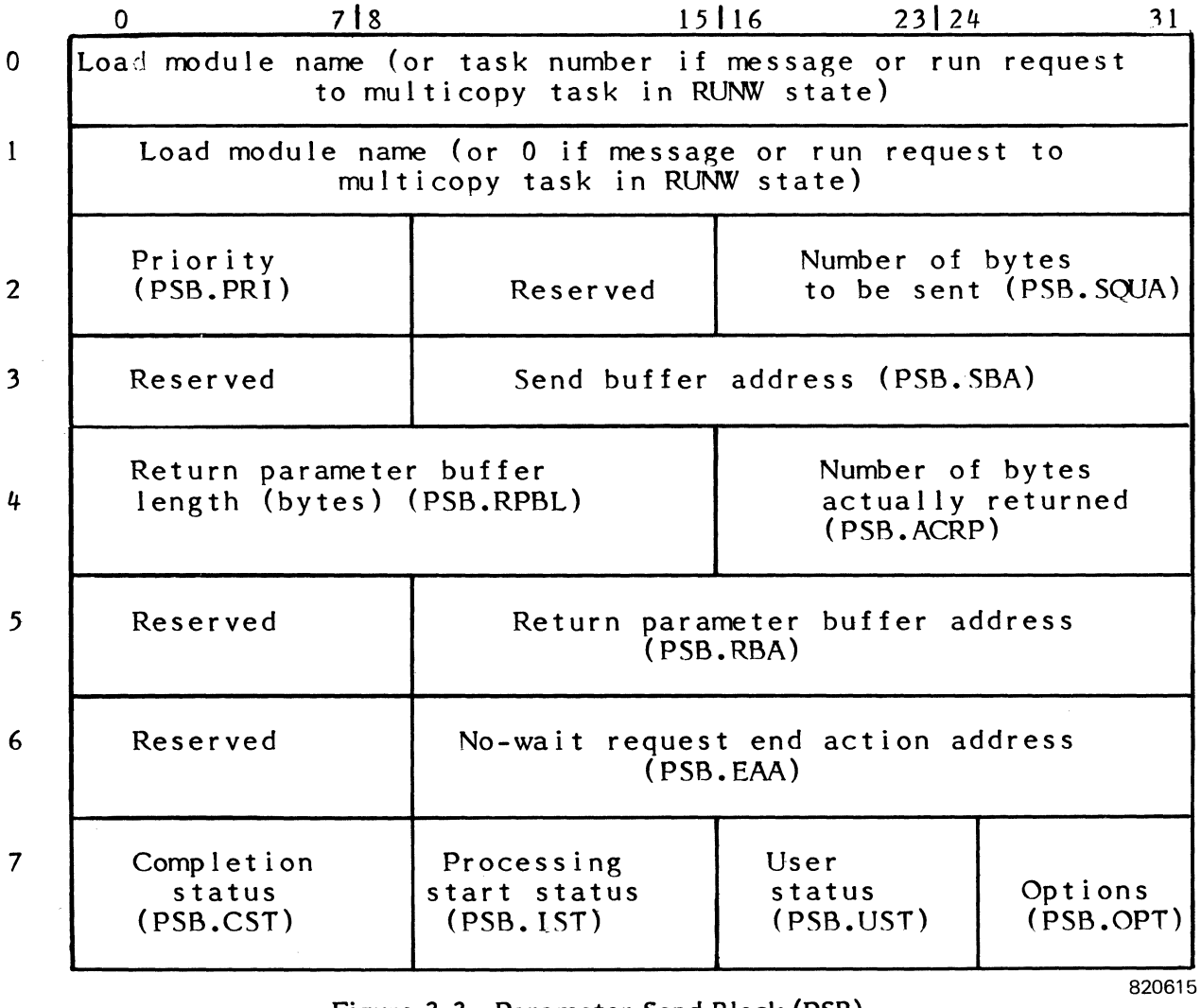
Parameters for run requests and messages are passed via parameter blocks established within the user task. The parameter blocks are described in this section.

3.4.3.1 Parameter Send Block (PSB)

The Parameter Send Block (PSB) is used to describe a send request issued from one task to another. The same PSB format is used for both message and run requests. The address of the PSB (doubleword bounded) must be presented as an argument when either the M.SMSGR or M.SRUNR services are invoked.

Please note that a task number, not a load module name, must be used if sending a message request or if sending a run request to a multicopied task which is waiting for a run request.

WORD



820615

Figure 3-3. Parameter Send Block (PSB)

WORD 0

Bits 0-31 Load Module Name - Characters 1-4 of the name of the load module to receive the run request or task number of the multicopied task to receive run request, or of the task to receive the message.

WORD 1

Bits 0-31 Load Module Name - Characters 5-8 of the name of the load module to receive the run request, or Unused if message send.
Note: Must be 0 if sending a run request to a multicopy task in RUNW state.

WORD 2

Bits 0-7 Priority - This field contains the priority of the send request (1-64). If the value of this field is zero, the priority used defaults to the execution priority of the sending task. This field is not examined if the sending task is privileged.

Bits 8-15 Reserved.

Bits 16-31 Number of Bytes to be Sent - This field specifies the number of bytes to be passed (0-768) with the message or run request.

WORD 3

Bits 0-7 Reserved.

Bits 8-31 Send Buffer Address - This field contains the word address of the buffer containing the parameters to be sent.

WORD 4

Bits 0-15 Return Parameter Buffer Length - Contains the maximum number of bytes (0-768) that may be accepted as returned parameters.

Bits 16-31 Number of Bytes Actually Returned - This field is set by the send message or run request service upon completion of the request.

WORD 5

Bits 0-7 Reserved.

Bits 8-31 Return Parameter Buffer Address - Contains the word address of the buffer into which any returned parameters will be stored.

WORD 6

Bits 0-7 Reserved.

Bits 8-31 No-Wait Request End Action Address - Contains the address of a user routine to be executed at a interrupt level upon completion of the request.

WORD 7

Bits 0-7 Completion Status - This bit encoded field contains completion status information posted by the operating system as follows:

Bit	Meaning When Set
0	Operation in progress (busy).
1	Destination task was aborted before completion of processing for this request.
2	Destination task was deleted before completion of processing for this request.
3	Return parameters truncated (attempted return exceeds return parameter buffer length).
4	Send parameters truncated (attempted send exceeds destination task receiver buffer length).
5	User end action routine not executed because of task abort outstanding for this task (may be examined in abort receiver to determine incomplete operation).
6-7	Reserved.

Bits 8-15 Processing Start (Initial) Status - This value encoded field contains initial status information posted by the operating system as follows:

Code	Definition
0	Normal initial status.
1	Message request task number invalid.
2	Run request load module name not found in System Master Directory (SMD).
3	File associated with run request load module name is password protected.
4	File associated with run request load module name does not have a valid load module format.
5	Dispatch Queue Entry (DQE) space is unavailable for activation of the load module specified by a run request.

6	An I/O error was encountered while reading the SMD to obtain the file definition of the load module specified in a run request.
7	An I/O error was encountered while reading the file containing the load module specified in a run request.
8-9	Reserved.
10	Invalid priority specification. Note: An unprivileged task may not specify a priority which is higher than its own execution priority.
11	Invalid send buffer address or send quantity exceeds 768 bytes.
12	Invalid return buffer address.
13	Invalid no-wait mode end action routine address.
14	Memory pool unavailable.
15	Excessive no-wait requests. Limit is 5 for unprivileged tasks and 255 for privileged tasks.

Bits 16-23 User Status - As defined by sending and receiving tasks.

Bits 24-31 Options - This field contains user request control specification. It is bit encoded as follows:

Bit	Meaning When Set
24	Request is to be issued in no-wait mode.
25	Do not post completion status or accept return parameters. This bit is examined only if bit 24 is set. When this bit is set, the request is said to have been issued in the "no call-back mode".

3.4.3.2 Parameter Receive Block (PRB)

The Parameter Receive Block (PRB) is used to control the storage of passed parameters into the receiver buffer of the destination task. The same format PRB is used for both message and run requests. The address of the PRB must be presented when either the M.GMSGP or M.GRUNP services are invoked by the receiving task.

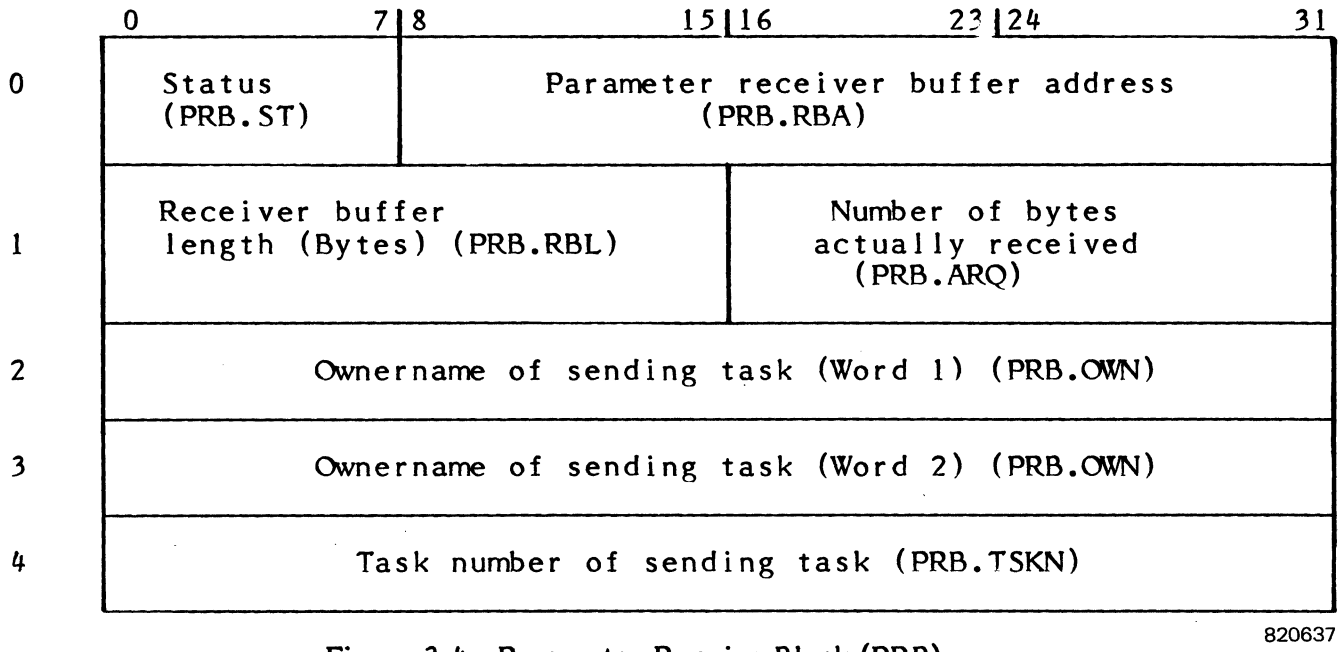


Figure 3-4. Parameter Receive Block (PRB)

WORD 0

Bits 0-7 Status-value encoded status byte

Code	Definition
0	Normal status
1	Reserved
2	Invalid receiver buffer address (PRB.RBAE)
3	No active send request (PRB.NSRE)
4	Receiver buffer length exceeded (PRB.RBLE)

Bits 8-31 Parameter Receiver Buffer Address - This field contains the word address of the buffer, into which the sent parameters are stored.

WORD 1

Bits 0-15 Receiver Buffer Length - Contains the length of the receiver buffer (number of bytes).

Bits 16-31 Number of Bytes Actually Received - This value is set by the operating system and is clamped to a maximum equal to the receiver buffer length.

WORDS 2,3

Bits 0-63 Ownername of Sending Task - Set by the operating system to contain the ownername of the task which issued the parameter send request.

WORD 4

Bits 0-31 Task Number of Sending Task - Set by the operating system to contain the task activation sequence number of the task which issued the parameter send request.

3.4.3.3 Receiver Exit Block (RXB)

The Receiver Exit Block (RXB) is used to control the return of parameters and status from the destination (receiving) task to the task that issued the send request. It is also used to specify receiver exit-type options. The same format RXB is used for both messages and run requests. The address of the RXB must be presented as an argument when either the M.XMSGR or M.XRUNR services are called.

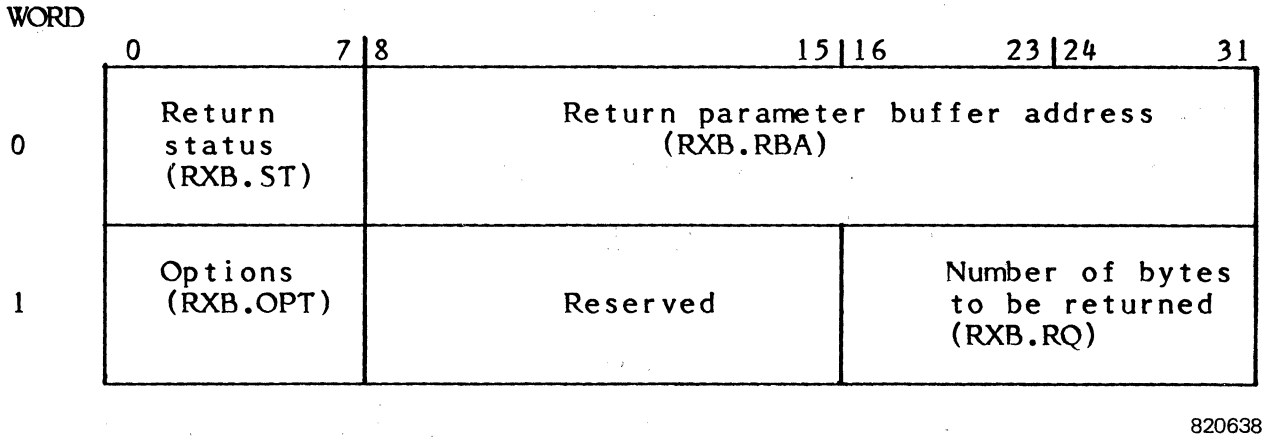


Figure 3-5. Receiver Exit Block (RXB)

WORD 0

Bits 0-7 Return Status - Contains status as defined by the receiver task. Used to set the user status byte in the Parameter Send Block (PSB) of the task which issued the send request.

Bits 8-31 Return Parameter Buffer Address - Contains the word address of the buffer containing the parameters which are to be returned to the task which issued the send request.

WORD 1

Bits 0-7 Options - This field contains receiver exit control options. It is encoded as follows:

Value	Exit Type	Meaning
0	M.XRUNR	Wait for next run request.
	M.XMSGR	Return to point of task interrupt.
1	M.XRUNR	Exit task, process any additional run requests. If none exist, perform a standard exit.
	M.XMSGR	N/A

Bits 8-15 Reserved.

Bits 16-31 Number of Bytes to be Returned - This field contains the number of bytes of information to be returned to the sending task.

3.4.4 Messages and Run Request Services Summary

The following table is provided as a summary of message and run request services provided by the MPX-32 system.

Table 3-1 Message and Run Request Services Summary

Run Request Services	Message Services	Function
Receiver Services		
N/A	M.RCVR recvaddr	Establish receiver address
M.GRUNP prbaddr	M.GMSGP prbaddr	Get parameters
M.XRUNR rxbaddr or M.EXIT	M.XMSGR rxbaddr	Exit receiver
N/A	M.ANYW time 1 or M.SUSP taskno,time1	Wait for receipt of next message
Send Services		
M.SRUNR psbaddr	M.SMSGR psbaddr	Send request
M.ANYW time1	M.ANYW time1 M.EAWAIT time 1	Wait for any request completion
M.XREA	M.XMEA	Exit user end action service

ARGUMENT	DESCRIPTION
recvaddr	Address of receiver
prbaddr	Address of Parameter Receive Block (PRB)
rxbaddr	Address of Receiver Exit Block (RXB)
psbaddr	Address of Parameter Send Block (PSB)
taskno	Contains zero
time1	Contains zero if indefinite wait, otherwise contains negative number of time units to be used as a wait time-out value

4. OPERATOR COMMUNICATIONS (OPCOM)

MPX-32 provides a set of commands that allow the user to exercise control over system operations either from the operator's console (OPCOM console) or from a terminal, where the user can run OPCOM interactively.

OPCOM commands allow the user to:

Activate and control system and user tasks

Activate and control batch jobs

Display system status information

Control peripherals associated with a batch job, user task, or system task

Display and access physical memory

Connect and disconnect tasks to/from interrupts

Set or delete timers to activate or resume tasks or request interrupts

Disable and re-enable hardware interrupt levels

4.1 Overview

This section describes general system characteristics that apply to use of OPCOM.

4.1.1 Task Names, Task Numbers, and Owners

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

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

A task name is the name supplied when a task is cataloged; it is also the name of the load module file that contains the task. More than one task with the same name can be active in the MPX-32 system at a time if it is cataloged as multicopied or shared.

The owner name for a task activated from a terminal is the name specified at logon. It cannot be changed except by logging on again. Normally, the logon owner name is associated with any task the owner activates on the system, except if batch or if the owner name is not restricted (see next sections). The OPCOM console has the default owner name CONSOLE.

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

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

The terminal user can be further restricted in his ability to control tasks. This is accomplished via the KEY utility described in Volume 2 and in Volume 1, Section 4.1.4. If access to tasks with other owner names is restricted, the user who issues a [RUN] OPCOM command from his terminal can control only those tasks which he has activated. If a terminal user issues a control command for a task that he does not own, the command is not accepted by OPCOM. If not restricted, an owner has the capabilities described next for CONSOLE.

The CONSOLE owner can activate a task under CONSOLE (by default) or under any owner name he chooses. The CONSOLE owner is not restricted in his ability to control tasks on the system (except by the multicopy and shared restrictions described previously).

4.1.2 Batch Jobs, Job Numbers, and Owner Names

OPCOM commands used to control batch jobs (REMOVE, REDIRECT, REPRINT, REPUNCH, DEPRINT, DEPUNCH, or URGENT) use job numbers to identify the job. If several jobs are submitted on a job file or device medium, a separate number is supplied for each occurrence of a $JOB job control statement. The job number is in a range from 0-9999, and uniquely identifies the job in the system.

Control commands for jobs do not have the same owner name restrictions that can be applied to tasks. Any owner can use the job number in the control commands above to control a job.

However, if you need to exercise control over a specific task within a job, you must be the CONSOLE owner name, have the owner name supplied on the $JOB card, or have the privilege of controlling tasks with other owner names. If a job is submitted from a terminal and the owner name is changed on the $JOB card, the $JOB owner name applies to the job only. The logon owner name is maintained for the terminal user. To issue OPCOM control commands pertaining to the task(s) activated by the job (see previous section), the restricted user must either use the OPCOM console (which has the owner name of CONSOLE and can access any task) or log off the terminal and log on again with the owner name used on the $JOB card.

4.1.3 Restricting OPCOM Commands

Any command verb can be restricted from a terminal user based on his owner name and OPCOM restriction bits in the M.KEY file. Each command is represented by a bit set in a particular position in the OPCOM restriction doubleword of the user's M.KEY entry. The module number of each command verb corresponds to its bit position. For example, the command verb "MODIFY" (module OC18), is represented by bit 18 of the OPCOM restriction doubleword. (Module numbers are indicated in the command summary chart at the beginning of Section 4.4.)

With the exception of EXIT (see Section 4.1.5), if a command verb bit is set, the command is not available to that owner name and generates an "INVALID COMMAND VERB" message. All commands are always available to the CONSOLE owner at the OPCOM console.

4.1.4 Restricting Owner Name Privileges

In addition to restricting use of OPCOM commands, the KEY utility can be used to restrict the following functions and privileges for an owner name:

- Access to tasks with different owner names can be prohibited.
- The ability to activate or establish tasks cataloged as privileged can be eliminated.

Like OPCOM commands, these privileges are removed by setting bits in the OPCOM restriction doubleword of the user's M.KEY entry. The same technique used for OPCOM commands is used for privileges. Bits 40-41 apply. See the KEY utility, Chapter 7, Volume 2.

4.1.5 The EXIT Command

The EXIT command (bit position 0 of the OPCOM restriction bits) cannot be restricted from terminal users. It should be set only to inhibit use of EXIT at the OPCOM console. If the EXIT bit is set and the OPCOM console operator issues an EXIT command, OPCOM issues a CR/line feed and suspends.

If the EXIT bit is not set, when the operator types EXIT, the TSM > prompt is returned, and the operator can use the console like a terminal, with one exception: system messages will be interspersed with any interactive I/O from the console. To reaccess OPCOM, type [RUN] OPCOM to the TSM > prompt.

Caution: if you type EXIT to the TSM > prompt, you will exit the system. To reactivate OPCOM at the console, depress the ATTENTION key. You will be prompted by TSM to log on. To obtain the normal capabilities of the OPCOM console, use the owner name CONSOLE at logon. OPCOM can be accessed for the console by:

```
TSM > [RUN] OPCOM
```

If a time out occurs while OPCOM is waiting for input, an exit occurs.

4.1.6 System Task Restrictions

System tasks J.SWAPR (the Swapper), J.TSM (Terminal Services Manager), and J.OPCOM (running on the terminal used to access OPCOM) cannot be activated nor controlled via OPCOM commands. The status of such tasks can however, be obtained via the LIST or STATUS commands.

4.2 Activating OPCOM

The OPCOM task is automatically active on the OPCOM console after a cold start, warm start, or restart. To enter a command, depress the ATTENTION key on the CPU front panel,

```
<Attention>
??
```

A double question mark prompt indicates that OPCOM is ready to accept a command. The ATTENTION key must be depressed before entering each unsolicited command. (See Section 4.3.)

To activate OPCOM from a terminal, respond OPCOM or RUN OPCOM to the TSM> prompt:

```
TSM>[RUN] OPCOM
??
```

The double question mark prompt is issued automatically by OPCOM after an OPCOM command has been processed.

Alternatively, an OPCOM command can be issued at the TSM prompt with immediate return to TSM rather than OPCOM. To do so, use an exclamation point as a synonym for 'OPCOM', followed by the OPCOM command verb:

```
TSM>!command
TSM>
```

For example:

```
TSM>!LIST
TSM>
```

4.3 Using OPCOM Commands

4.3.1 At the OPCOM Console

There are certain system messages and operations geared to the OPCOM console, and not to terminals running OPCOM, e.g.,

MOUNT and DISMOUNT messages for magnetic tape

Abort messages and codes for tasks running in the real time environment as described in Appendix C

I/O error conditions that can either be corrected offline with I/O resumed or aborted

Outputting these messages to the console assumes a central location of configured system hardware, including the OPCOM console, the CPU, printers, tape units, etc. User tasks can also output messages to the OPCOM console.

4.3.1.1 Information Messages

Commands or responses to prompts at the OPCOM console always take precedence over messages output to the console. The maximum uninterruptable output string for a message is 72 characters. The operator can depress the ATTENTION key to interrupt. The OPCOM prompt (??) will be issued after the current line of output is printed. The message is continued as soon as the OPCOM command has been processed.

4.3.1.2 Action Messages

If a system message requires operator intervention and reply, correct the condition noted if applicable and enter the reply directory, using a carriage return (CR) as the terminator. For example,

*CR7800 INOP : R,A? R <cr>

A card reader is inoperational. The operator fixes it and responds R and a carriage return for resume. Input continues.

4.3.1.3 Commands

Each complete input command is terminated by depressing a carriage return (CR). OPCOM checks the command verb and parameters. If correct, it completes the operation indicated. It acknowledges the command only after processing is complete (see 4.3.1.6).

4.3.1.4 Aborting a Command

If you depress the ATTENTION key after a command has been entered but before the command has been acknowleged, the command is processed up to the point of first output; output is suppressed, two asterisks (**) are printed to indicate suppression, and the OPCOM prompt (??) is returned. The command is processed completely if it produces no console output.

If you depress the ATTENTION key and then do not want to issue an OPCOM command, type just CR in response to the ?? prompt. OPCOM will suspend until ATTENTION is depressed.

4.3.1.5 Correcting Command Line Errors

Any typing error on the command line can be corrected by entering a Control H (CTRLH) or BACKSPACE key on the terminal. A command line can be erased completely by depressing the RUB key or DELETE key.

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

INVALID COMMAND VERB

and displays the OPCOM prompt (??). Re-enter the command.

If a command parameter is wrong, OPCOM displays an error message that identifies the invalid parameter. The ATTENTION key must be depressed before re-entering the command.

4.3.1.6 Command Processing

An OPCOM command that is processed to completion issues a carriage return/linefeed to the OPCOM console that acknowledges completion. If a command is legal but cannot be executed at this time, it issues the message:

REQUEST NOT EXECUTED

OPCOM always acknowledges a command with either a message or a CR/linefeed at the console.

4.3.2 At the Terminal

The terminal user obtains only the OPCOM messages that pertain to commands he issued from his or her terminal. Terminal users also do not have any equivalent to depressing the ATTENTION key. After issuing an OPCOM command successfully, the OPCOM prompt (??) is simply returned. The EXIT command is used at the terminal to return to the TSM >prompt.

4.3.2.1 Commands

Each complete input command is terminated by depressing a carriage return (CR). OPCOM checks the command verb and parameters. If correct, it completes the operation indicated. It acknowledges the command only after processing is complete. The acknowledgement is return of the OPCOM ?? or TSM > prompt.

4.3.2.2 Aborting a Command

To abort a command, depress the Break key or equivalent. If the command produces no terminal output, Break will have no effect. If the command produces output, the command is processed up to the point of first output, a double asterisk (**) is displayed to indicate suppression, and the OPCOM ?? or TSM > prompt is returned.

4.3.2.3 Command Line Errors

Any typing error on the command line can be corrected by entering a Control H (CTRLH) or BACKSPACE key on the terminal. A command line can be erased completely by depressing the RUB key or DELETE key.

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

INVALID COMMAND VERB

It then returns the OPCOM prompt (??) or the TSM > prompt so that you can reissue the command. If a command parameter is wrong, OPCOM displays an error message that identifies the invalid parameter. Reissue the command as described above.

4.3.2.4 Command Processing

An OPCOM command that is processed to completion issues a carriage return/linefeed and the ?? or TSM > prompt to acknowledge completion. If a command is legal but cannot be executed at this time, it issues the message:

REQUEST NOT EXECUTED

OPCOM always acknowledges a command with a message or a CR/line feed. It then reissues the ?? or TSM> prompt at the terminal.

4.4 OPCOM Commands

OPCOM commands are summarized below and described in detail in pages which follow. OPCOM commands may be abbreviated to the first four characters. A command verb is separated from associated parameters by one or more blanks or any valid delimiter (commas, parentheses, or equal signs). Where a comma is required, it is shown in the syntax statement.

Command	Function
ABORT (OC01)	Enters the specified task's abort receiver, if any. If none, deletes task.
ACTIVATE (OC02)	Activates a task.
BATCH (OC03)	Reads batch job(s) from specified device or file.
BREAK (OC04)	Enters pseudo interrupt receiver for specified task.
CONNECT (OC05)	Connects specified task to indirectly connected interrupt level defined via SYSGEN.
CONTINUE (OC06)	Releases task or device from HOLD.
DEBUG	Accesses the System Debugger. See Volume 3.
DELETETIMER (OC27)	Deletes specified timer attached to task.
DEPRINT (OC07)	Deletes specified SLO files (or current file only) from system output queue for specified job or task.
DEPUNCH (OC08)	Same as DELETEPRINT for SBO files.

DISABLE (OC09)	Disables an interrupt at specified priority level.
DISCONNECT (OC10)	Disconnects specified task from indirectly connected interrupt level defined at SYSGEN.
DUMP (OC11)	Dumps specified word locations to SLO.
ENABLE (OC12)	Enables an interrupt at specified priority level.
ENTER (OC13)	Updates system date and time.
ESTABLISH (OC39)	Activates and suspends a task.
EXIT (OC00)	Returns to TSM> prompt.
HOLD (OC14)	Inhibits a task from getting CPU control, stops spooled output to a printer, punch or magnetic tape, or stops spooled input from a card reader or magnetic tape until operator issues CONTINUE.
KILL (OC15)	Deletes specified task from system.
LIST (OC16)	Lists entries in system dispatch queue, output print or punch queue, account file, system patch file, or job queue as specified.
MODE (OC17)	Selects continuous batch, unidirectional file allocation, inhibits banner page on SLO, or locks OPCOM console.
MODIFY (OC18)	Changes a physical memory word in operating system address space.
OFFLINE (OC19)	Makes a device unavailable for allocation.

ONLINE (OC20)	Makes a device available for allocation.
PURGEAC (OC21)	Deletes current contents of accounting file M.ACCT.
REDIRECT (OC22)	Overrides SLO or SBO on job statement to SYSGEN'd LOD or POD device.
REMOVE (OC23)	Aborts batch task if active and removes the SYC file for the batch job.
REPRINT (OC24)	Reprints previous and current SLO files, current SLO, or all SLO files for a job.
REPUNCH (OC25)	Same as REPRINT for SBO files.
REQUEST (OC26)	Generates an RI (Request Interrupt) instruction for a specified interrupt priority level.
RESUME (OC38)	Resumes specified task.
SAVEAC (OC28)	Saves the current contents of M.ACCT on magnetic tape.
SEARCH (OC29)	Searches physical memory for a specified value.
SEND (OC30)	Sends a message to a task which has established a message receiver.
SETTIMER (OC31)	Sets specified timer for resumption of a task, activation of an established task, or execution of an RI instruction at an interrupt level.
SNAP (OC32)	Dumps physical word locations in physical memory to console.

START (OC33)	Resumes batch jobs after restart or warm start.
STATUS (OC34)	Lists amount of memory in current use, channel and device and I/O queue status information, device type and status information, or task attributes and current status.
SYSASSIGN (OC35)	Establishes availability of device for selection as SLO and/or SBO destination or temporarily overrides SYSGEN-selected SID device. Specifies use of device for batch output, real time output, or both.
TIME (OC36)	Prints time and date on terminal.
TURNON (OC40)	Activates a specified task at a specified time.
URGENT (OC37)	Changes the priority of a batch job.

4.4.1 ABORT Command

The ABORT command is used to abort the specified task. Only tasks found in the system dispatch queue can be aborted.

Syntax:

ABORT { T,taskname / taskno }

where:

taskname — specifies the name of the task (load module). If task name is used to abort a task, the task must be a unique copy load module. (See Section 4.4.1.)

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

Response:

If successful: LF (line feed)

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

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

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

If the task is scheduled to leave the system as a result of a previously issued ABORT command, the response is TASK SCHEDULED TO LEAVE SYSTEM.

Examples:

```
??ABORT T,PGMTEST
```

Comment:

If the task with taskname PGMTEST is a single copy load module, with no outstanding allocation requirements or outstanding I/O, it is aborted.

```
??ABORT 02000001
```

Comment:

If the task with task number 02000001 has no outstanding allocation requirements or outstanding I/O, it is aborted.

4.4.2 ACTIVATE Command

The ACTIVATE command is used to activate the specified task. System modules J.SWAPR and OPCOM cannot be activated by this command. ACTIVATE is used to initiate tasks independent of the interactive or batch environment, i.e., at their base priorities. The other alternative is to use the ESTABLISH command.

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

Syntax:

ACTIVATE loadmod [,ownername[,key]]

where:

loadmod — is the one- to eight-character name of the permanent load module file where the task is cataloged. It must be a system file (no user name). The name of the task and the name of the load module file are always identical.

ownername — is an optional one- to eight-character ownername. Default: logon ownername.

key — is the one- to eight-character key, if any, associated with the ownername in the M.KEY file.

Response:

LF (line feed)

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

If activation is not successful, e.g., if an assigned device is not configured in the system, an abort code and message are output to the OPCOM console.

Examples:

```
??ACTIVATE PGMTEST
```

Comment:

The permanent file with filename PGMTEST is activated. If OPCOM is running on behalf of a terminal user, PGMTEST is activated with the user's owner name, otherwise the task has CONSOLE as its owner.

??ACTI PGMTEST,GIPSON

Comment:

The permanent load module file named PGMTEST is activated. PGMTEST is activated with the owner name GIPSON.

4.4.3 BATCH Command

The BATCH command is used to read batch jobs from the current System Input Device (SID), specified device, or permanent file. The file must be a STOREd file.

Syntax:

```
BATCH  [ D,devmnc [,density,parity]                          ]
       [ F,jobfile  ,[username[,key]] [,password]            ]
```

where:

BATCH — if no parameters are specified, the job file is read from the System Input Device (SID).

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

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

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

jobfile — is the name of a permanent file from which batch input is to be read.

username — is a one- to eight-character user name associated with the jobfile. Default: if OPCOM console, system files; if a terminal, the last name specified or implied with the USERNAME command (TSM default is owner name = user name).

Any search for a file based on user name that fails (no match) is followed by search for a system file with the specified name.

key — is a one- to eight-character key, if any, associated with the user name in the M.KEY file.

password — is a one- to eight-character password associated with the specified permanent file for read access.

Response:

LF (line feed)

The job file is read to an SYC file, along with any additional records specified in $SELECT directives. When the SYC file is complete, the job is entered into the batchstream (dynamic job stream queue) at the current batch priority.

If continuous batch has not been specified with the MODE command, reading stops at the $$ statement. If continuous batch, continues reading until $$$ statement.

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

Examples:

??BATCH

Comment:

Batch jobs are read from the current System Input Device (SID).

??BATCH D,CR7A00

Comment:

Batch jobs are read from the card reader on channel 7A, subaddress 00.

??BATCH F,PGMTEST

Comment:

Batch jobs are read from the file named PGMTEST.

4.4.4 BREAK Command

The BREAK command is used to interrupt a task and enter the task's pseudo-interrupt receiver.

Syntax:

BREAK { T,taskname / taskno }

where:

taskname — is used to break into a task, the task must be a single copy load module.

taskno — is the task number assigned at activation.

Response:

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

If successful: LF (line feed)

Examples:

```
??BREAK T,PGMTEST
```

Comment:

If the task named PGMTEST is a unique load module with a break receiver, enters its break receiver.

```
??BREAK 02000001
```

Comment:

If the task with task number 02000001 has a break receiver, enters its break receiver.

4.4.5 CONNECT Command

The CONNECT command is used to connect the specified task to the specified interrupt level so that when the interrupt occurs, the task is resumed. The interrupt level must be described as indirect during SYSGEN.

Before using CONNECT, use the ACTIVATE or ESTABLISH command to activate the task.

Syntax:

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

where:

taskname — is the name of the load module file containing the task. If the task name is used to connect a task, the task must be a unique copy load module.

intlevel — is the 2-character hexadecimal interrupt priority level.

taskno — is the 8-digit task number assigned at activation.

Response:

LF (line feed)

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

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

The STATUS command can be used to indicate that a task is indirectly connected to an interrupt ('f' flag). However, the user is responsible for keeping track of what task is indirectly connected to a particular interrupt level.

Examples:

??CONNECT T,PGMTEST,2F

Comment:

If the task with taskname PGMTEST is a unique copy load module, it is connected to interrupt level '2F'. This level must be defined at SYSGEN for indirect connection.

??CONNECT 02000001,2F

Comment:

The task with task number 02000001 is connected to interrupt level '2F'.

4.4.6 CONTINUE Command

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

Syntax:

$$\underline{\text{CONTINUE}} \left\{ \begin{array}{l} \text{PRINT [,devmnc]} \\ \text{PUNCH [,devmnc]} \\ \text{READ [,devmnc]} \\ \text{T,taskname} \\ \text{taskno} \end{array} \right\}$$

where:

PRINT	if no device mnemonic, continues system task controlling SLO output to the system LOD device. Not valid under memory-only MPX-32.
PUNCH	if no device mnemonic, continues system task controlling SBO output to the system POD device. Not valid under memory-only MPX-32.
READ	if no device mnemonic, continues system task controlling input from the system SID device. Not valid under memory-only MPX-32.
devmnc	is a device mnemonic. Continues system output task controlling SLO (PRINT) or SBO (PUNCH) output to the specified device. Continues system input task controlling SID (READ) input from the specified device.
taskname	is the name of the load module containing the task. If task name is used to continue a task, the task must be a unique copy load module.
taskno	is the eight-digit task number assigned at activation.

Response:

LF (line feed)

The HOLD bit is turned off in the DQE for the task and the task continues at the address following the hold. (J.SSIN and J.SOUT system tasks control the device I/O described above.)

On a memory-only MPX-32 system, an entry other than a task name or task number is an error.

Examples:

```
??CONTINUE PRINT
```

Comment:

Output to the current system Listed Output Device (LOD) is continued.

??CONT READ,CR7801

Comment:

Input from the card reader on channel 78 with subaddress 01 is continued.

??CONT T,PGMTEST

Comment:

If the task with taskname PGMTEST is a unique copy load module, it is continued.

??CONT 02000001

Comment:

The task with task number 02000001 is continued.

4.4.7 DELETETIMER Command

The DELETETIMER command deletes the specified timer. If the timer is not in the system, the operator is informed. Timers are set up via the SETTIMER command.

Syntax:

DELETETIMER timer

where:

timer is the two-character ASCII name of the timer to be deleted.

Response:

If successful: LF (line feed)

4.4.8 DEPRINT Command

The DEPRINT command is used to:

delete the SLO file currently being output to a particular device

delete an SLO file generated for a particular task

delete all SLO files for a job

Syntax:

DEPRINT [J,jobno | T,taskname | taskno | D,devmnc]

where:

DEPRINT	if no parameters are specified, the SLO file currently being output to the system LOD device is deleted.
J,jobno	is the job sequence number (1-9999) assigned when the job was queued. All SLO files for the job are deleted.
T,taskname	is the name of the load module file containing the task. If the task name is used to delete the SLO file for a task, the task must be a unique copy load module.
taskno	is the eight-digit task number assigned at activation. Deletes the SLO file for the task.
D,devmnc	is a device mnemonic. Used to delete the SLO file currently being output on other than the SYSGEN-defined LOD device.

Response:

LF (line feed)

The system output task(s) associated with the task or job producing the SLO file(s) delete the file(s) from the system output queue. If a specified SLO file is being printed, printing stops.

Examples:

??DEPRINT J,700

Comment:

Deletes all SLO files for job number 700.

??DEPR T,PGMTEST

Comment:

If the task with task name PGMTEST is a single copy load module, all of its SLO files are deleted.

??DEPR 02000001

Comment:

Deletes all SLO files for task with task number 02000001.

??DEPR

Comment:

Deletes the SLO file currently being printed on the system Listed Output Device (LOD).

??DEPRINT D,LP7A00

Comment:

Deletes the SLO file currently being output to a printer on channel 7A, subaddress 00.

4.4.9 DEPUNCH Command

The DEPUNCH command is used to:

delete the SBO file currently being output to a particular device

delete an SBO file generated for a particular task

delete all SBO files generated for a job

Syntax:

DEPUNCH [J,jobno | T,taskname | taskno | D,devmnc]

where:

DEPUNCH	if no parameters are specified, the SBO file currently being output to the system POD device is deleted.
jobno	is the job sequence number (1-9999) assigned when the job was queued. All SBO files for the job are deleted.
T,taskname	is the name of the load module file containing the task. If task name is used to depunch a task, the task must be a unique copy load module.
taskno	is the eight-digit task number assigned at activation. The SBO file for the task is deleted.
D,devmnc	is a device mnemonic used to delete the SBO file currently being output on other than the SYSGEN-defined POD device.

Response:

If successful: LF (line feed)

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

Examples:

??DEPUNCH J,700

Comment:

Deletes all SBO files for job number 700.

??DEPU T,PGMTEST

Comment:

If the task with task name PGMTEST is a single copy load module, all of its SBO files are deleted.

??DEPU 02000001

Comment:

Deletes all SBO files for task with task number 02000001.

??DEPU

Comment:

Deletes the SBO file currently being punched on the current system Punch Output Device (POD).

??DEPUNCH D,CP7400

Comment:

Deletes the SBO file currently being punched on the card punch, channel 74 with subaddress 00.

4.4.10 DISABLE Command

The DISABLE command causes a Disable Interrupt (DI) instruction to be executed for the specified hardware interrupt level.

The CPU will not respond to an external interrupt signal at the specified level until an ENABLE Interrupt (EI) instruction is received (see the ENABLE command).

Syntax:

DISABLE intlevel

where:

intlevel — is the two-character hexadecimal interrupt priority in the range 00 to 7F.

Response:

LF (line feed)

If a hardware interrupt level is RTOM-jumpered for constant enable (as opposed to software enable/disable), this command is not honored.

4.4.11 DISCONNECT Command

The DISCONNECT command disconnects the specified task from its indirectly connected interrupt level.

Syntax:

DISCONNECT { T,taskname / taskno }

where:

taskname — is the name of the load module containing the task. If the task name is used to disconnect a task, the task must be a unique copy load module.

taskno — is the task number assigned at activation time.

Response:

LF (line feed)

Examples:

??DISCONNECT T,PGMTEST

Comment:

If the task with task name PGMTEST is a unique copy load module, it is disconnected from its indirectly connected interrupt level.

??DISCONNECT 02000001

Comment:

The task with task number 02000001 is disconnected from its indirectly connected interrupt level.

4.4.12 DUMP Command

The DUMP command is used to output the word locations specified by the starting and ending physical addresses. OPCOM dynamically allocates an SLO file for output. Output is listed in side-by-side ASCII-coded hexadecimal with ASCII format.

DUMP can also be used for automated dump on abort of a task running independent of the batch or interactive environment.

Syntax:

DUMP { start,end / ON / OFF }

where:

start	is the starting hexadecimal physical word address.
end	is the ending hexadecimal physical word address.
ON	indicates that a dump is required on abort of an independent task.
OFF	indicates that an independent task abort dump is not required. The indication may have been previously set (=ON) via a DUMP ON command.

Response:

LF (line feed) upon completion of the request

or

"DUMP NOT PERFORMED-TRY AGAIN LATER" if an SLO file cannot be allocated dynamically at the time of the request.

Examples:

??DUMP 3000,3FFF

Comment:

Dumps the contents of physical memory between logical address 30000 and logical address 3FFF to the SLO file.

4.4.13 ENABLE Command

The ENABLE command is used to execute an Enable Interrupt (EI) instruction for the specified hardware interrupt level. The level will respond to external interrupts generated by an associated device. (Hardware interrupt levels are automatically enabled by defining them at SYSGEN.)

Syntax:

ENABLE intlevel

where:

intlevel is the two-character hexadecimal interrupt priority level.

Response:

LF (line feed)

The specified hardware interrupt level is enabled. If directly connected, the privileged user task associated at SYSGEN will execute on an interrupt.

If indirectly connected, the task connected, if any, will resume on the interrupt. If a timer is set to generate an RI instruction on timeout, the specified interrupt will respond to the instruction.

4.4.14 ENTER Command

The ENTER command updates the date and time currently stored by the system.

Syntax:

ENTER month/day/year,hour:minute:second

where:

month	is the two-digit decimal month
day	is the two-digit decimal day
year	is the two-digit decimal year
hour	is the two-digit decimal hour
minute	is the two-digit decimal minute
second	is the two-digit decimal second

Response:

LF (line feed)

Example:

??ENTER 10/16/78,13:15:43

4.4.15 ESTABLISH Command

The ESTABLISH command activates the specified task and suspends it at the end of the activation sequence. Tasks that are established remain inactive until they are activated by a timer (see the SETTIMER command), connected to an indirectly connected interrupt level (see the CONNECT command), or resumed (see the RESUME command). When activated, they are brought into execution at their base (cataloged) priority.

This command enables a user task to activate and suspend for resumption by a timer, interrupt, or RESUME command without building the suspension into the task itself. ESTABLISH also allows the task to resume with all devices and memory allocation complete. (For further description, see Section 2.1.2.)

If a task activated with ESTABLISH is defined as RESIDENT when it is cataloged, it is not swappable; otherwise it can be swapped.

System modules J.SWAPR and J.OPCOM cannot be established through this service.

Syntax:

ESTABLISH loadmod [,ownername [,key]]

where:

loadmod	is the one- to eight-character name of the permanent load module file containing the cataloged task. It must be a system file (no user name).
ownername	is an optional one- to eight-character owner name for the task. Default: logon owner name (and key).
key	if ownername is used, supplies the one- to eight-character key associated with the ownername for a user if one has been established in the M.KEY file.

Response:

LF (line feed)

The task is activated, then suspended.

Examples:

??ESTABLISH PGMTEST

Comment:

The permanent load module file named PGMTEST is established. PGMTEST is established with the logon owner name. (See Section 4.1.1.)

??ESTA PGMTEST,GIPSON

Comment:

The permanent load module file named PGMTEST is established. PGMTEST is established with GIPSON as its owner.

4.4.16 EXIT Command

The EXIT command is used to terminate OPCOM and return to the TSM prompt. It cannot be used at the OPCOM console if the EXIT bit has been set. (See Section 4.1.5.)

Syntax:

EXIT

Response:

TSM >

4.4.17 HOLD Command

The HOLD command holds a system output task, system input task, or a specified user or system task.

Syntax:

$$\text{HOLD} \left\{ \begin{array}{l} \text{PRINT [,devmnc]} \\ \text{PUNCH [,devmnc]} \\ \text{READ [,devmnc]} \\ \text{T,taskname} \\ \text{taskno} \end{array} \right\}$$

where:

PRINT	if no device mnemonic, holds the system task controlling SLO output to the system LOD device. Not valid under memory-only MPX-32.
PUNCH	if no device mnemonic, holds the system task controlling SBO output to the system POD device. Not valid under memory-only MPX-32.
READ	if no device mnemonic, holds the system task reading input from the system SID device. Not valid under memory-only MPX-32.
devmnc	is a device mnemonic. Holds system task controlling SLO or SBO output to the specified device or reading input from the specified input device.
taskname	is the name of the load module containing the task. If task name is used to hold a task, the task must be a unique copy load module.
taskno	is the task number assigned at activation.

Response:

A hold bit is turned on in the DQE for the task. Its current status is retained so that it can be continued where it left off.

LF (line feed)

On a memory-only MPX-32 system, an entry other than a task name or task number is an error.

If the task is scheduled to leave the system the response is: TASK SCHEDULED TO LEAVE SYSTEM and the task is not held.

Examples:

```
??HOLD PRINT
```

Comment:

Output to the current system Listed Output Device (LOD) is held.

??HOLD READ,CR7801

Comment:

Input from the card reader device on channel 78 with subaddress 01 is held.

??HOLD T,PGMTEST

Comment:

If the task with task name PGMTEST is a unique copy load module, it is held.

??HOLD 02000001

Comment:

The task with task number 02000001 is held.

4.4.18 KILL Command

The KILL command is used to terminate all outstanding I/O requests and run requests, and delete the specified task from the system dispatch queue.

Syntax:

KILL $\begin{Bmatrix} T,taskname \\ taskno \end{Bmatrix}$

where:

taskname — is the name of the load module that contains the task. If task name is used to kill a task, the task must be a unique copy load module.

taskno — is the task number assigned at activation time.

Response:

Processing is not deferred for outstanding I/O or run requests. User files and devices are closed and deallocated. The DQE for the task is deleted.

LF (line feed)

Examples:

??KILL T,PGMTEST

Comment:

If the task with task name PGMTEST is a unique copy load module, it is killed.

??KILL 02000001

Comment:

The task with task number 02000001 is killed.

WARNING: Use of this command may impact integrity of blocked files. Blocking buffers are not purged, and end-of-files (EOFs) are not written. Also, any I/O pending can be incomplete.

The only exception to this is for SLO/SBO where an end of file is written if the file is output active.

4.4.19 LIST Command

The LIST command is used to display the entries in:

the system dispatch queue (all or selectively)

system output print queue

system output punch queue

accounting file

system patch file (M.PATCH)

job queue as specified

Syntax:

```
                ┌ EXECUTION                                                          ┐
                │ PRINT                                                              │
                │ PUNCH                                                              │
LIST            │ ACCOUNT  [,OWNE=name] [,PROJ=proj] [,DATE=date]                    │
                │                [ORIG= {TSM.nnnn}]                                  │
                │                       {BATCH }                                     │
                │ PATCHES                                                            │
                │ JOBS                                                               │
                └ taskname , [ownername] [,pseudonym]                                ┘
```

where:

LIST — with no parameters, lists all entries in the System Dispatch Queue.

EXECUTION — same as LIST above.

PRINT — specifies entries in the SLO file output queue for all jobs. Not valid under memory-only MPX-32.

PUNCH — same as PRINT for SBO file output queue. Not valid under memory-only MPX-32.

ACCOUNT — copies contents of Job Accounting file to an SLO file. Not valid under memory-only MPX-32.

PATCHES — copies contents of System Patch file to an SLO file. Not valid under memory-only MPX-32.

JOBS — lists all jobs waiting or active on system. Not valid under memory-only MPX-32.

In response to LIST with one or more of the task identifiers task name, owner name, and/or pseudonym, OPCOM displays the status of task(s). Any one of these parameters, or any combination of these parameters can be used to select tasks; however, they must

be entered in the order shown in the syntax statement, supplying a comma for any missing parameter.

On a memory-only MPX-32 system, an entry other than EXECUTION or a task name is an error.

taskname — specifies the name of the load module containing a task. If not specified, all tasks with the specified owner and/or pseudonym are listed.

ownername — specifies the owner name for a particular task name or all tasks belonging to the owner. If task name has not been specified, the comma must still be used, e.g.,

```
LIST ,ownername
```

If ownername is not specified, all tasks with the specified task name and/or pseudonym are listed.

pseudonym — specifies the pseudonym for a task. A pseudonym is established by some system tasks (e.g., TSM uses TSM*terminal number, and job control uses .Odevmnc) and can be established by user tasks. The pseudonym allows identification of a particular copy of a task without the task number.

For example, a TSM pseudonym allows you to identify the TSM copy for a particular terminal and in so doing, see what task and owner are currently active on the specified terminal.

If task name and owner name are not specified, two commas must precede the pseudonym, e.g.,

```
LIST ,,pseudonym
```

Response:

In response to LIST with no parameters, all entries in the System Dispatch Queue (DQE's) are displayed; same as LIST EXECUTION, next.

In response to LIST EXECUTION or tasks selected by task identifiers:

```
taskno taskname ownername pseudonym priority state swap
(for all tasks on system or each task selected)
```

where:

taskno — is the task number assigned at activation.

taskname — is the name of the load module containing the task.

ownername — owner of the task, i.e., the owner who activated it.

pseudonym	pseudonym name of task. The pseudonym allows identification of a particular copy of a task without the task number.
priority	is the current software priority level, from 1 to 64.
state	is a four-character identifier corresponding to the state of the task. See STATUS command in this section, description of response to STATUS T, for details.
swap	is the swap status of the task: IN means the task is in memory OUT means the task is outswapped

In response to LIST PRINT or LIST PUNCH:

jobno jobname files pseudonym
(for SLO or SBO files generated by batch jobs)

taskno pseudonym
(for each SLO or SBO file generated by a task running independent of the batch or interactive environment)

where:

jobno	is the job's sequence number.
jobname	is the one- to eight-character job name as specified on the $JOB statement.
files	is the number of SLO or SBO files generated by the job.
pseudonym	is the pseudonym of the J.SOUT task that is currently processing the SLO or SBO files, e.g., .Odevmnc.
taskno	is the task number of the task within the job which created the SLO or SBO file.

In response to LIST ACCOUNT:

With no parameters, outputs the current contents of the Accounting file on an SLO file.

In response to LIST ACCOUNT with one or more keywords specified, only statistics of requested data is output. Keyword parameters can contain leading or trailing wild card characters in the form of question marks (?).

OWNE=name	specifies the 1-8 character owner name associated with the file
PROJ=proj	specifies the 1-8 character alphanumeric project name/number associated with the file
DATE=date	specifies the 8 character numeric date associated with the file, entered in the format month/day/year
ORIG={TSM.nnnn / BATCH}	specifies the mode of operation in which the file was used

Note: When a special data type is requested, the keyword must be specified along with the pertinent information.

The ORIG= keyword can be used in one of six ways:

ORIG=TSM	ouptuts statistics of all jobs run under TSM
ORIG=TSM.20??	output statistics of all jobs run under TSM on devices 20xx
ORIG=TSM.2009	outputs statistics of all jobs run under TSM on device 2009
ORIG=BATCH	outputs statistics of all jobs run under Batch
ORIG=JOB#20??	outputs statistics of all jobs run under Batch with job numbers 20xx
ORIG=JOB#2033	outputs statistics of job number 2003 run under Batch

Examples:

```
LIST ACCO,OWNE=MCNORTON,DATE=01/22/80,ORIG=TSM.20??
```

Outputs statistics on all jobs with owner name MCNORTON run on January 22, 1980 on any TSM device 20xx.

```
LIST ACCO,OWNE=DRUCKER,PROJ=087212
```

Outputs statistics on all jobs with owner name DRUCKER and project number 087212.

In response to LIST PATCHES:

Outputs the contents of the System Patch file to an SLO file for printing.

In response to LIST JOBS:

```
jobno ownername jobname priority taskno SLO=pseudonym SBO=pseudonym
```

where:

jobno	is the job's sequence number.
ownername	owner name for the job as specified on the $JOB statement.
jobname	is the one- to eight-character job name as specified on the $JOB statement.
priority	is the job's software priority (1-64).
taskno	is the task number of the last task in the job that was activated or is "WAITING".
SLO/SBO	these fields are printed only if SLO and/or SBO assignments are specified for the job.
pseudonym	indicates the pseudonym of the J.SOUT task processing the job's SLO or SBO files. The pseudonym indicates the destination device mnemonic, e.g., .OLP0100. If the destination is a permanent file, this field contains the permanent file name.

Examples:

```
??LIST PGMTEST
```

Comment:

Lists all entries in the dispatch queue with task name PGMTEST.

```
??LIST PGMTEST,GIPSON
```

Comment:

Lists all entries in the dispatch queue with task name PGMTEST and ownername GIPSON.

??LIST ,GIPSON

Comment:

Lists all entries in the dispatch queue with ownername GIPSON.

??LIST

Comment:

Lists all entries in the dispatch queue.

4.4.20 MODE Command

The MODE command is used to define the following special system operations.

Continuous Batch - Batchstream input from SID is processed until the job control statement, $$$, is encountered. All $$ job control statements are ignored.

Inhibit Banner Page - Suppresses the banner page which is produced by system output tasks when processing SLO files.

OPCOM Console Lock - Allows entering continuous OPCOM commands without intervening Console Interrupts.

Uni-Directional File Allocation - Treats all requests for temporary disc space as permanent file requests, i.e., space is allocated from the high end of the disc downward.

Inhibit Mount Message - Suppresses the mount message produced by the system when a single magnetic tape is assigned for I/O. The mount message will, however, be displayed when a multivolume magnetic tape operation is assigned for I/O.

Syntax:

MODE { SCBT | RCBT | SIBP | RIBP | SOPC | ROPC | SUFA | RUFA | SIMM | RIMM }

where:

SCBT	sets continuous batch
RCBT	resets continuous batch
SIBP	sets inhibit banner page
RIBP	resets inhibit banner page
SOPC	sets OPCOM console lock
ROPC	resets OPCOM console lock
SUFA	sets uni-directional file allocation
RUFA	resets uni-directional file allocation
SIMM	sets inhibit magnetic tape mount message
RIMM	resets inhibit magnetic tape mount message

Response:

LF (line feed)

4.4.21 MODIFY Command

The MODIFY command is used to reset a memory word at the specified physical address to the specified value. A mask can be used to modify only some bit positions in the word and leave the others alone. If no mask is specified, a mask of binary zeroes is used, i.e., all bits are to be reset as indicated by the specified value.

The MPX Debugger can be used to access and modify locations in a task's logical address space using logical addressing.

If a mask is used, a "logical AND" operation is performed between the specified memory word and the mask word, followed by a "logical OR" operation performed between the result of the "logical AND" and the specified value. This result is stored in the specified memory word.

Syntax:

MODIFY address,value[,mask]

where:

address	is the hexadecimal physical word address.
value	is the hexadecimal value of the word.
mask	is the hexadecimal word mask. (The absence of this argument results in the use of a mask of binary zeros.)

Response:

LF (line feed)

Examples:

??MODIFY 3000,52535253

Comment:

The hex value '52535253' will be stored at physical location 3000.

??MODI 12000,52535253,00000000

Comment:

The hex value '52535253' will be stored at physical location 12000.

??MODIFY 3004,52530000, 0000FFFF

Comment:

The upper half word of physical location 3004 is changed to '5253', while the lower half word is unchanged.

??MODI 12004,00530055,FF00FFFF

Comment:

Byte one of the word at physical location 12004 is changed to '53'. Byte three of the word at physical location 12004 is "logically OR'd" with '55' and the result stored at that location. Bytes zero and two of the word are unchanged.

4.4.22 OFFLINE Command

The OFFLINE command is used to inhibit the specified device from all further allocation. The device can be brought back online with the ONLINE command.

Syntax:

OFFLINE devmnc

where:

devmnc is a device mnemonic.

Response:

LF (line feed)

Any task that has assigned a specific device that is offline will generate an abort error. If the device is a system device (LOD, POD, or SID) it will be bypassed for auto selection; if another device is SYSGENed or SYSASSIGN'd as available for auto selection, it will be used automatically so that tasks producing SLO and SBO output can proceed. If no other device is SYSGENed for auto selection, the SYSASSIGN command can be used to do so.

The REDIRECT command can also be used to divert output from a batch job to the LOD or POD device if needed.

Example:

??OFFLINE LP7A

Comment:

Line printer on channel 7A with subaddress 00 is taken offline.

4.4.23 ONLINE Command

The ONLINE command makes the specified device available for allocation.

Syntax:

ONLINE devmnc

where:

devmnc is a device mnemonic.

Response:

LF (line feed)

Example:

??ONLINE LP7A

Comment:

Line printer on channel 7A with subaddress 00 is placed online.

4.4.24 PURGEAC Command

The PURGEAC command is used to delete the current contents of the accounting file, M.ACCNT.

Syntax:

PURGEAC

Response:

LF (line feed)

4.4.25 REDIRECT Command

The REDIRECT command is used to override the destination device for SLO or SBO files specified on a $JOB statement. The job's SLO or SBO files are redirected to the system Listed Output Device (LOD) or Punched Output Device (POD) as specified by the SYSGEN DEV directive. Any completed output is reprocessed.

Syntax:

REDIRECT jobno [B]

where:

jobno	is the job's sequence number. The job must be active, its SLO or SBO files must be queued, or output must be in progress.
B	specifies the job's SBO files. If not entered, SLO files are assumed.

Response:

LF (line feed)

Examples:

??REDIRECT 700

Comment:

The SLO files for the job with job number 700 are redirected to the system Listed Output Device (LOD).

??REDIRECT 700 B

Comment:

The SBO files for the job with job number 700 are redirected to the system Punched Output Device (POD).

4.4.26 REMOVE Command

The REMOVE command terminates any further processing of the specified job. The job need not be active. If the job is active, any task executing within the job is killed (this includes the termination of all outstanding I/O requests and run requests). Output that is already spooled to an SLO or SBO file is processed normally.

Syntax:

REMOVE jobno

where:

jobno is the job's sequence number

Response:

LF (line feed)

Example:

??REMOVE 700

Comment:

The job with job number 700 is removed from the system.

WARNING

Use of this command may impact integrity of blocked files. Blocking buffers are not purged, and end-of-files (EOFs) are not written. Also, any I/O pending can be incomplete.

The only exception to this is for SLO/SBO where an end-of-file is written if the file is output active.

4.4.27 REPRINT Command

The REPRINT command is used to:

reprint the SLO file currently being output on a particular device

reprint the current and previous SLO file output on a particular device

reactivate a spooled output task that has aborted

reprint all SLO files for a particular job

Syntax:

REPRINT [[CURR / JOB,jobno] [,PAGE,pageno / ,times] ,] [devmnc]

where:

REPRINT	if no parameters are specified with the REPRINT command, the previous and current SLO files being output to the SYSGEN-defined LOD device are reprinted. SLO files being output on other devices can be specified by providing the device mnemonic. (See devmnc.)
CURR	optionally limits reprinting to current SLO file only. Default: current and previous.
times	can optionally specify number of times to reprint the current file (1-12). Default is 1.
PAGE	can optionally specify a page number on the current listing where reprinting should start. Default is page 1.
pageno	specifies the decimal page number (1-32767).
devmnc	used to reprint SLO files being output on other than the SYSGEN-defined LOD device. Supply a device mnemonic, e.g., 'MT1000,SAVE'. See Appendix A, which describes MPX device addressing. For this command, a channel and subaddress must be specified.

JOB — used to reprint all SLO files for a particular job. Output from the specified job must be in process, or the command is ignored.

jobno — specifies the job sequence number (1-9999) assigned when the job was started and shown on the output.

Response:

LF (line feed)

Printing always begins at the beginning of the specified SLO file(s) unless overridden by the PAGE parameter. The REPRINT JOB command is ignored if the job's SLO files are not currently being printed.

Examples:

The LOD device malfunctions in the middle of printing UPDATE diagnostics. You fix it, then:

```
REPRINT CURR, PAGE,5
```

PAGE is used to start printing at the point where the malfunction occurred.

You want two copies of the SLO file currently being output to the LOD device:

```
REPRINT CURR,2
```

It stops printing the file, then reprints two copies.

The LOD device malfunctions, you fix it, and you want to reprint SLO files at the beginning of the job that was interrupted:

```
REPRINT JOB,54
```

A system output task is aborted before or while printing SLO files.

```
REPRINT LP0700
```

The system output task is reactivated and all available SLO output for this device is printed.

4.4.28 REPUNCH Command

The REPUNCH command is used to:

repunch the SBO file currently being output on a particular device

repunch the current and previous SBO file output on a particular device

reactivate a spooled output task that has aborted

repunch all SBO files for a particular job

Syntax:

REPUNCH [[CURR] [,devmnc] | JOB,jobno]

where:

REPUNCH — if no parameters are specified with the REPUNCH command, the previous and current SBO files being output to the SYSGEN-defined POD device are repunched. SBO files being output on other devices can be specified by providing the device mnemonic. (See devmnc.)

CURR — optionally limits reprinting to current SBO file only. Default: current and previous. Use comma if supplying a device mnemonic.

devmnc — used to reprint SBO files being output on other than the SYSGEN-defined POD device. Supply a device mnemonic, e.g., 'MT1000,SAVE'. See Appendix A, which describes MPX device addressing. For this command, a channel and subaddress must be specified. Do not use a comma before devmnc if CURR is not specified.

JOB — used to reprint all SBO files for a particular job. Output from the specified job must be in process, or the command is ignored.

jobno — specifies the job sequence number (1-9999) assigned when the job was started and shown on the output.

Response:

LF (line feed)

Punching always begins at the beginning of the specified SBO file(s). The REPUNCH JOB command is ignored if the job's SBO files are not currently being punched.

Examples:

The POD device malfunctions in the middle of a job. You fix it, then:

```
REPUNCH CURR
```

The POD device malfunctions, you fix it, and you want to repunch SBO files at the beginning of the job that was interrupted:

```
REPUNCH JOB,54
```

A system output task is aborted before or while punching SBO files.

```
REPUNCH LP0700
```

The system output task is reactivated and all variables SBO output for this device is printed.

4.4.29 REQUEST Command

The REQUEST command causes a Request Interrupt (RI) instruction to be executed for the specified interrupt priority level. See also ENABLE and SETTIMER command descriptions.

Syntax:

REQUEST intlevel

where:

intlevel is the two-character hexadecimal interrupt priority level.

Response:

LF (line feed)

4.4.30 RESUME Command

The RESUME command is used to resume execution of the specified task. The task must be suspended in order to be resumed. A task can also be resumed via a timer or an interrupt (see SETTIMER and CONNECT).

Syntax:

RESUME { T,taskname / taskno }

where:

taskname — is the name of the load module containing the task. If task name is used to resume a task, the task must be a unique copy load module.

taskno — is the task number assigned at activation.

Response:

LF (line feed)

A task can be suspended awaiting a timeout, an interrupt, or resumption by another task. This command inswaps the task if needed and moves it into the ready to run queue.

If the task is not suspended, the command is ignored.

Examples:

```
??RESUME T,PGMTEST
```

Comment:

If the task with taskname PGMTEST is a unique copy load module, it is resumed.

```
??RESUME 02000001
```

Comment:

The task with task number 02000001 is resumed.

4.4.31 SAVEAC Command

The SAVEAC command provides for saving the current contents of the accounting file, M.ACCNT, on magnetic tape. A message is output to the operator's console specifying that the reel identified as "ACCT" be mounted on the magnetic tape unit specified by the device mnemonic.

Syntax:

SAVEAC devmnc

where:

devmnc is the device mnemonic of a magnetic tape unit. See Appendix A.

Response:

LF (line feed)

SAVE COMPLETED (Contents of accounting file have been saved.)

UNABLE TO ALLOCATE DEVICE (Specified device is unavailable for system use.)

4.4.32 SEARCH Command

The SEARCH command is used to search memory within the specified physical addresses for a value under control of a mask. A "logical AND" operation is performed between the memory word and the task word. The result is compared to the value, and if equal, the memory address and contents are printed on the OPCOM console.

The Debugger can be used to search memory locations in a task's logical address space using logical addressing. (See Volume 2.)

Syntax:

SEARCH start,end,value [,mask]

where:

start	is the starting hexadecimal physical word address.
end	is the ending hexadecimal physical word address.
value	is the hexadecimal value used in comparison.
mask	is the hexadecimal word mask. (Absence of this argument results in the use of a mask of binary ones.)

Response:

LF (line feed)

address content (repeated for each successful comparison)

where:

address — is the hexadecimal physical word address.

content — is the hexadecimal word content.

Examples:

??SEARCH 3000,3FFF,52535253

Comment:

All words between physical locations 3000 and 3FFF with the value '52535253' will be displayed (along with its location).

??SEARCH 12000,12FFF,52535253,FFFFFFFF

Comment:

All words, between physical locations 12000 and 12FFF with the value '52535253' will be displayed (along with its location).

??SEARCH 12000,12FFF,00530000,00FF0000

Comment:

All words between physical locations 12000 and 12FFF with the value '53' in byte one will be displayed (along with location).

4.4.33 SEND Command

The SEND command is used to send a message to a task which has established a message receiver.

Syntax:

SEND {T,taskname / taskno} [,message]

where:

taskname — is the name of the load module containing the task. If task name is used to send a message to a task, the task must be a unique copy load module.

taskno — is the task number assigned at activation.

message — is the message to be sent to the specified task. It may optionally be typed on the same line, in which case no response will be given. If more space is needed for a message, omit it from the command line, and receive the following response:

"ENTER MESSAGE"
??

Type the message in response to the OPCOM?? prompt (72 characters maximum), terminated by a CR. The message will be transmitted.

Examples:

```
??SEND T,PGMTEST
"ENTER MESSAGE"
??THIS IS A TEST
```

Comment:

If the task with task name PGMTEST is a unique copy load module and has a message receiver, the message 'THIS IS IS A TEST' is sent to its receiver buffer address.

```
??SEND 02000001,THIS IS A TEST
```

Comment:

The message is short enough to be included on the command line. If the task with task number 02000001 has a message receiver, the message 'THIS IS A TEST' is sent to its receiver buffer address.

Operator at OPCOM console types:

```
?? LIST J.OPCOM
   03000004 J.OPCOM SYSTEM  62 TD T
   04000009 J.OPCOM GIPSON  62 TD T


??SEND 04000009
"ENTER MESSAGE"
??EXIT
```

At owner GIPSON terminal:

```
?? TIME
   01/29/78 11:59:35
**EXIT
TSM>
```

Comment:

The operator at the OPCOM console can send OPCOM commands to a terminal user's interactive OPCOM task. The command is queued and processed after the user's current command is complete as shown above.

Owners who are not restricted in their ability to access tasks with different owner names can also use this capability.

4.4.34 SETTIMER Command

The SETTIMER command is used to cause one of three types of events at specified time intervals:

a task can be activated

a task can be resumed

a Request Interrupt (RI) instruction can be executed for a hardware interrupt level.

The time intervals are specified in terms of time units. The duration of a time unit is defined by SYSGEN, based on the NTIM and MTIM directives.

Syntax:

SETTIMER timer,t1,t2, { ACP,loadmod / RST,taskno / RQI,intlevel }

where:

timer	is the two-character ASCII name of the timer being created.
t1	is the decimal number of time units until the first timeout of this timer.
t2	is the decimal number of time units used to reset this timer upon each timeout. If "t2" is zero, the timer will timeout only once.
ACP	means the timeout event is to be the activation of the specified task.
loadmod	is the name of the load module file containing the task.
RST	means the timeout event is to be the resumption of the specified task.
taskno	is the eight-digit task number assigned at activation.
RQI	means the timeout event is the execution of a Request Interrupt (RI) for the specified interrupt level.
intlevel	is the two-character hexadecimal interrupt priority level.

Response:

LF (line feed)

Examples:

??SETTIMER AB,2,0,ACP,PGMTEST

Comment:

The file with filename PGMTEST is activated when its associated timer (AB) expires. The timeout is set to occur only once.

??SETTIMER AB,2,3,RST,02000001

Comment:

The task with task number 02000001 is resumed when its associated timer (AB) expires. The timer AB is reset to 3 upon each following time event.

??SETTIMER AB,2,0,RQI,2F

Comment:

A Request Interrupt (RI) is issued for interrupt level 2F when its associated timer (AB) expires. The timeout is set to occur only once.

4.4.35 SNAP Command

The SNAP command dumps the word locations specified by the starting and ending physical address to the OPCOM console.

The Debugger can be used to snap memory locations in a task's logical address space using logical addresses. (See Volume 2.)

Syntax:

SNAP start,end

where:

start	is the starting hexadecimal physical word address.
end	is the ending hexadecimal physical word address.

Response:

LF (line feed)

Address content (repeated for each successful comparison)

where:

address	is the starting hexadecimal physical word address of the current line.
content	is the hexadecimal word content.

Examples:

??SNAP 3000,3FFF

Comment:

Displays the contents between physical locations 3000 to 3FFF.

4.4.36 START Command

The START command is used after a system restart or warm start to initiate processing of batch jobs which remain buffered on disc from the time of the prior system failure. The command is also used to resume processing an active job when its associated job control task terminates abnormally.

Syntax:

START

Response:

LF (line feed)

Any active job is resumed. System tasks that control output accumulation and queuing for temporary files such as SLO and SBO are also resumed.

4.4.37 STATUS Command

The STATUS MEM command displays a current memory utilization map.

The STATUS CHA command returns information relating to a particular I/O channel, including the number of controllers, the number of devices attached, and the number of I/O entries currently queued on the channel. The status of each device connected to the channel is also displayed.

The STATUS DEVICE command displays information such as device type and device status for the specified device.

The STATUS T command or the STATUS command with a task number returns the current status of the task and its significant attributes.

The STATUS IPU command returns information relating to the IPU if one is configured on the system.

Syntax:

STATUS { MEM [,class] | CHA,address | D,devmnc | T,taskname | taskno | IPU }

where:

MEM	with no class, displays physical address boundaries and other information for each class of memory.
class	restricts display to a particular memory class: E - memory below 128KW H - high speed memory above 128KW S - slow speed memory above 128KW
address	is a two-character hexadecimal channel number or GPMC device address in the range 80-FF.
devmnc	is a device mnemonic. See Appendix A.
taskname	is the name of the load module containing the task.
taskno	is the task number assigned at activation.
IPU	returns online or offline status and which task, if any is running.

Response:

In response to STATUS MEM:

MEM low -- high $\left\{\begin{matrix}\text{FREE}\\ \text{ALOC}\\ \text{SHAR}\end{matrix}\right\}$ class [MALFUNC] [NONPRES]

where:

class	E for memory below 128KW H for high speed memory above 128KW S for slow speed memory above 128KW
low	is the low physical memory address boundary
high	is the high physical memory address boundary
FREE	free memory
ALOC	allocated memory
SHAR	shared memory
MALFUNC	(malfunctioning memory). Field is blank if memory is operational.
NONPRES	non-present memory. Field is blank if specified memory is configured.

In response to STATUS CHA, address:

chaddr UNITS units I/O queue

DEV devaddr ON/OFF {FREE / ALOC / SHAR} [taskno] dest

(device line is repeated for each device on the channel)

where:

chaddr	is the channel address or GPMC device address and the controller type of this controller.
units	is the number of units on this controller.
queue	is the number of I/O requests queued for this controller.
devaddr	is the device address and the device type.
OFF	device is off-line.
ON	device is on-line.
FREE	device is free.
ALOC	device is allocated.
SHAR	device is shared.
taskno	is the task number of the task to which the device is allocated. This field is blank if the device is not allocated.
dest	this field indicates the device's availability for automatic selection as a destination device for SLO and SBO files as follows:

RT SLO (real-time SLO files)

RT SBO (real-time SBO files)

BT SLO (batch SLO files)

BT SBO (batch SBO files)

In response to STATUS DEVICE,devmnc, the same information displayed for each device on a channel is displayed for the specified device. See STATUS CHA.

In response to STATUS T,taskname or STATUS taskno:

taskno taskname ownername pseudonym priority type origin MEM=class MAPBLKS=mapblocks priv res state abcdefgh DEV=dev NWIO=requests RRCT=requests MRCT=requests GQID = enque information

where:

taskno	is the task number assigned at activation.
taskname	is the load module name of the task.
ownername	is the owner of the task.
pseudonym	is the pseudonym of the task.
priority	is the current priority level from 1 to 64.
type	RT (real-time, priority 1-54) TD (time-distribution, priority 55-64)
origin	B (batch) T (terminal Ø (independent)
class	E for memory below 128KW H for high speed memory above 128KW S for slow speed memory above 128KW
mapblocks	number of 8K or 2K map blocks used by the task.
priv	U (unprivileged) P (privileged)
res	N (nonresident) R (resident)
state	is a four-character identifier corresponding to the state of the task described below: Current Task in Execution (CURR) Real Time Priority - Ready to Run (SQRT) Time Distribution Levels - Ready to Run (SQ54 - SQ64) Current IPU Task in Execution (CIPU) Requesting IPU Task (RIPU)

Wait Interactive (SWTI)

Waiting for Wait I/O (SWIO)

Wait, Sending Message (SWSM)

Wait, Sending Run Request (SWSR)

Wait, Low Speed Output (SWLO)

Wait, Suspended for Message Interrupt, Timeout, or Resume (SUSP)

Wait for Wait Run Request or Timeout (RUNW)

Wait, Operator Hold (HOLD)

Wait for any No-wait I/O, No-wait Run Request, or any Message Interrupt or Break (ANYW)

Wait, Disc Space (SWDC)

Wait, Peripheral (SWDV)

Wait, FISE (SWFI)

Wait, Memory (MRQ)

Wait, Memory Pool (SWMP)

Preactivation Phase - See Section 2.1.1 (PREA)

Dispatach queue available for allocation by a task (FREE)

General wait queue - resourecmark (SWGQ)

a
0 (in memory)
1 (outswapped)

b
0 (swappable)
1 (unswappable)

c
0 (no abort requested)
1 (abort requested)

d
0 (no message receiver)
1 (has message receiver)

e
0 (no break receiver)
1 (has break receiver)

f
0 (not indirectly connected)
1 (indirectly connected)

g	0 (multicopied or shared load module) 1 (unique copy load module)
h	reserved
DEV	is only displayed if state = SWDV or SWDC
NWIO	number of no-wait I/O requests outstanding
RRCT	number of run receiver requests outstanding
MRCT	number of message receiver requests outstanding
GQID	is displayed only if the state = SWGQ. Three words of information containing the ENQUE ID are DQE.PRS, DQE.PRM and the function code DQE.GQFN.

In response to STATUS IPU:

{ON / OFF} TASK=taskname OWNER=ownername TASK#=taskno

where:

ON	indicates the IPU is online
OFF	indicates the IPU is offline
taskname	is the task name of the task running in the IPU
ownername	is the owner name of the task running in the IPU
taskno	is the task number of the task running in the IPU

If a task is not running in the IPU, the following message will be displayed:

{ON / OFF} THERE IS NO TASK RUNNING IN THE IPU

If an IPU is not on the system, the following message will be displayed:

THERE IS NO IPU ON THIS SYSTEM

Examples:

??STATUS MEMORY

Comment:

Displays current memory utilization map for each class of memory (e.g., E, H, S)

??STATUS MEMORY,E

Comment:

Displays current memory utilization map for memory class E.

??STATUS CHA,7A

Comment:

Displays the status of channel 7A and all devices connected to the channel.

??STATUS DEVICE,LP7A01

Comment:

Displays status of the line printer device on channel 7A with subaddress 01.

??STATUS T,PGMTEST

Comment:

Displays the current status of all tasks named PGMTEST.

??STATUS 02000001

Comment:

Displays the status of the task with task number 02000001.

4.4.38 SYSASSIGN Command

The SYSASSIGN command is used to establish the availability of a device for automatic selection as the final destination device for printed (SLO) and punched (SBO) output. It can also be used to change the default system input device (SID) for batch.

Syntax:

$$\underline{\text{SYSA}}\text{SSIGN}\left[\begin{array}{l}\left\{\begin{array}{l}\text{ON}\\\text{OFF}\end{array}\right\},\text{devmnc}\quad,\quad\left\{\begin{array}{l}\text{L}\\\text{P}\end{array}\right\}\quad,\left\{\begin{array}{l}\text{R}\\\text{B}\end{array}\right\}\\\text{SID, devmnc}\left[,\text{density,parity}\right]\end{array}\right]$$

where:

ON	makes the device available for automatic selection.
OFF	makes a device unavailable for automatic selection.
devmnc	is a device mnemonic, e.g., MT0100. (For description, see Appendix A.)
L	device is destination for listed output (SLO files).
P	device is destination for punched output (SBO files).
R	use to select device for real-time output.
B	use to select device for batch output.
density	is "H" (High density) or "L" (Low density) if the specified device is 7-track magnetic tape; otherwise omit this parameter.
parity	is "E" (Even parity) or "O" (Odd parity) if the specified device is 7-track magnetic tape; otherwise omit this parameter.

Response:

LF (line feed)

Examples:

??SYSASSIGN ON,LP7A,L,R

Comment:

Makes the line printer on channel 7A with subaddress 00 available for automatic selection for spooled real-time SLO output.

??SYSASSIGN OFF,CP7401,P,B

Comment:

Makes the card punch on channel 74 with subaddress 01 unavailable for automatic selection for spooled batch SBO output.

??SYSASSIGN SID,CR7801

Comment:

Changes the default System Input Device (SID) to the card reader on channel 78 with subaddress 01.

4.4.39 TIME Command

The TIME command is used to print the date and time-of-day on the console.

Syntax:

TIME

Response:

month/day/year hour:minute:second

where:

month	is the two-digit decimal month
day	is the two-digit decimal day
year	is the two-digit decimal year
hour	is the two-digit decimal hour
minute	is the two-digit decimal minute
second	is the two-digit decimal second

LF (line feed)

Example:

```
??TIME
01/29/78 11:59:35
```

4.4.40 TURNON Command

The TURNON command is used to activate or resume a specified task at a specified time and reactivate it at specified intervals by creating a timer table entry using a specified timer ID.

Syntax:

TURNON loadmod,time,[reset],timerid

where:

loadmod	is the 1-8 character name of the permanent load module file where the task is cataloged. It must be a system file (no user name).
time	is the time of day the task will be activated or resumed, entered in the format hour:minute:second where hour, minute, and second are the 2-digit decimal hour, minute, and second on the 24 hour clock.
reset	is the time interval to elapse before resetting the clock upon each timeout, entered in the format hour:minute:second where hour, minute, and second are the 2-digit decimal hour, minute, and second on the 24 hour clock. The task will be reactivated at each timeout. If a reset value is not specified, the comma denoting the field must still be specified, and the task will be activated or resumed only once.
timerid	is the 2-character ASCII name of the timer that will be created.

4.4.41 URGENT Command

The URGENT command is used to change the priority of a batch job. The job must be spooled to disc, waiting, or active.

Syntax:

URGENT jobno,priority

where:

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

priority — is the software priority at which the job is to run in the range 1 through 64.

Response:

LF (line feed)

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

Example:

??URGENT 700,60

Comment:

Changes the current priority of job number 700 to priority 60.

5. INTERACTIVE PROCESSING

The MPX-32 Terminal Services Manager (TSM) provides interactive, timeshared access to the MPX-32 system for terminals connected either through TLC, ADS, or ALIM controllers. It is an integral part of the MPX-32 operating system and allows the terminal user to:

log on to MPX-32

access any MPX-32 processor

run interactive or noninteractive (i.e., batch or real time) tasks in the interactive environment

access other environments in the system, e.g., to activate a task at its base (normally real-time) priority in the independent environment or submit a job to run in the batch environment

return to the interactive environment on exit from another processor

initiate processing from a command file

log off MPX-32

This section describes TSM functions and TSM commands. Many TSM commands are used for 'job control', i.e., to accomplish the same f#nctions as job statements used when submitting a job for batch processing. Other TSM commands allow the user to activate and run tasks online and send messages to users at other terminals.

Two TSM command verbs are optional: RUN (or EXECUTE) and SELECT. RUN is used to initiate a task and SELECT is used to access a command file. When no command verb is supplied, TSM checks first for a task load module file. If it finds none, it checks for a user command file. The verbs RUN (or equivalent) and SELECT can be used to avoid ambiguity about the type of file to access.

Terminals must be initialized before they can be used with the MPX-32 system. If a terminal hardware interface has been disrupted, e.g., the terminal has been unplugged, it must be reinitialized. See Section 5.7.

5.1 User Interaction

5.1.1 Logging On

To log on to MPX-32 from a terminal, depress the wakeup character defined for terminals at SYSGEN, e.g., CTRL E, ?, etc. TSM responds:

```
GOULD S.E.L.  MPX-32  1.5X  TSM  *systemname*

ENTER OWNERNAME AND KEY: ownername [key]
```

Respond by entering a one-to-eight character owner name and optional key. If an M.KEY file has been established (via the KEY utility), it contains valid owner names and can contain owner keys (to restrict access to an owner name). TSM checks the owner name and key, if any, with M.KEY. (If M.KEY does not exist, any one-to-eight character owner name or key is valid.)

If the owner name and/or key is not valid, or if another user has logged on with the owner name you supply, TSM displays an appropriate message:

UNAUTHORIZED NAME OR KEY

or

NAME IN USE

You are prompted again to enter an owner name and key. The prompt is reissued until you enter a valid owner name and key or use just a carriage return (CR) in response to the prompt. Using just CR logs you off the terminal.

If the owner name (and key) are valid and are not currently in use at another terminal, one of two prompts are issued.

(1) If project accounting is not present on your system, you are logged on and the TSM> prompt is displayed, and you can enter any TSM command.

(2) If project accounting is present on your system, i.e., a M.PRJCT file exists and your owner name does not have a default project name/number established in the KEY utility, you are prompted to enter a valid 1-8 character alphanumeric project name/number:

ENTER PROJECT NAME/NO.:

If you enter an invalid project name/number, TSM responds:

UNAUTHORIZED PROJECT NAME/NO.
ENTER PROJECT NAME/NO.:

If you enter a carriage return, the TSM > prompt is displayed but actual access to TSM has not been gained. Capabilities available to you at this point are limited to those tasks which do not require activation. If you attempt to perform an activation function, i.e., RUN, DEBUG, INIT, you are returned to the ENTER PROJECT NAME/NO.: prompt.

In short, when project accounting is present on your system, the only way to gain full access to TSM is by entering a valid project name/number when one is requested.

The M.CNTRL file is a TSM command file which is selected by J.TSM automatically when a user logs on (the user files are searched before the system files to locate the M.CNTRL file).

Since the M.CNTRL file may contain TSM commands as well as comments, it provides a convenient way to establish defaults, send messages, and to further restrict access to the TSM environment.

Examples of usage:

```
EDT>COL
1. NOTE LOG ON AT
2. !TIME
3. <cr>
EDT>STO M.CNTRL SYS
```

This will cause the time and date to be automatically displayed each time a user logs on the system. The word NOTE indicates the line is a comment line.

```
EDT>COL
1. , Msg
2. <cr>
EDT>STO M.CNTRL
```

This will cause a message to be displayed to a user who signs on under the username M.CNTRL is stored on.

5.1.2 Accessing Batch, Independent, or Interactive Processing Environments

TSM provides the terminal user with access to batch and real time environments as well as to the interactive environment. As illustrated in Figure 5-1, the batch or real time environments are accessed by using the [RUN] OPCOM command in response to the TSM > prompt. TSM also recognizes an exclamation point as a synonym for 'OPCOM', so that the user can issue an OPCOM command by typing '!command'. This allows a one-shot access to OPCOM with automatic return to TSM.

A user task can be put into execution in any of the three processing environments after logging on. To activate a task from the interactive environment at its cataloged priority (e.g., to activate a task for independent processing in the real time environment), the task must be cataloged with all, except dynamic, assignments. The interactive user can enter OPCOM and ACTIVATE or ESTABLISH the task. For example:

```
TSM > OPCOM        (or)        TSM > !ACTIVATE MYTASK
?? ACTIVATE MYTASK             TSM>
?? EXIT
TSM>
```

The user can activate the same task in the interactive environment by supplying just the task name in response to the TSM prompt, e.g.:

```
TSM > MYTASK
```

In the interactive environment, the user can supply assignments. The task executes at the time distribution priority of TSM and has special TSM support for breaks, wakeup, SYC assignments, and other functions as described in Section 5.2

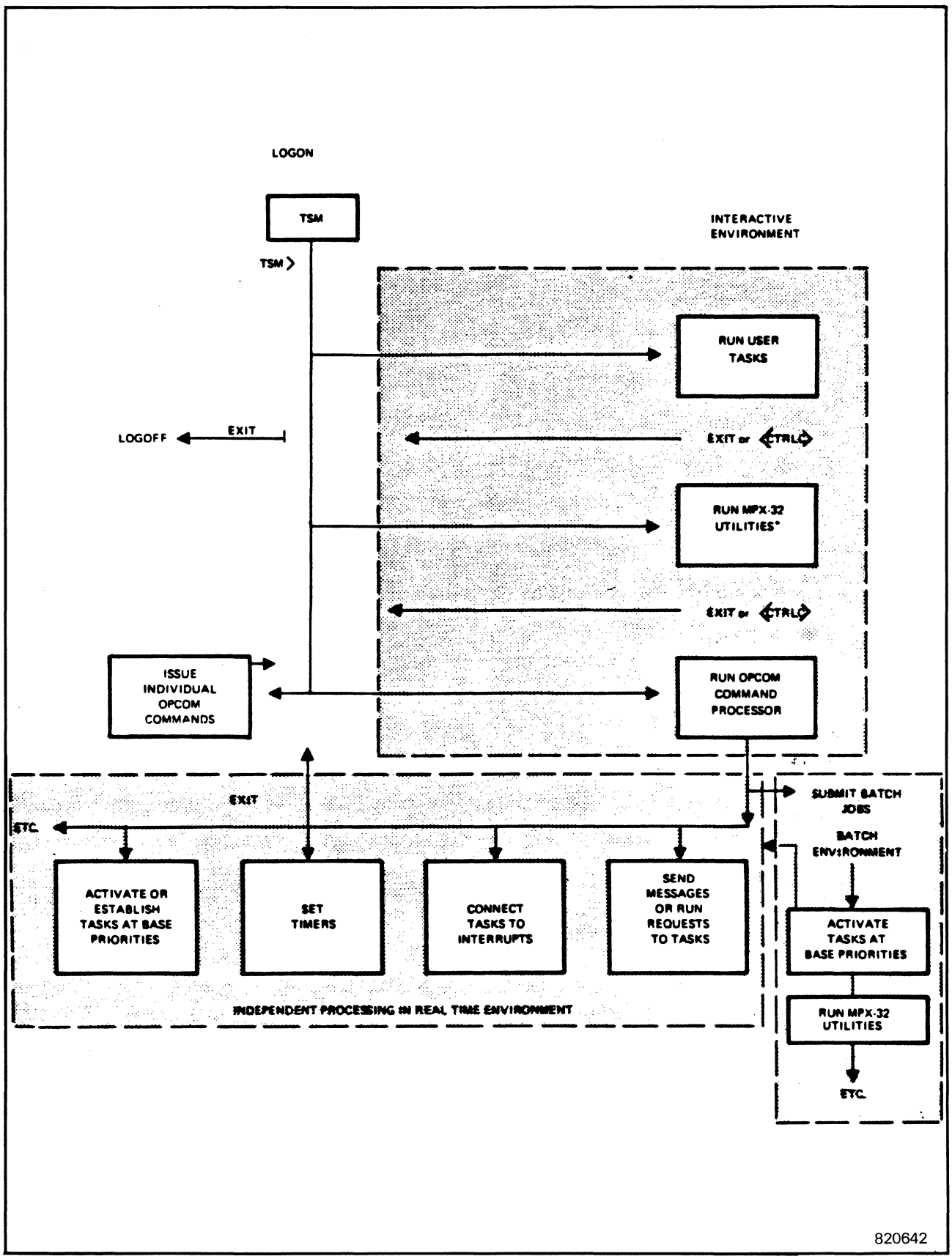


Figure 5-1. Interactive/Batch/Real Time Environments

Or the user can activate the task as part of a batch job (in the batch environment) by creating a job file and submitting it with an OPCOM or Text Editor BATCH command, e.g.,

```
TSM > OPCOM      (or)   TSM > !BATCH jobfile

?? BATCH jobfile        TSM >

?? EXIT

TSM >
```

or

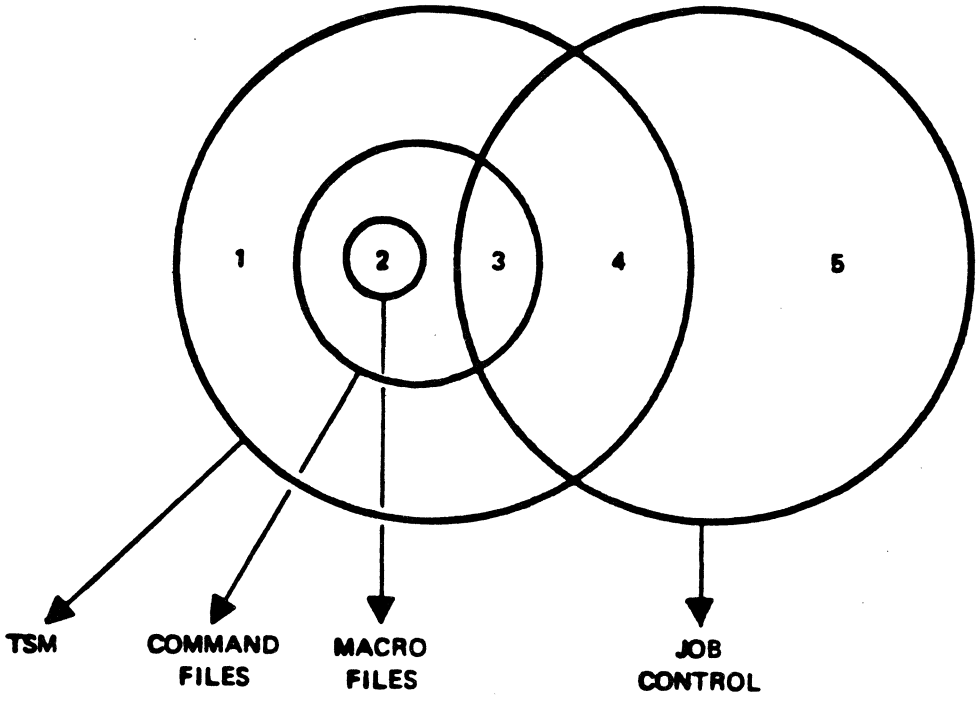
```
TSM > EDIT

EDT > BATCH jobfile

EDT > EXIT

TSM >
```

Figure 5-2 shows TSM, Command Files, command files with parameter passing (Macro Files) and Job Control commands, and their valid/non-valid usages are described in relation to each other.



1 Commands valid throughout TSM but not valid in Job Control.
ACCOUNT, CLEAR, CPUTIME, EXIT, INIT, RESTART, RUN, SCAN, SELECT, SIGNAL, WAIT, WHO

2 Commands valid only in Macro Files. Never valid in Job Control.
DEFM, ENDM, IFA, IFP, IFF%, IFT%, %

3 Commands valid in Job Control but not valid in TSM unless in a file.
DEFNAME, GOTO, IFF flag, IFT flag

4 Commands valid throughout TSM and Job Control.
ALLOCATE, ASSIGN1, ASSIGN2, ASSIGN3, ASSIGN4, DEBUG, EOJ, EXECUTE, JOB, NOTE, OPTION, RESETF, SETF, USERNAME

5 Commands valid only in Job Control.
ACTIVATE, OBJECT, SELECTD, SELECTF, SELECTLD, SELECTLF, SELECTS

820639

Figure 5-2. TSM/Job Control Commands

A task can be activated at its base priority in the real time environment via the batch statement $ACTIVATE. In batch, assignment and option statements can precede $ACTIVATE so that the task need not be cataloged with all non-dynamic resource assignments.

When a batch job is submitted from the terminal, some messages pertaining to the job are displayed at the submitting terminal as well as the OPCOM console. They are indicated in Section 6.8 of this volume. An end of job message is displayed at the submitting terminal when the job is complete:

jobno owner $EOJ jobname
JOB EXECUTION TIME = xx HOURS- xx MINUTES- xx.xx SEC

5.1.3 Returning to TSM

The terminal user must eventually return to TSM via an EXIT command or Control C <CTRLC > key sequence in order to log off the system or access a different processor.

5.1.4 Logging Off

To log off a terminal, use the TSM EXIT command in response to the TSM> prompt. The following message will then be displayed:

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

RING IN FOR SERVICE

where xx is a decimal number.

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

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

5.1.5 Special Keys

Table 5-1 indicates special keys used to correct characters, correct a line of terminal input, and perform other special interactive functions.

Key	Function
RETURN	Terminates the current input or command line. Also used in other places, for example, to continue output after a pause at bottom of screen. Typing RETURN is assumed in all cases at the end of an input line or command line and is not discussed in the documentation. Referred to in text or examples as < CR> or carriage return.
<CTRLH>	Deletes the character just typed. Can be used repeatedly. Enter the correct character(s).
RUB or DELETE	Deletes the line just typed. Must be used before a carriage return. The cursor moves to the next line. Enter the corrected line.
<CTRLI>	Used to display tab spacing on input. If terminal has TAB key, produces same result. Shown in text and examples as <CTRLI>.
<CTRLC>	Used to exit a processor when an EXIT command is not available. Returns you to the TSM > prompt. Shown in text and examples as <CTRLC>. Simulates EOF function from terminals.
BREAK	See Section 5.2.5.
Wakeup	See Section 5.2.4.

Table 5-1 Special Keys

5.1.6 Communicating with Other Terminals

The TSM SIGNAL command can be used to send a message (72-characters maximum) to a specific terminal user or to all terminals. When a message is entered, it is stamped with the owner name of the sender and the date and time sent. If a message is sent to an owner who is not currently logged on, the command is denied. If a message is sent to all terminals, it is queued for output to any terminal not currently logged on and displayed the next time a user logs on to that terminal.

To communicate with a particular task, see the OPCOM SEND command. SEND allows tasks with message or run receivers to accept control messages or data from the OPCOM task running on a terminal, i.e., from the user who has accessed OPCOM.

5.1.7 File Access

The owner name you supply at logon is always taken as a user name for files accessed interactively unless overridden with the TSM USERNAME command. The TSM USERNAME command can be used to access files belonging to a different user. Sample use: before assigning an input file that has a different user name for a task to be run interactively.

The user name you establish with the TSM USERNAME command remains in effect for interactive processing until you use USERNAME again or log off the system. When you log on again, the original user name = user name association is established.

User names are associated with files in the System Master Directory (SMD). When a file name is specified, the system always looks first in the SMD for files with the current user name. If a file with the specified name does not exist with the current user name, the system searches for a system file (no user name).

An equivalent USERNAME command is available for batch jobs and for various utilities. When used in the batch environment or with a utility, the modified user name applies only to the batch or utility being run. When you return to TSM, the user name supplied with the most recent TSM USERNAME command is returned.

5.1.8 A Typical Terminal Session

The following example demonstrates the interaction between the user and the system during a typical session.

```
?                                   User depresses special wake-up character, '?' in this case.

SYSTEMS MPX TSM *BIGFOOT*
ENTER YOUR OWNERNAME AND KEY:BABBAGE OK

    ENTER PROJECT NAME/NO.: S

TSM>ASSIGN4 5=UT                    Lfc 5 is this terminal.

TSM>A4 6=UT                         Lfc 6 is also this terminal.

TSM>OPTION PROMPT                   Generate prompt for input.

TSM>EXECUTE ANYTASK                 Run the task.

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

ANY>LIST                            Prompt is issued before read.

(task lists 24 lines)

ENTER CR FOR MORE <CR>              Bottom of screen is reached; user enters CR.

(program resumes listing)

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

BREAK ON ANYTASK. AT 2003BC44. ET: 1.02 SEC.

CONT, ABORT, OR DEBUG? A

ANYTASK,02000001 ABORTED. PSW:2003BC44 BIAS:34000 REASON:TS01

TSM > CLEAR

TSM > OPCOM

?? ACTIVATE REALTIME

?? EXIT

TSM>EXIT

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

5.2 Executing Tasks Under TSM

Any cataloged load module can be run interactively by issuing the TSM RUN command. Tasks which are designed to run in any environment (online, independent-realtime, or batch) can be run from terminals.

Files and devices can be assigned and options selected for a task to be run interactively as described in Sections 5.2.1 and 5.2.2 next. Assignments and options apply only to one task. After the task is run, they are cleared automatically by TSM.

The CLEAR command can be used to clear assignments and options before a task is run. CLEAR also performs the function of terminating input from a command file.

5.2.1 Assignments

TSM overrides the cataloged assignment of a logical file code to SYC, if any, with an assignment to the user's terminal (ASSIGN3 UT = terminal number, ASSIGN2 SYC = UT). This provides a task designed to run from a control file (SYC) in batch with automatic reads from the terminal or command file, for control commands and directives. The user can associate any logical file code he wants to UT for interactive operation by cataloging a task with ASSIGN4 lfc=UT, or by providing the assignment with an ASSIGN4 command at the TSM > prompt.

If a task is activated with TSM that requires no input from SYC and has no assignments to UT, the task simply runs with no prompt for terminal interaction until it exits or aborts. The user can communicate with the task via the Break key or the wakeup character. A task can also be designed to run interactively. Special TSM features available to the programmer are described in Section 5.6.

When a task is run interactively, regardless of its internal structure, the terminal is automatically associated with the task by TSM, (UT = terminal number).

5.2.2 Options

5.2.2.1 Prompting Option

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

5.2.2.2 Lower Case Option

When running a task (e.g., the EDITOR utility) online, the task may force a translation of lower case characters to upper case. This translation may be inhibited via the TSM OPTION LOWER command to allow entering lower case characters. The only strings that must be entered in all upper case are file names.

5.2.2.3 Internal Processing Unit (IPU) Options

If an Internal Processing Unit (IPU) is installed on your system, two options are available for its use:

IPUBIAS (Bit 26) - When set, tasks that are IPU compatible will be run by the IPU processor. At any point during execution where compatibility ceases, the CPU is trapped and the task is transferred to the CPU processor for execution.

CPUONLY (Bit 25) - When set, the IPU processor is ignored and all tasks are executed by the CPU processor.

Default: Tasks are executed by the first available processor.

5.2.2.4 Error/Noerror Option

When a TSM task aborts, the abort code is automatically interpreted and an explanation is displayed at the user's terminal. To inhibit the automatic display, specify OPTION NOERROR. The automatic display can be reset by specifying OPTION ERROR.

5.2.2.5 Command/Nocommand Option

When TSM reads commands from a macro or command file, they are echoed at the user's terminal prefixed by an asterisk. To inhibit this operation, specify OPTION NOCOMMAND. This option remains in effect until an end-of-file (EOF) or end macro (ENDM) is encountered, at which time the default of listing the commands is reset. The listing operation can be reset at any time by specifying OPTION COMMAND at the TSM > prompt.

5.2.2.6 Text/Notext Option

When a processor reads commands from a macro or command file, they can be echoed at the user's terminal (prefixed by an asterisk) by specifying OPTION TEXT. This option remains in effect until an end-of-file (EOF) or end macro (ENDM) is encountered, at which time the default of not listing the commands is reset. The listing operation can be inhibited at any time by specifying OPTION NOTEXT at the TSM > prompt.

5.2.3 Breaks

If you depress the Break key and a task does not have its own break receiver, TSM prompts:

```
** BREAK ** ON:taskname  AT:location  ET:n SEC.

CONT,ABORT,OR DEBUG?
```

If you respond C (Continue), the task continues in execution. If you respond A (Abort), TSM aborts the task, displays the message below, and returns the TSM> prompt.

```
taskname,taskno ABORTED.  PSW:psw-contents  BIAS:bias

REASON:  TS01

TSM>
```

TS01 indicates user requested (from the terminal) to be aborted from a BREAK request.

If you respond D (DEBUG), the MPX-32 Debugger is attached to the task and you are prompted for a DEBUG command (See the DEBUG description in Volume 2.)

If the task has its own break receiver, TSM defers to it when you depress the Break key.

5.2.4 Wakeup's

If a task is running and does not respond to the Break key, it is in a wait-for-resources state. To communicate with a task that is waiting for resources (memory, a device, a file, etc.) you can enter the wakeup character used to logon. TSM responds:

```
<wakeup>

TASK state.  CONTINUE OR DELETE?
```

TSM displays the type of resource for which the task is queued. To delete the task, respond D (Delete). The task is killed as if you had entered a KILL command in OPCOM, and TSM displays an abort message on the terminal:

```
taskname,taskno ABORTED.  PSW:psw-contents  BIAS:bias

REASON:  TS02

TSM>
```

TS02 indicates user requested (from terminal) a task to be deleted from a wait state queue.

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

The wakeup character is also used to reestablish dialogue with TSM after entering a special wait (for messages) state. (See the TSM WAIT command.)

5.2.5 TSM Screen Logic

TSM bottom end-of-screen logic is automatically available for a task run interactively. TSM uses the predefined screen length for a terminal. At the bottom of the screen, on a write, TSM prompts the terminal user:

ENTER CR FOR MORE:

If the user responds with a CR, TSM writes the next line on the screen and returns to the task. If the user responds with other than a carriage return, TSM throws away the line that caused it to go to the end of screen. It sets bit 7, Word 3 in the FCB that indicates end of medium before returning to the task. If the task has the logic to check end of medium, the task will be able to process one of two ways according to the situation. If the task does not have the logic to check and, for example, stop output, TSM continues output to the next bottom of screen.

When programming in a high level language, the programmer must consider testing for end of medium before each write to a TSM device. If no testing is used, the language will handle a write while at end of medium based on the language being used.

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

5.2.6 Tabs

The M.KEY file in MPX-32 can contain tab settings for each ownername specified in the logon file as described in the KEY processor in Volume 2. TSM defaults to these tabs if they exist. Or, if no tabs are set in M.KEY, TSM sets system tabs when you access the EDITOR SET TABS command in Volume 2. You can override the tab settings (M.KEY/system) when using the EDITOR SET TABS command.

The most recent tabs set with SET TABS will remain in effect as long as you remain on the system (until you exit TSM), i.e., they will remain in effect as you access various processors, move from one system environment to another, etc. When you exit from TSM, the SET TABS are not saved. When you log on again, either the M.KEY tabs (or system tabs if none exist in M.KEY) are set.

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

The tab character used to define tab positions in the SET TABS command is either a backslash (\) or the character you define when you enter tab settings. The most recent tab character defined in SET TABS is the one used when entering a tabbed record unless you want to use <CTRL I>or TAB. Like tab settings, the tab character remains in effect until you exit TSM. When you log on again, the tab character is a backslash.

5.2.7 Project Names/Numbers

When using the ACCOUNT utility under MPX-32, project names/numbers have to be established for owner names on the system to gain access to TSM. This can be done in one of two ways.

(1) A M.PRJCT file can be created via the Text Editor containing the 1-8 character (per entry) alphanumeric project names/numbers which will be valid on the system in use. This file can then be stored unnumbered.

(2) Default project names/numbers can be established for each owner name via the KEY utility.

In addition, a file must be created (it can be created by the File Manager CREATE directive) and named M.ACCNT (file size is determined by individual needs). This file is where job accounting information is collected for use by TSM.

When M.ACCNT is approximately 90% full, TSM displays the following message on the console:

WARNING THE ACCOUNTING FILE IS OVER 90% FULL.

M.ACCNT should be listed to the printer (see Section 4.4.19) or saved to magnetic tape (see Section 4.4.31) and then purged (see Section 4.4.24). Until M.ACCNT is purged, the warning message is displayed each time MPX-32 is restarted. When M.ACCNT becomes completely full, it is automatically purged and all previous accounting data is lost.

Project names/numbers can be changed by a user while logged on via the PROJECT command under TSM. When you change a project name/number, you terminate the ACCOUNT utility for the previous project name/number and initiate the ACCOUNT utility for the new project name/number which remains in effect until changed again or you log off the system.

Upon logging off the system, default project names/numbers are restored, if any.

5.3 Using Command Files

MPX-32 accepts a file of TSM commands in lieu of terminal input, allowing the user to:

make assignments, define options, and accomplish other runtime operations associated with activating a task or processor

activate and run tasks or processors in the interactive environment

perform conditional processing

select other command files

perform argument replacements in any line of code regardless of whether used at the TSM > prompt or used when in a particular processor.

5.3.1 Activating Tasks from Command Files

The user can activate any task from a command file. Any parameters passed with the RUN command can be accessed by an interactive task via the M.TSCAN service. Records following the RUN command line within the command file are accessible to tasks; however, any command preceded by a dollar sign ($) is interpreted as a command to TSM and a pseudo end of file for the currently executing task.

If a command file contains more commands following the pseudo end of file, there are two different paths that can be taken, depending on whether the currently executing task interprets pseudo end of file as an exit or not.

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

If the task does not exit at pseudo end of file, the task can read from the terminal. Reading from the terminal continues until the task does exit. At that time, TSM processes the $command from the command file.

5.3.2 Chaining Command Files

SELECT commands can be embedded in command files to access other command files. Without conditional processing, the chaining capability is directly from one command file to the next, i.e., the first SELECT encountered accesses a new command file. However, with conditional processing, the user can select alternate command files to gain more flexibility in command file processing.

5.3.3 Command File-to-Terminal Interplay

TSM provides two file codes for interactive processing: SYC (system control) and UT (user's terminal). When processing a command file, the command file is assigned by TSM to SYC. With no command file, SYC = UT. UT is always assigned to the user's terminal.

As TSM reads a command file, it displays commands and responses at the terminal (to UT) as if it were operating interactively. When it reads an EXECUTE or equivalent command to activate a task, TSM turns control over to the task. Whether commands and responses to the task are displayed or not depends on whether the task uses UT for I/O.

When all commands on the command file have been processed, the user enters a CLEAR command, TSM encounters a DEFM command, or a task reads from the terminal, TSM returns to the terminal for input.

5.3.4 Error Processing

Errors that may be encountered by TSM in command file processing include:

1. An entry in the command file that is not a TSM command, a load module (task) name, or a command file

2. Supplying invalid parameters for a TSM command, a task, or a command file.

3. Interactive errors which are encountered by a task or processor activated via a command file.

In the first case, command file processing terminates and the control returns to the terminal. In the second case, the user can correct the command line that caused the error and continue processing from the command file. Or the user can abort command file processing by issuing a CLEAR command.

If an error is encountered by a task or processor that has been activated from a command file, the error and recovery are the same as in normal interactive processing. The user can, of course, issue a break at any time and follow a continue, abort, or debug path as described in Section 5.2.3. When a task is aborted, control returns to TSM. TSM continues processing commands from the command file at the point following the abort. IFT and IFF commands can be used for conditional processing on abort. (See command descriptions.) Error messages are descriptive and most will not need further reference. However, if a file allocation error code is given, refer to Section 7.8.1 of this volume for a description of the error.

5.3.5 Conditional Processing and Parameter Passing

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

Substitute parameters. When the file is accessed, the user can pass up to eight predefined variables to it, adapting the one command file to various runtime uses.

Provide parameter defaults. When the runtime user passes parameters, he can obtain a default value by entering nothing or if a gap, by using an extra comma or other valid non-blank TSM delimiter, to indicate a missing parameter.

Use conditional execution capabilities; e.g., based on a value actually supplied for a particular condition, the command file can branch to a particular set of commands or select a different command file altogether.

Please note that using task names identical to command file names should be avoided. The task names will take precedence over the command file unless the SELECT command is used.

5.3.6 Concatenating a Value to a User-Supplied Parameter

In a command file that uses parameter substitution, the command file can use two optional forms of a parameter name:

value%par
or
%par;value

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

5.3.7 Breaks and Wakeups

If the break key is depressed or the wakeup character used to log on is entered, the next command in the command file will not be processed. The following message will be displayed if either event occurs:

A BREAK OCCURRED DURING COMMAND FILE PROCESSING.
ENTER CR TO CONTINUE OR 'CLEAR' TO TERMINATE.

If a carriage return is entered, TSM reads the next command in the command file.

If the CLEAR command is entered, TSM terminates command file processing and returns the TSM prompt.

5.4 TSM Commands

TSM commands are summarized in the chart which follows and described in detail in the following pages. Valid abbreviations are shown by underlining.

Any TSM command can be preceded by a dollar sign ($). This allows command files to be processed in Batch, if desired. It also facilitates utility or task processing (see Section 5.3.1).

Command	Function
ACCOUNT	Displays contents of job accounting file to logical file code UT.
ALLOCATE	Overrides cataloged memory allocation for tasks run online.
ASSIGN1	For tasks run online, associates permanent disc file (optionally unblocked) with lfc.
ASSIGN2	Associates system SBO, SLO, SYC or SGO file with lfc for task run online. SYC is automatically associated with the user's terminal.
ASSIGN3	For task run online, associates device (optionally unblocked) with lfc. If channel/subaddress specified, denotes specific device. User's terminal is preassigned as "UT=terminal". To assign the terminal for other lfc's, use ASSIGN4.

ASSIGN4	Associates lfc with another lfc for task run online. All lfc's referring to user's terminal should be assigned as "UT", e.g., SI=UT.
CLEAR*	Clears all previous ASSIGN, OPTION, and ALLOCATE commands.
CPUTIME*	Displays elapsed CPU execution time for all tasks activated by an owner since logon.
DEBUG	Loads the specified task with the interactive Debugger (DEBUG) attached and passes control to the Debugger.
DEFM*	Defines parameters for a command file.
DEFNAME	Establishes a name to branch to for conditional processing.
ENDM*	Defines end of command file. Optional.
EOJ	Deletes an SGO file established via the JOB command.
ERR?*	Defines any specified error code.
EXECUTE or RUN	Activates a task in online environment. The task name specified must be a cataloged load module.
EXIT* or X*	Ends online activity. Logs user off terminal.
GOTO	Branches to a name in a command file.
IFA*	Branches to a name if the specified parameter is absent.
IFF	Branches to a name if a condition is false.
IFP*	Branches to a name if the specified parameter is present.
IFT	Branches to a name if a condition is true.
INIT*	Initializes terminals. See Section 5.7.
JOB	Creates an SGO file for processing.

* Not a valid command under Batch. (See Section 6.5.)

LINESIZE* Dynamically modifies the screen width defined for a terminal.

message* Sends a message to the terminal.

%name* Establishes a name to branch to for conditional processing.

NOTE Sends a message to the terminal.

OPTION Provides options for tasks running online:

1-20: task-dependent options.

21-32: system-defined options available to all tasks.

DUMP (Option 24): dumps task's area of memory to SLO on abort.

PROMPT (Option 21): uses first three characters of task initiated with RUN plus as prompting character for task read requests through lfc UT. The prompt is preceded by a carriage return/linefeed.

LOWER (Option 22): suppresses automatic conversion of lower case characters to upper case on input; good for data, but user is cautioned that commands and key words must be shifted to upper case.

IPUBIAS (Option 26): IPU compatible tasks will be executed by the IPU processor.

CPUONLY (Option 25): tasks are executed only by the CPU processor. Default: tasks are executed by the first available processor.

COMMAND: echoes commands in a macro or command file to the user's terminal as they are read by TSM.

NOCOMMAND: inhibits commands in a macro or command file from being echoed to the user's terminal as they are read by TSM.

ERROR: causes automatic display of abort description at the user's terminal when an abort occurs.

NOERROR: inhibits display of abort description at the user's terminal when an abort occurs.

TEXT: echoes commands in a macro or command file to the user's terminal as they are read by a processor.

NOTEXT: inhibits commands in a macro or command file from being echoed to the user's terminal as they are read by a processor.

* Not a valid command under Batch. (See Section 6.5)

PAGESIZE*	Dynamically modifies the screen length defined for a terminal.
PROJECT*	Changes the project name/number for accounting purposes.
RESTART*	Reboots the resident MPX-32 operating system. See Volume 3.
RESETF	Sets flag(s) to FALSE for conditional processing.
SCAN**	Dynamically modifies the screen width defined for a terminal.
SELECT**	Selects a command file of TSM commands.
SETF	Sets flags(s) to TRUE for conditional processing.
SIGNAL*	Sends a message to another logged on user or to all terminals.
USERNAME	Modifies current user name for file access.
WAIT*	Waits for messages when terminal would otherwise be inactive. Requires wakeup to resume interactive processing.
WHO*	Displays owner names and terminal addresses of all logged on users.
$$	Ignored by TSM.
$$$	Ignored by TSM.

* Not a valid command under Batch. (See Section 6.5.)
** Functionality different under Batch. (See Section 6.5.)

5.4.1 ACCOUNT Command

The ACCOUNT command is used to display the entries in the job accounting file to logical file code UT.

Syntax:

ACCOUNT [[,OWNE=name] [,PROJ=proj] [,DATE=date] [ORIG= {TSM / BATCH}]]

where:

	If no parameters are specified, the entire contents of the job accounting file are displayed.
OWNE=name	specifies the 1-8 character owner name associated with the files to be displayed.
PROJ=proj	specifies the 1-8 character alphanumeric project name/number associated with the files to be displayed.
DATE=date	specifies the 8 character numeric date associated with the files to be displayed, entered in the format month/day/year
ORIG={TSM / BATCH}	specifies the mode of operation in which the files to be displayed were used.

Note: Keyword parameters can contain leading or trailing wild card characters in the form of question marks (?).

Examples:

```
ACCO,OWNE=JIM,DATE=06/16/80,ORIG=TSM.20??
```

Displays statistics on all jobs with owner name JIM run on June 16, 1980 on any TSM device 20xx.

```
ACCO,PROJ=087212
```

Displays statistics on all jobs run under project number 087212.

5.4.2 ALLOCATE Command

A task is always allocated enough memory to accommodate a cataloged load module. ALLOCATE is used to increase the memory allocation for a task at execution time.

The ALLOCATE command gets additional memory when the task is run, i.e., it is dynamic.

Syntax:

ALLOCATE bytes

where:

bytes — specifies the number of additional bytes (in hex) to allocate to the task.

5.4.3 ASSIGN1 Command

The ASSIGN1 command is used to assign permanent files for logical file codes used by the task being run.

Syntax:

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

where:

lfc	is a logical file code used in the task.
filename	is an 8-character maximum name of a disc file to assign to the lfc. Any one of the optional parameters following the file name may be entered in the order shown in the syntax statement. Commas separate options. If an option is missing, the comma must be supplied, as in: filename,,U Blanks should be used between lfc assignments. (See Examples.)
password	is an 8-character maximum password for the disc file if it has been password-protected. If RO protected, the password is required to write to the file. If PO, the password is required to read or write to the file.
U	the file is optionally unblocked. Default: blocked.

Examples:

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

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

5.4.4 ASSIGN2 Command

The ASSIGN2 command is used to supply system file assignments to logical file codes. An lfc assignment to a system file results in IOCS creating one of the types of files described below for use by the task:

SBO — System Binary Output. A type of temporary file created and used by IOCS for buffering output to the device defined at SYSGEN or via the OPCOM SYSASSIGN command as POD (Punched Output Device). Output from the user task directed to the lfc associated with SBO will be buffered and routed by IOCS to the POD.

SLO — System Listed Output. A type of temporary file created and used by IOCS for buffering output to the device defined at SYSGEN or via the OPCOM SYSASSIGN command as LOD (Listed Output Device). Output from the user task directed to the lfc associated with SLO will be buffered and routed by IOCS to the LOD.

SYC — System Control File. TSM automatically assigns SYC to UT. Tasks should use ASSIGN4 to equate any other logical file codes to UT.

SGO — System General Object. This is a temporary file used to accumulate object code.

For further description of all of the above system files, see Chapter 7.

Syntax:

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

where:

lfc — is a 3-character logical file code used in the task

SBO — System Binary Output file

cards — is the number of cards you expect to output as an object deck. Determines size of SBO temporary file required.

SLO — System Listed Output file

printlines — number of printlines required for listed output. Determines size of SLO temporary file required.

SYC — System Control File.

SGO — System General Object file. Note: SYSGEN size can be overridden with the JOB command. (See JOB.)

5.4.5 ASSIGN3 Command

The ASSIGN3 command is used to supply device assignments for logical file codes used by the task being run.

Syntax:

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

where:

lfc	is the logical file code used in the task
devmnc	is the device mnemonic of a configured perhiperal device. See Appendix A.
blocks	number of disc blocks (192 words) to be allocated for this file.
reel	specifies a 1-4 character identifier for the reel. Default: SCR (scratch)
vol	if multivolume tape, indicates volume number. Default: 0 (not multivolume)
U	specifies the tape or disc is unblocked. Default: Blocked

Note: There must be no embedded blanks within a lfc assignment. Commas must be inserted for all nonspecified options (see Examples). One or more blanks are the legal separator between one lfc assignment and the next.

Examples:

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

Disc: A3 IN=DC,20

5.4.6 ASSIGN4 Command

The ASSIGN4 command is used to associate one or more logical file codes used by the task being run with an existing lfc assignment. This assignment will remain for the associated file or device even if the original assignment is deallocated.

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

Syntax:

ASSIGN4 lfc=lfc [lfc=lfc]

where:

lfc=lfc — is a pair of logical file codes, where the first lfc is the new assignment and the second is the lfc already associated with a file or device in any previous ASSIGN directive (including ASSIGN4). When running interactively, the user's terminal is preassigned to UT as in ASSIGN3 UT=terminal.

Any number of lfc to lfc associations can be established.

5.4.7 CLEAR Command

The CLEAR command is used to clear all previous ASSIGN, OPTION, and ALLOCATE commands. CLEAR is also used to terminate processing from a command file before end of file and to return control to the terminal user.

Syntax:

CLEAR

5.4.8 CPUTIME Command

The CPUTIME command is used to return the elapsed CPU execution time for all interactive tasks activated by the owner who issues the command.

Syntax:

CPUTIME

Response:

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

5.4.9 DEBUG Command

The DEBUG command is used to debug a task (cataloged load module) using the interactive Debugger described in Volume 2.

Syntax:

DEBUG taskname

where:

taskname	is the name of a cataloged load module. Same as the name of the file containing the load module. Must be a system file.

5.4.10 DEFM Command (Command File Only)

The DEFM command supplies parameters that are processed by TSM when a command file is executed. A 'parameter' can be a substitution string for part of a value or a 'parameter' can be any value normally supplied as a variable on a given command line. Up to eight parameters can be defined for any given command file.

A single parameter must have no embedded commas, blanks, or other TSM delimiters and can be 8-characters maximum in length. The parameters typed on the command line when the file is executed must match the order of the parameters supplied on the DEFM command. If a missing value is followed by other values, the missing value must be indicated by an extra comma or other non-blank TSM delimiter.

Syntax:

DEFM [par1] [,par2] ...

where:

par1 — up to eight parameters can be defined for substitution. A parameter is eight characters maximum. If more than eight characters are supplied (in the command file or from the terminal when the command file is accessed) the extra characters are truncated.

Response:

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

Documentation of any command file containing DEFM parameters should include at a minimum, definition of the parameters to enter when the command file is accessed and the exact order in which to enter the parameters.

Example:

This command file is used to make assignments and select standard options for assembly. "SI" is the parameter indicating a base file name (e.g., if the user were to leave out the first parameter, SI would be the name of the input file for the assembly). The macro selects 1 as the default option via the second parameter.

Command File (ASSEMBLE):

```
DEFM SI,1
ASSIGN1 SI=%SI
ASSIGN1 BO=OB%SI
OPTION %1 3 4
ASSEMBLE
ENDM
```

User Supplies: TSM SELECT ASSEMBLE SRCE

Expansion of Command File as Executed:

```
ASSIGN1 SI=SRCE
ASSIGN1 BO=OBSRCE
OPTION 1 3 4
ASSEMBLE
```

Comments:

Notice that file names are generated that are unique to the user and yet consistently related by concatenation in the two ASSIGN commands. See Section 5.3.6. Use of the TSM SELECT command is illustrated because it forces the identification of the file named ASSEMBLE to be a command file and not a load module.

5.4.11 DEFNAME and %name Commands (Command File Only)

The DEFNAME and %name commands are used to establish names to branch to for conditional processing.

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

Syntax:

$$\left\{\begin{matrix}\% \\ \underline{\text{DEFNAME}}\end{matrix}\right\} \text{name}$$

where:

name	specifies a one- to eight-character name corresponding to a name referenced in a previous command. The name must contain at least one numeric character.

5.4.12 ENDM - End Macro

The ENDM command can be used to terminate a command file. It is not required.

Syntax:

```
ENDM
```

5.4.13 EOJ Command

The EOJ command is used to signal the deletion of SGO. If JOB is used, EOJ should also be used. (See the JOB command).

Syntax:

EOJ

5.4.14 ERR? Command

The ERR? command is used to display the description of any specified abort code.

Syntax:

ERR? code

where:

code is a four character abort code as defined in Appendix C

Example:

```
TSM>ERR? MS91

TASK HAS ATTEMPTED NORMAL EXIT WITH A TASK INTERRUPT STILL
ACTIVE.

TSM>
```

Errors:

If an invalid error code is requested, TSM will respond:

```
<<<UNRECOGNIZABLE ERROR CODE>>>

        (or)

<<<TOO MANY CHARACTERS IN ERROR CODE>>>
```

5.4.15 EXECUTE or RUN Command

The EXECUTE (or RUN) command is used to activate a task in the online environment. Sections 5.2 and 5.3 describe running tasks online. This command is the default command for TSM, i.e., it is implied if you do not enter a TSM command verb.

Syntax:

[RUN / EXECUTE] taskname

where:

taskname is the name of any cataloged load module file.

5.4.16 EXIT Command

The EXIT command is used to log off the system.

Syntax:

EXIT

or

X

5.4.17 GOTO Command (Command File Only)

The GOTO command is used to skip subsequent commands until the specified name is found in the command file. The name is defined via a DEFNAME or %name command.

Syntax:

GOTO name

where:

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

5.4.18 IFA and IFP Commands (Command File Only)

The IFP/IFA commands allow branching to a name based on whether a particular parameter is supplied (IFP) or not (IFA) at runtime. IFP/IFA can be used, for example, to set up a default course when a parameter is absent or they could be used to vary processing depending upon what the runtime user supplies when he selects a command file.

Syntax:

$$\left\{\begin{matrix}\text{IFP}\\\text{IFA}\end{matrix}\right\}\text{\%par name}$$

where:

%par	a parameter supplied with the DEFM command, preceded by a percent sign.
name	a name that marks a point to get to on the command file to continue processing commands. The name can be either a command or a string that marks a branch forward through the command file.

Example:

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

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

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

Expanded Commands Are:

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

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

5.4.19 IFF Command (Command File Only)

The IFF command is used to branch to a name in a command file when a condition is false. There are several types of conditions that can be tested:

- o strings - the first character string is not equal to the second character string.
- o flags - the specified flag or flags are false. Note that flags are set to TRUE or FALSE via the SETF (TRUE) or RESETF (FALSE) commands
- o ABORT - the preceding task does not abort
- o files - the specified file name does not exist

If a condition does test false, the user provides a name to branch to in the command file. If a condition does not test false, TSM continues processing with the next command in the command file.

Syntax:

IFF { %string { EQ / NE } string / flagno / ABORT / FILE filename } name

where:

string	is any string, eight characters maximum. (The string may be the result of argument substitution or concatenation.) If more than eight, characters to the right are truncated. The value can be an alphanumeric string. If the value of the expression is false, TSM branches. If it is true, TSM processes the next command on the command file. See Example.
flagno	is a numeric flag number from 1 - 16. One flag number can be specified. The flag specified must then be FALSE (see the RESETF command) to cause a branch. If the flag tests TRUE, TSM processes the next command on the command file.
ABORT	if the task immediately preceding IFF was not aborted, TSM branches. If the task was aborted, TSM processes the next command on the command file.
FILE filemname	specifies the name of a user or system file. If the file does not exist, TSM branches. If the file exists, TSM processes the next command on the command file.

name — is a one-to-eight character string defined via a DEFNAME or % name command that marks a point on a command file. When the user branches, he branches to one of the names on the command file. If no name exists that matches the name referenced for a branch, TSM continues to the end of the command file and returns to the terminal user.

Examples

IFF %A EQ B (Branches when the user does not enter B as the value of parameter %A.)

IFF %A NE B (Branches when the user enters B as the value of parameter %A.)

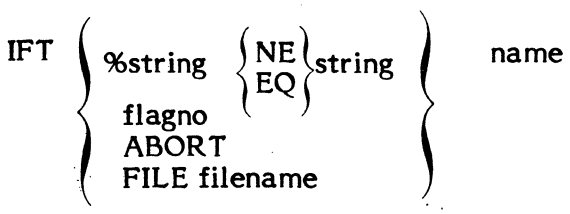
5.4.20 IFT Command (Command File Only)

The IFT command is used to branch to a name in a command file when a condition is true. There are several types of conditions that can be tested:

- o strings - the first character string is equal to the second character string.
- o flags - the specified flag or flags are TRUE. Note that flags are set to TRUE or FALSE via the SETF (TRUE) or RESETF (FALSE) commands.
- o ABORT - the preceding task aborts.
- o files - the specified file exists.

If a condition tests true, the user provides a name to branch to in the command file. If a condition does not test true, TSM continues processing with the next command on the command file.

Syntax:



where:

string	is any string, eight characters maximum (the string may be the result of argument substitution or concatenation). If more than eight, characters to the right are truncated. The value can be an alphanumeric string. If the value of the expression is true, TSM branches. If it is false, TSM processes the next command on the command file.
flagno	is a numeric flag number from 1-16. One number can be specified. If the specified flag is TRUE (set by the SETF command), TSM branches. If the flag tests FALSE, TSM processes the next command on the command file.
ABORT	if the task immediately preceding IFT aborts, TSM branches. If the task does not abort, TSM processes the next command on the command file.
FILE filename	specifies the name of a user or system file. If the file exists, TSM branches. If the file does not exist, TSM processes the next command on the command file.

name	is a one- to eight-character string defined via a DEFNAME or % label command that marks a point on a command file. When the user branches, he branches to one of the labels on the command file. If no label exists that matches the label referenced for a branch, TSM continues to the end of the command file and returns to the terminal user.

Example:

The example is used to show conditional processing for parameters. The NEW parameter is the point of concentration. The user can, at runtime, enter the string CREATE to create a new library and directory file before running the Subroutine Library Editor. If the string is absent, or other than CREATE, the existing library will be updated only.

Note that CREATE and DELETE tasks shown in the example are demonstration tasks only. (See Section 5.6.5, which uses DELETE as a sample interactive task.)

Command File (EXAMPLE):

```
DEFM SI,ASSEMBLER,NEW,OP
          .
          .
          .
IFT %NEW NE CREATE OLD
DELETE DIR
DELETE LIB
CREATE LIB, DM0800,128
CREATE DIR, DM0800,40
OPTION 1
%OLD
A4 LLO=UT
A1 DIR=DIR,,U
A1 LIB=LIB,,U
A1 LGO=OB%SI
LIBED
```

User Enters: TSM>EXAMPLE SRCE,ASSEMBLE,CREATE

Expanded Commands Are:

```
          .
          .
          .
DELETE DIR
DELETE LIB
CREATE LIB, DM0800,128
CREATE DIR, DM0800,40
OPTION 1
A4 LLO=UT
          .
          .
          .
```

The IFT command line is translated as "if the NEW parameter is any valid string other than CREATE, branch to OLD". If the runtime value string supplied is CREATE, delete existing library and directory files and create new file spaces, and select option 1 before making LIBED assignments. The name %OLD is used to move the runtime user past CREATE and DELETE tasks if he wants to update existing library files.

5.4.21 JOB Command

The JOB command is used to obtain an SGO file for TSM processing. SGO files are used to accumulate object code from an assembly or compilation and move it through cataloging or into a library. (See Section 7.6.) JOB and EOJ commands delimit commands pertaining to the utility processors that are using the SGO file. JOB and EOJ can be used interactively or on a command file. For TSM processing, all parameters except SGO are ignored.

Syntax:

JOB [jobname] [ownername] [,key] [SGO=size]

where:

jobname	is a one- to eight-character job identifier. If more than eight characters are specified, the first eight characters are used. Ignored by TSM.
ownername	is a one- to eight-character owner name. Ignored by TSM.
key	is a one- to eight-character key associated with the owner name in the M.KEY file, if any. Ignored by TSM.
size	specifies the number of 192-word blocks of disc space that are to be allocated for the SGO file. If the SGO field is omitted, the size of the SGO file is determined by the SYSGEN directive SGOSIZE (see Volume 3). A size of zero is interpreted as one.

5.4.22 LINESIZE Command

The LINESIZE command is used to dynamically modify the screen width defined previously for a terminal at SYSGEN. It remains in effect for the terminal until the user logs off.

Syntax:

LINESIZE maxchars

where:

maxchars	specifies the maximum character position to be displayed/(written to) the terminal in the range of 72-236. Anything less than 72 or greater than 236 is an error.

5.4.23 Message Command

A message can be included in a command file by using any valid TSM closing delimiter as the first nonblank character in a line. The message is displayed at the terminal when TSM comes to that point in the command file. This command is equivalent to the NOTE command. If used at the terminal, nothing happens.

Syntax:

{ , =) ; } message

where:

message	is any 72-character maximum message (predicated on starting the message at the beginning of the line).

5.4.24 NOTE Command

The NOTE command is used to send a message to the user's terminal. It is identical in function to the Message command.

Syntax:

NOTE message

where:

message	is a 72-character maximum message to be displayed on the terminal.

5.4.25 OPTION Command

The OPTION command supplies various options for running a task online.

Syntax:

```
OPTION [ n          ]
       [ DUMP       ]
       [ PROMPT     ]
       [ LOWER      ]
       [ CPUONLY    ]
       [ IPUBIAS    ]
       [ COMMAND    ]
       [ NOCOMMAND  ]
       [ ERROR      ]
       [ NOERROR    ]
       [ TEXT       ]
       [ NOTEXT     ]
```

where:

If a parameter is not specified, all previous options are cleared.

n — a number from 1-32 specifying a particular option available for an MPX-32 utility or user task. (See utility documentation OPTION sections.)

DUMP — specifies an automatic dump on abort for the MPX-32 utility or user task to be run. On abort, the task's area of memory will be dumped to an SLO file by TSM and printed (output to the LOD).

PROMPT — specifies an automatic prompt before read from the terminal (UT). The first three characters of the utility or task being run will be displayed by TSM. See Section 5.2.2.

LOWER — inhibits translation of lower case input to upper case, which allows the user to enter and edit lower case as well as upper case characters. Commands and key words can be entered in any combination of upper/lower case; the names of files must, however, be all upper case. See Section 5.2.2.

CPUONLY — specifies tasks are to be executed only by the CPU processor. See Section 5.2.2.

IPUBIAS — specifies IPU compatible tasks will be run by the IPU processor. Default: tasks are executed by the first available processor. See Section 5.2.2.

COMMAND — causes commands in a macro or command file to be echoed to the user's terminal and prefixed by an asterisk as they are read by TSM. Default.

NOCOMMAND inhibits commands in a macro or command file from being echoed to the user's terminal as they are read by TSM. Remains in effect until an EOF or EOM is encountered. Reset by OPTION COMMAND.

ERROR causes automatic interpretation and displays a description of abort code at the user's terminal when an abort occurs. Default.

NOERROR inhibits automatic interpretation and display of abort code description at the user's terminal. Reset by OPTION ERROR.

TEXT causes commands in a macro or command file to be echoed to the user's terminal and prefixed by an asterisk as they are read by a processor. Remains in effect until an EOF or EOM is encountered.

NOTEXT inhibits commands in a macro or command file from being echoed to the user's terminal as they are read by a processor. Default.

5.4.26 PAGESIZE Command

The PAGESIZE command is used to indicate how many consecutive output records (lines) to write at the terminal without an intervening read or CR FOR MORE message. It dynamically modifies the page size specified previously at SYSGEN. It remains in effect for the terminal until the user logs off.

A pagesize of zero can be specified to output lines without intervening CR messages. Be careful not to specify zero if a task you are using does not have a BREAK receiver. If you do, output cannot be stopped with the BREAK key.

Syntax:

PAGESIZE maxlines

where:

maxlines — specifies the maximum number of lines to display (write) before another read (or CR) from the terminal.

5.4.27 PROJECT Command

The PROJECT command is used to change the project name/number you are working under for accounting purposes. It causes the ACCOUNT utility to terminate accumulating data on the old project name/number and initiate accumulation of data on the new project name/number.

Syntax:

PROJECT number

where:

number — is a 1-8 character alphanumeric project name/number previously established and recognizable by the ACCOUNT utility.

5.4.28 RESETF Command

The RESETF command is used to specify a FALSE (=0) condition for up to sixteen distinct flags. The flags can then be tested by IFT and/or IFF commands within a command file. RESETF should precede any IFF or IFT commands that check the condition of the specified flags.

Syntax:

RESETF flagno [flagno] ...

where:

flagno — specifies a flag number in the range 1 through 16 (decimal). Any number of flags may reset with a single RESETF command. Flags remain reset false within a command stream until set true with a subsequent SETF command.

5.4.29 SCAN Command

The SCAN command is identical in function to the LINESIZE command. It is used to specify the length of a logical record (line) to 'print' (display at the terminal).

Syntax:

SCAN maxchars

where:

maxchars — specifies the last character position to be printed in the range 72-255. Anything less than 72 is an error.

5.4.30 SELECT Command

The SELECT command is used to read TSM commands from a file instead of from the terminal. The file must be blocked and uncompressed, i.e., use the EDITOR STORE command (rather than SAVE) when building it. TSM looks for the specified file under the user name that is in effect when the SELECT command is issued.

A SELECT command can be used to select another command file from the current command file. Processing continues at the beginning of the newly selected file; if there are any commands following SELECT on the first file, they are skipped. Any number of command files can be chained via SELECT commands.

Syntax:

SELECT cmdfile [par1][par2 ...]

where:

The command verb SELECT is optional. It can be used to avoid any ambiguity about whether a task or a command file is to be executed.

cmdfile — is the name of a permanent file containing TSM commands.

par1 — is a parameter to pass for the command file. Up to eight parameters can be passed as described in Section 5.3.5.

Response:

TSM commands which are read from the file are echoed to the terminal preceded by an asterisk (to indicate they are coming from the file). The commands are executed immediately. When a RUN or equivalent command is used to activate a task via the command file, TSM activates the task. The task is then run interactively, typically with input from the user at the terminal or input from the command file. When the task exits, TSM reads the next command on the command file, if any. Command file processing terminates at end of file, when the user issues a CLEAR command, or at an ENDM command.

Errors:

If a file allocation error occurs with a code number given, refer to Section 7.8.1 of this volume for a description of the error.

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

The CLEAR command can be used to terminate processing from the command file and return control to the terminal. If any valid command other than CLEAR is executed, TSM returns control to the command file for further processing.

Entering just a carriage return when TSM encounters an error on the command file causes TSM to skip the command that was invalid and execute the next command on the command file.

5.4.31 SETF Command

The SETF command is used to specify a TRUE (=1) condition for up to sixteen distinct flags. The flags can then be tested by IFT and/or IFF commands within a command file. SETF should precede any IFF or IFT commands that check the condition of the specified flags.

Syntax:

SETF flagno [flagno] ....

where:

flagno — specifies a flag number in the range 1 through 16 (decimal). Any number of flags can be set TRUE within a single SETF command. Flags remain set (TRUE) within a command stream until set FALSE with a subsequent RESETF command.

5.4.32 SIGNAL Command

The SIGNAL command is used to send a message to another logged on owner or to all terminals. See Section 5.1.6.

Syntax:

SIGNAL [ownername]

where:

ownername — is a valid owner name of another logged on user. If the owner is not logged on, the message is denied.

Default: No ownername sends the message to all terminals.

Response:

In response to SIGNAL, TSM types:

"ENTER MESSAGE"

The message is 72 characters maximum, terminated by a carriage return. The message will be transmitted unless sent to a specific owner who is not currently logged on.

5.4.33 USERNAME Command

The USERNAME command is used to change your current user name and gain access to another user's files as described in Section 5.1.7.

Syntax:

USERNAME [username] [key]

where:

username	is the name of a valid user on the M.KEY file. Default if no name supplied is no user name, i.e., system files are created and only system files are accessed.
key	specifies a valid key if required to use this user name.

Response:

If another person is concurrently logged on with the same user name you supply, the message

```
*WARNING* MEYERS ON 2003 HAS SAME USERNAME
TSM >
```

is displayed, where MEYERS is the owner name and 2003 is the terminal ID of the other person logged on with the same user name. The person who performed 'USER MEYERS' has access to the same files as owner Meyers logged on a different terminal.

A WHO entered at the TSM > prompt shows both owners logged on with the same user name:

```
TSM> WHO
ADDRESS    OWNERNAME  USERNAME
TY7E00     CONSOLE
TY2003     MEYERS MEYERS
* TY2009   MCNORTON   MEYERS
TSM>
```

5.4.34 WAIT Command

The WAIT command is used to put a terminal into a special wait state so that it can receive messages. For example, if you have submitted one or more batch jobs and want to see messages pertaining to the job(s) before continuing terminal operation or logging off, you can use WAIT. Messages output by job control will then be displayed as the job is processed. You can continue to wait for other messages (e.g., if you have submitted more than one job) or you can exit the wait state by depressing the wakeup character to return to normal interactive operation.

If WAIT is not used, an inactive terminal is in a read condition and a read is not interrupted by messages (unless the terminal times out). You have to issue carriage returns to get out of a read condition and display a message.

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

For a description of the job control messages output to a terminal, see Section 5.1.2.

Syntax:

```
WAIT
```

Example:

```
      EDT>COL
      1. $JOB X FADEN,G SLOF=SLOF1
      3. $EXECUTE FILEMGR
      4. LOGU
      5. $EOJ
      6. $$
1     EDT>BATCH
      EDT>EXIT
      TSM>WAIT
2     0002 FADEN   $EOJ  X
      JOB EXECUTION TIME =  00 HOURS- 00 MINUTES- 01.28 SECONDS
3     TSM><wakeup>
      TSM>
```

Comments:

1 The user submits the workfile displayed above.

2 When the job is done (successfully or with an abort), the batch end of job message is displayed as well as elapsed execution time. The form of the EOJ job message is:

```
jobno owner  $EOJ jobname
```

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

3 At this point the terminal is still in the special ANYW state. The user can leave it waiting to receive messages from other jobs that have completed or use the wakeup character to enter a TSM command.

5.4.35 WHO Command

The WHO command is used to show owner names and terminal addresses of all logged on terminal users.

Syntax:

```
WHO
```

Response:

TSM displays the device mnemonic of each active terminal with the owner name used to log on and the user name currently in effect for file access. An asterisk is displayed before the mnemonic of the terminal that issued the WHO command.

Example:

```
TSM>WHO

ADDRESS     OWNER NAME   USER NAME

TY7D00        CONSOLE
*TY2000       FADEN        FADEN
TY200B        MARK         MARK
TSM>
```

5.5 Sample Command Files

5.5.1 Example 1

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

```
JOB
OPTION 2 3 4 5
A4 LO=UT
ASSEMBLE
START       M.EXIT
            END START
$CATALOG LMSRCE
LMSRCE
EOJ
```

LESTER is then executed as:

```
TSM>LESTER
```

5.5.2 Example 2

This command file uses parameter substitution for assembling, cataloging, and executing a task. Cataloging and execution are treated as conditional parameters in the command file.

Command File (GEN):

```
1       DEFM SI,T,OPT
2       IFA %OPT 1
        OPTION %OPT
        %1
        OPTION 2 3 4
3       A1 SI=%SI
        A1 BO=%SI;BO
        A4 LO=UT
4       ASSEMBLE
5       IFT %T EQ C EXIT
        A1 SGO=%SI;BO
        CATALOG LM%SRCE U60 NOM
6       IFT %T NE XA EXIT
        LM%SRCE
        %EXIT
```

User Enters: TSM>GEN SFIL

Expanded Commands Are:

```
OPTION 2 3 4
A1 SI=SFIL
A1 BO=SFILBO
A4 LO=UT
ASSEMBLE
A1 SGO=SFILBO
CATALOG LMSRCE U60 NOM
LMSRCE
EXIT
```

1 Parameters are SI (base file name), T (C=do not catalog or execute; XA=catalog and execute, blanks or anything else = catalog but do not execute). OPT is a numeric option in addition to options 2, 3, and 4 (which are selected automatically by the command file).

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

3 Assign a file named in the first parameter to SI or use SI if nothing. Assign the same file name suffixed by BO for binary output. Note that a semicolon is used to concatenate the parameter with a string. Assign listed output to the terminal.

4 Execute the Assembler.

5 If user enters C as the second parameter, exit; else, make the SGO assignment and catalog.

6 If user enters anything other than XA as the second parameter, exit; else execute the task that has been cataloged on LMSRCE.

5.5.3 Example 3

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

```
        DEFM SI,ASSEMBLE,NEW,OP
1       DELETE OB%SI
        CREATE OB%SI,DM0800,20
        A4 LO=UT
        A1 BO=OB%SI
        A1 SI=%SI
2       IFA %OP ASSM
        OPTION %OP
        GOTO NOP
        %ASSM
        OPTION 1
        %NOP
        OPTION 3 4
3       IFT %ASSEMBLE EQ ASSEMBLE ASSM1
        OPTION 15
        ALLOCATE 20000
        %ASSEMBLE
4       GOTO EOC
        %ASSM1
        ASSEMBLE
        %EOC
5       IFT %NEW NE CREATE OLD
        DELETE DIR
        DELETE LIB
        CREATE LIB,DM0800,128
        CREATE DIR,DM0800,40
        OPTION 1
        %OLD
        A1 DIR=DIR,,U
        A1 LIB=LIB,,U
        A1 LGO=OB%SI
6       LIBED
        ENDM
```

1 The command file unconditionally deletes any existing object file, creates a new 20 block file space on DM08, and makes assignments for assembly.

2 If the user does not select an option, he gets options 1, 3, and 4.

3 If ASSEMBLE, skip option 15 and ALLOCATE, and run; else implement.

4 GOTO EOC moves past the alternate ASSEMBLE.

5 As described in previous examples, set up for output to either an old or a new subroutine library file.

6 Execute the Subroutine Library Editor.

5.6 Developing an Interactive Task

There are a number of functions that TSM automatically handles for a task that is activated in the online environment. It handles sequencing for all interactive tasks using time distribution priorities 55-64 and timesharing algorithms to provide maximum response to each terminal on an equal time slice basis. It returns the TSM prompt to the terminal when a task activated in the online environment aborts or exits. It handles break and wakeup interrupts to let the terminal user communicate with a task. It supplies override assignments for SYC that allow a task cataloged for reads from SYC to read instead from the terminal without special programming or cataloging. The user interaction in these cases is described in previous sections.

This section describes interactive services available to the programmer. None of these services have to be used in order to run a task interactively.

M.TSCAN - Access the TSM scanner

M.TBRKON - Used by a task to access TSM break handling capability as described in Section 5.2.3.

M.CONADB, M.CONAHB, M.CONBAD, and M.CONBAH - Convert ASCII hex or decimal values to binary and vice versa.

This section also describes how regular calls to IOCS (read, write, etc.) are handled by TSM for a task with an assignment to UT.

5.6.1 TSM Scanner (M.TSCAN)

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

The M.TSCAN service can be called by the user task to have TSM pass any fields supplied by the terminal user in addition to the task name.

For subsequent interface to the terminal, the task can also use the TSM scanner. Each read from a logical file code assigned to UT causes TSM to put the line typed at the terminal into the OS line buffer and initialize the scanner to point to the first field.

Functional Description

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

Comment:

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

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

A field accessed by this service is left-justified and blank-filled by TSM and stored into Registers 6 and 7.

Register 5 contains the character count of the last field found by the scanner. There are no more fields in the line when the character count in R5 is zero and the delimiter in R4 is a carriage return.

Valid delimiter characters for fields are blanks, commas, semicolons, equal signs, carriage return, and left or right parentheses. Therefore, these characters must not be used in file names.

The M.RWND service can be used to reset the cursor at the first field in the current input record following the task name. The input line can be scanned via M.TSCAN without any additional IOCS calls.

M.TSCAN ignores all blanks and left parentheses before the first parameter or delimiter encountered. If a delimiter is encountered before the first parameter, all blanks and left parentheses are still ignored until the first parameter is encountered. If multiple blanks, left parentheses, or a combination of them are used after the first parameter, the first one encountered is treated as the delimiter. Do not use any others.

Entry Conditions

Calling Sequence:

```
M.TSCAN

(or)

SVC  1,X'5B' or M.CALL  H.TSM,2
```

Exit Conditions

Return Sequence:

```
M.RTRN  4,5,6,7
```

Registers:

R4	delimiting character
R5	number of significant characters before delimiter
R6,7	left-justified character string

5.6.2 TSM Break Processor (M.TBRKON)

Functional Description

M.TBRKON processes a pause or break from the terminal or calling task. The user receives the following prompt on the terminal:

```
***BREAK*** ON taskname AT location
CONTINUE, ABORT, or DEBUG?
```

If the user enters C, the task resumes execution at the instruction following the call. If the user enters A, the task is aborted.

If the user enters D, TSM loads the Debugger as an overlay and transfers control to the Debugger.

Entry Conditions

Calling Sequence:

```
M.TBRKON
```

(or)

```
ZR   R2          or LW     R2,TCW
SVC  1,X'5C'     or M.CALL H.TSM,6
```

This routine is also the default receiver for any online task, and therefore will be entered as a result of a hardware or software break. If a user Transfer Control Word is loaded in register 2, it will be printed along with the break message. Fortran PAUSE uses this capability.

Exit Conditions

Output:

None

Return Sequence:

M.RTRN

Registers:

None

Abort Cases:

TS01	If user enters 'A' (Abort) to the prompt described above.

5.6.3 TSM Conversion Services

5.6.3.1 Convert ASCII Decimal to Binary (M.CONADB)

Entry Conditions

Calling Sequence:

```
M.CONABD [n]

(or)

LD   6,n
SVC  1,X'28' or M.CALL H.TSM,7
```

where:

n — is the address of a left-justified, doubleword-bounded, ASCII-coded decimal number, blank-filled. If not specified with M.CONADB, contents of registers 6 and 7 will be converted.

Exit Conditions

Return Sequence:

```
M.RTRN  6,7
```

Registers:

R6 — 0 if a character is non-numeric

R7 — binary equivalent of input

5.6.3.2 Convert ASCII Hexadecimal to Binary (M.CONAHB)

Entry Conditions

Calling Sequence:

```
M.CONAHB [n]

(or)

LD   6,n
SVC  1,X'29' or M.CALL  H.TSM,8
```

where:

n — is the address of a left-justified, doubleword-bounded, ASCII-coded decimal number, blank-filled. Takes registers 6 and 7.

Exit Conditions

Return Sequence:

```
M.RTRN  6,7
```

Registers:

R6	0 if a character is not hexadecimal
R7	binary equivalent of input

5.6.3.3 Convert Binary to ASCII Decimal (M.CONBAD)

Entry Conditions

Calling Sequence:

```
M.CONBAD [n]
```

(or)

```
LW  5,n
SVC 1,X'2A' or M.CALL  H.TSM,9
```

where:

n	the address of a positive binary number

Exit Conditions

Return Sequence:

```
M.RTRN  6,7
```

Registers:

R6,7	ASCII result, right-justified with leading ASCII zeros

5.6.3.4 Convert Binary to ASCII Hexadecimal (M.CONBAH)

Entry Conditions

Calling Sequence:

```
M.CONBAH [n]

(or)

LW  5,n
SVC 1,X'2B' or M.CALL H.TSM,10
```

where:

n	is the address of a binary number

Exit Conditions

Return Sequence:

```
M.RTRN  6,7
```

Registers:

R6,7	ASCII result, right-justified with leading ASCII zeros

5.6.4 Terminal I/O

TSM provides optional pre and post processing for user I/O requests on the TSM terminal. This processing is inhibited by certain control flags in the user's FCB; Word 3:

Bit	0	No-wait I/O
	2	Data formatting inhibited
	6	Expanded FCB

Otherwise, I/O operations on the terminal are restricted as indicated.

5.6.4.1 Reads

On a read (M.READ), the maximum input is limited to the width of the terminal (specified at SYSGEN). I/O is automatically buffered to enable task swapping during I/O wait. The TSM scanner is initialized at I/O completion. Error returns are not honored, but error status is returned in the FCB. Carriage returns are replaced by blanks, but the actual input byte count is returned in the FCB. The entire input buffer is blank-filled prior to input to insure proper parsing by the scanner. The special control characters described in section 5.1.5 are honored.

5.6.4.2 Writes

The maximum output record on a write (M.WRITE) is limited to the width of the terminal. A line counter is maintained to detect bottom of screen, and the bottom of screen logic (see Section 5.2.6) is in effect. Output is only buffered if required by the controller. Error returns are not honored, but error status is returned in the FCB. The carriage control characters in byte 0 (see Section 7, Table 7-7) are also in effect.

5.6.4.3 Close and Open

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

5.6.4.4 Rewind

M.RWND can be used to return the TSM scanner to the first field in the TSM line buffer.

5.6.5 Sample Interactive Task

```
$EXECUTE ASSEMBLE
         PROGRAM              DELETE
         LIST                 NOMAC
         M.EQUS
         M.TBLS
*        THE FUNCTION OF THIS TASK IS TO ACCEPT ONE TERMINAL
*        PARAMETER IN SEMI FILEMGR FORMAT AND DELETE THE
*        NAMED FILE.
*
*        FAILURE TO COMPLETE IS SIGNALED WITH A FAILURE
*        MESSAGE TO THE TERMINAL
*
*        INPUT FORMAT:
*                             DELETE FILENAME
*
*
DELETE   EQU                  $
         M.TSCAN                                 GET 1ST ARG
         STD                  R6,FILENAME        SAVE FOR DELETE
         M.DELETE                                FILENAME
         TRR                  R7,R7
         BNZ                  EXIT               NOERRS
         TRR                  R6,R5
C.MSG    EQU                  $
         M.FILE               TERM,RW            OPEN TERMINAL FILE
                                                 (OPTIONAL)

         M.CONBAD                                CONV BIN TO ASCII
                                                 DEC

         STB                  R7,ERRTYPE
         M.WRIT               TERM
         M.CLSE               TERM               OPTIONAL CLOSE
EXIT     EQU                  $
         M.EXIT                                  RET TO SYSTEM
*
*
*
FILENAME REZ                  1D
*
*
TERM     DATAW                C' UT '
         GEN                  12/B(TY.LEN),20/B(MESSAGE)
         REZ                  6W
MESSAGE  EQU                  $
         DATAB                C'"M"J DELETE FAILED..ERRTYPE  '
TY.LEN   EQU                  $-MESSAGE
ERRTYPE  EQU                  $-1B
         END                  DELETE
```

5.7 Terminal Initialization (INIT)

Terminal initialization is the process of defining hardware characteristics of each terminal that is to be used with the MPX-32 operating system. These characteristics include, for example, baud rate, parity, HALF or FULL duplex, etc.. The wakeup ('ring') character for all terminals is also defined as a hardware characteristic.

Characteristics are typically defined by a user-created system file named LOGONFLE, or in its absence, by a set of system supplied defaults.

Terminal initialization is handled automatically when a system is installed (after SYSGEN) or restarted. Additionally, the TSM INIT command can be used to reinitialize all terminals or just one terminal after the automatic execution.

A terminal connected to a TLC controller is not initialized via INIT. INIT initializes only terminals connected through model 9122 ADS, 9110 ALIM or 8510 8-line Asynchronous Communications controllers.

5.7.1 The LOGONFLE

The form of a LOGONFLE is shown below. LOGONFLE must contain a record for the wakeup character definition and one record for each terminal. Only characters 1-72 of each record are interpreted.

```
Record 1: wakeup

Record 2: ccaa field field  ...
          .
          .
          .
          EOF
```

where:

wakeup — is a 2-character field defining the hexadecimal character which must be typed at interactive terminals to start a log-on sequence. If LOGONFLE was not created, system default is X'3F' (question mark).

ccaa — is the channel number and subaddress of the terminal to initialize. Must be supplied in LOGONFLE for each terminal. If LOGONFLE was not created, system default is to initialize all addresses defined at SYSGEN as device type 'TY' with the system default parameters.

field — is a 1-8 character keyword (only the first 4 characters are significant) describing the characteristics of a terminal. Fields can be entered in any order and duplicated, and are evaluated left to right, thus later entries can overrule earlier inconsistent entries.

Note: The LOGONFLE file must contain the logon records in blocked, uncompressed format. The Editor STORE command can be used to create this file.

An asterisk (*) in column 1 from Record 2 on indicates that line is a comment line. A semicolon (;) or an exclamation point (!) in any position of a line from Record 2 on permits comments to follow on the line.

Keyword	ADS	ALIM	8-Line Async	Default
19200	Not used	Not used	19200	
9600	9600 bps	9600 bps	9600 bps	Yes
7200	7200 bps	7200 bps	7200 bps	
4800	4800 bps	4800 bps	4800 bps	
3600	3600 bps	3600 bps	3600 bps	
2400	2400 bps	2400 bps	2400 bps	
2000	Not used	Not used	2000 bps	
1800	1800 bps	1800 bps	1800 bps	
1200	1200 bps	1200 bps	1200 bps	
900	900 bps	900 bps	Not used	
600	600 bps	600 bps	600 bps	
300	300 bps	300 bps	300 bps	
150	150 bps	150 bps	150 bps	
134	134.5 bps	134.5 bps	134.5 bps	
110	110 bps	110 bps	110 bps	
75	75 bps	75 bps	75 bps	
50	50 bps	50 bps	50 bps	
EXT	External rate	External rate	Not used	
HOST	Host operation	Not used	Not used	No HOST
LOCAL	DTE mode	Not used	Not used	Yes
REMOTE	DTE off Dial-up bit set in UDT	Switched bit set	Switched bit set	
ODD	Odd parity	Odd parity	Odd parity	
EVEN	Even parity	Even parity	Even parity	Yes
NONE	No parity Decrease in size of the serial character due to the absence of the parity bit.	No parity	No parity	
HALF	Not used	Half duplex operation	Half duplex operation	Yes
FULL	Not used Full duplex bit set in UDT	Full duplex operation	Full duplex operation	

Keyword	ADS	ALIM	8-Line Async	Default
EFLAG	Bit 16 of LCW1	Not used	Not used	No flag
OFLAG	Bit 17 of LCW1	Not used	Not used	No flag
MODEM	DTE off Modem bit set in UDT	Modem	Hardware ring	
NOTSM	Non TSM device TSM bit turned off in UDT	Non TSM device	Non TSM device	
INIT	2 words	5 words	1 word	

This keyword can be used to set up specific initialization. Hex initialization information must be correct length. For 8-Line Asynch, the last byte of the word can be used to define a special read termination character. Refer to the appropriate technical manuals for specific initialization information.

B19	Bit 19 of LCW1 on (may not support TSM)	Not used	Not used	Not B19
S1	1 stop bit	1 stop bit	1 stop bit	Yes
S1.5	2 stop bits	1.5 stop bits	1.5 stop bits	
S2	2 stop bits	2 stop bits	2 stop bits	
8	8 bit characters	8 bit characters	8 bit characters	
7	7 bit characters	7 bit characters	7 bit characters	Yes
6	6 bit characters	6 bit characters	6 bit characters	
5	5 bit characters	5 bit characters	5 bit characters	

5.7.2 ADS Terminal Record Syntax and Defaults

ccaa [baud] [LOCAL / REMOTE] [HOST] [parity] [charsize] [stopbits] [MODEM] [NOTSM] [EFLAG] [OFLAG] [B19]

where:

ccaa — is the channel and subaddress of the terminal.

baud — specifies baud rate: 9600, 7200, 4800, 3600, 2400, 1800, 1200, 900, 600, 300, 150, 134, 110, 75, 50, or EXT. Note: If EXT is entered, the baud rate is set externally. Default: 9600 baud

LOCAL,REMOTE defines interface. Default: LOCAL

HOST — specifies current loop interface operation is desired. If not desired, leave the field blank. Default: the field is ignored.

parity — specify ODD, EVEN, or NONE. Default: EVEN. If NONE is specified, there is a decrease in size of the serial character due to the absence of the parity bit.

charsize — specify character size: 5, 6, 7, or 8. Default: 7

stopbits	specify number of stop bits: 1,2, or None. Default: 1
MODEM	sets modem bit in UDT. Default: not set
NOTSM	specifies a non-TSM device. Default: not set
EFLAG	bit 16 of LCW1 set. READY/RING/FLAG E (OUT). Default: reset.
OFLAG	bit 17 of LCW1 set. READY/RING/FLAG O (OUT). Default: reset.
B19	bit 19 of LCW1 set. HDX mode. Default: reset.

5.7.3 ALIM Terminal Record Syntax and Defaults

ccaa [baud] [HALF / FULL] [parity] [charsize] [stopbits] [REMOTE] [MODEM] [NOTSM]

where:

ccaa	is the channel and subaddress of the terminal
baud	specifies baud rate: 9600, 7200, 4800, 3600, 2400, 1800, 1200, 900, 600, 300, 150, 134, 110, 75, 50 or EXT. Note: If EXT is entered, the baud rate is set externally. Default: 9600 baud
HALF,FULL	specifies HALF or FULL duplex operation. Default: HALF
parity	specify ODD, EVEN or NONE. Default: EVEN. If NONE is specified, there is a decrease in size of the serial character due to the absence of the parity bit.
charsize	specify character size: 5, 6, 7 or 8. Default: 7
stopbits	specify number of stop bits: 1, 1.5, 2 or None. Default: 1
REMOTE	sets dial-up bit in UDT. Default: not set
MODEM	sets modem bit in UDT. Default: not set
NOTSM	specifies a non-TSM device. Default: TSM device.

5.7.4 8-Line Asynchronous Communications Controller Record Syntax and Defaults

ccaa [baud] [HALF / FULL] [parity] [charsize] [stopbits] [REMOTE][MODEM] [NOTSM]

where

ccaa	is the channel and subaddress of the terminal
baud	specifies baud rate: 19200, 9600, 7200, 4800, 3600, 2400, 2000, 1800, 1200, 600, 300, 150, 134, 110, 75, or 50. Defualt: 9600 baud
HALF,FULL	specifies HALF or FULL duplex operation. Default: HALF
parity	specify ODD, EVEN or NONE. Default: EVEN. If NONE is specified, there is a decrease in size of the serial character due to the absence of the parity bit.
charsize	specify character size: 5, 6, 7 or 8. Default: 7
stopbits	specify number of stop bits: 1, 1.5, 2 or None. Default: 1
REMOTE	sets dial-up bit in UDT. Default: not set
MODEM	sets modem bit in UDT. Default: not set
NOTSM	specifies a non-TSM device. Default: TSM device.

5.7.5 Sample LOGONFLE

```
1 Record   1  3F
2 Record   2  6000  9600   LOCAL          EVEN     7
  Record   3  6001  9600   LOCAL   HOST   EVEN     7  2
3 Record   4  6002  4800
  Record   5  2000  9600   FULL    EVEN   7        1.5
4 Record   6  2002
5 Record   7  2004  2400          ODD    S2
6 Record   8  2007  MODEM  REMOTE INIT   00001518 7A06CC00 01808000 80800580 80000000
```

Comments

1 Record 1 is the wakeup (ring in) character '?'.

Records 2, 3, and 4 are for ADS terminals.

2 Record 2 defaults to non-host operation.

3 Record 4 uses default values following the baud rate: LOCAL interface, non-HOST operation, EVEN parity, 7 as the character size, and 2 as the number of stop bits.

Records 5 and 6 are for ALIM terminals.

4 Record 6 uses all default values: 9600 baud rate, HALF duplex, EVEN parity, 7 as the character size, and 1.5 as the number of stop bits.

5 Record 7 uses a 2400 baud rate, ODD parity, and 2 stop bits

6 Record 8 configures a BELL 103F dial-in modem on an ALIM.

5.7.6 Using INIT

INIT is an interactive task that can be thought of as a TSM command. LOGONFLE is assigned for input by default.

TSM > INIT ccaa

where:

INIT — with no channel and subaddress uses the current version of LOGONFLE to reinitialize all terminals that are currently free to allocate.

ccaa — a specific terminal can be reinitialized by supplying the appropriate channel and subaddress (hexadecimal). The record from LOGONFLE that matches the channel and subaddress is used to reinitialize the terminal.

5.7.7 INIT Errors

Message	Description
DEVICE NOT PRESENT ADDR=ccaa	The specified terminal is not plugged in to the CPU.
DEVICE NOT TERMINATED, ADDR=ccaa	Specified terminal not plugged in on terminal end of line.
M.ALOC1 DENIAL,NO LOGON FILE, DEFAULT USED	There is no file named LOGONFLE on the system. Default parameters have been set. See Sections 5.7.1/5.7.3.
M.ALOC3 DENIAL,ADDR=ccaa	The terminal at the specified channel and subaddress is in use and cannot be initialized now.
NON TSM DEVICE,ADDR=ccaa	The channel and subaddress of a device other than a terminal has been specified in LOGONFLE.
NO UDT ENTRY FOR ADDR=ccaa	Specified terminal not SYSGEN'd.
ON ADDRESS ccaa, FIELD UNIDENTIFIED: xxxxxxxx	The string xxxxxxxx is not a valid keyword for a characteristic's statement.
TERMINAL SET-UP COMPLETE	Initialization is done (unsuccessful initialization will also generate this message).
NO LOGONFLE ENTRY FOR ADDR=ccaa	There is no entry in the LOGONFLE for the channel and subaddress specified by INIT ccaa.

6. BATCH PROCESSING

Batch processing consists of spooling batch jobs to disc, interpreting job control statements, and directing listed and punched spooled output to destination files and devices. The number of jobs which can be active concurrently is established at SYSGEN. Tasks comprising batch processing compete with each other and with nonbatch tasks for computer resources under standard MPX-32 allocation algorithms.

6.1 Job Flow

Each job is spooled to a separate System Control (SYC) disc file prior to processing as shown in Figure 6-1. Jobs may be spooled to SYC files from card, magnetic tape, and paper tape peripheral devices, and from blocked temporary and permanent disc files. The OPCOM BATCH command may be used to initiate spooling from peripheral devices and permanent files. The Submit Job From Disc File system service (M.CDJS) may be used to initiate spooling from permanent and temporary disc files.

When spooling of each batch job to its SYC file is complete, the job is assigned a sequence number in the range 1 through 9999. Job sequence numbers reflect the order that jobs are entered and uniquely identify each job and its tasks. A job is eligible for processing as soon as the complete job is spooled to disc.

Jobs are selected for processing in the order in which they are entered unless overridden by the OPCOM URGENT command. This command also specifies the priority at which batch tasks which comprise the job are to execute. Nonurgent batch tasks execute at the priority specified at SYSGEN. Since tasks from separate jobs compete independently for computer resources, the order in which jobs are completed is not necessarily the same as the order in which they were entered.

Upon job completion, a job's spooled listed and punched output is automatically routed to usable peripheral devices if no particular device(s) or permanent file(s) are specified for the job. Usable devices for automatic selection are specified via SYSGEN and OPCOM commands. Spooled output destination devices include line printers, card punches, magnetic tape, and paper tape. Spooled output is selected for processing based on the software priority of jobs and, within a given priority, on the order in which processing of jobs was completed.

Job data flow is illustrated in Figure 6-1.

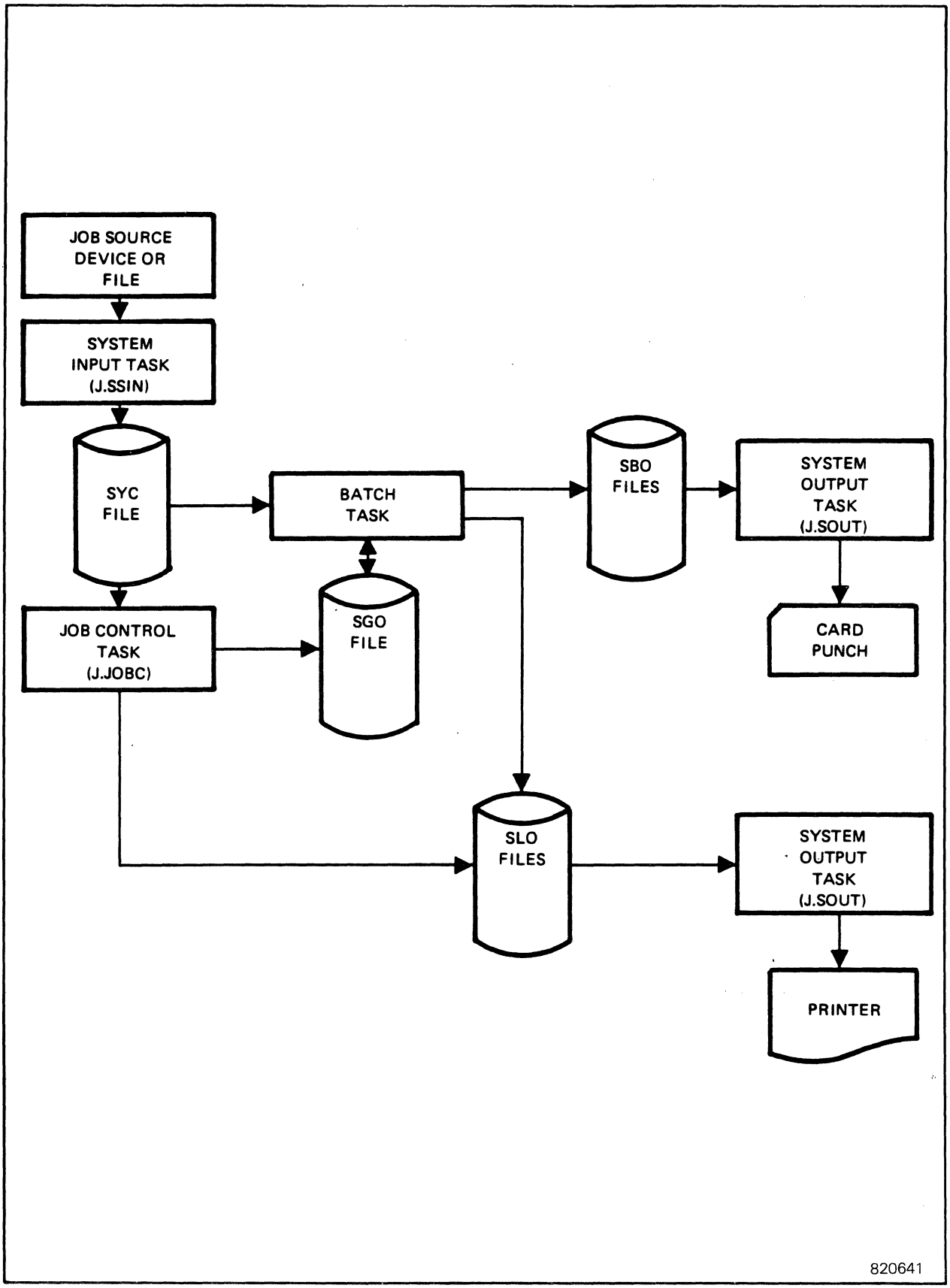


Figure 6-1. Data Flow for a Job

6.2 System Files

Four classes of system disc files provide intermediate storage for batch jobs.

SYC - System Control
SGO - System General Object
SLO - System Listed Output
SBO - System Binary Output

A separate System Control (SYC) file exists for each job and is used for disc buffering the job on input. SYC files are system disc files whose definitions are kept in a System Input Directory (M.SID). Jobs are spooled to SYC files from disc files and peripheral devices by System Input tasks. Job Control reads its statements from this file, and batch tasks read directive and data statements from it. (Batch tasks are those activated via $EXECUTE and $DEBUG statements. SYC files are automatically deleted when processing of the associated job is complete. SYC files are preserved through system restart if the associated job was not active. After system restart, processing of jobs in these files may be initiated by operator command.

System General Object (SGO) files are used for accumulating object records. A separate SGO file is automatically allocated for each job. Object records which follow the $OBJECT statement are written to SGO by Job Control; the Macro Assembler and compilers also write object records to SGO when optionally specified. The object records are accumulated on SGO until either the contents are read or until the end of a job is reached, whereupon the contents are deleted. Any batch task can read or write records to the job's SGO file.

System Listed Output (SLO) and System Binary Output (SBO) files are spooled output files used for listed output and punched output, respectively. The System Output tasks direct data from these files to destination peripheral devices. The definitions of these files are kept in a System Output Directory (M.SOD) and are automaticaly deleted when printing or punching of the files is complete. SLO and SBO files are preserved through system restart.

SYC files are automatically expanded by the System Input tasks as required to accommodate each job's size. The default size of each SGO file is established by SYSGEN but may be overridden via the JOB statement. SLO and SBO file sizes are specified on Cataloger, TSM, or Job Control ASSIGN statements or when dynamically allocated via system service calls.

Note: If one of these special system files is to be referenced by more than one logical file code, an ASSIGN2 and an ASSIGN4 must be used to allocate it rather than multiple ASSIGN2's.

Detailed information on the use of system files is contained in Section 7.

6.3 Spooled Input Control via $SELECT

Batch data are spooled to SYC files from devices and files specified by the OPCOM BATCH command, the Editor BATCH command, and the Submit Job From Disc File system service. These devices and files are designated as primary system input sources. While a job is being spooled to its SYC file, data from alternate sources may be merged with data from the primary source. Alternate system input sources are designated by $SELECT statements which may be included anywhere in a job followng the $JOB statement. When a $SELECT statement is encountered during input spooling, the $SELECT statement is not written to the SYC file but is replaced by data from the device or file specified on the statement. However, $SELECT statements which contain errors are written to the SYC file.

Each $SELECT statement encountered which specifies a valid device or file establishes a new alternate level. A maximum of three alternate levels is provided. The conditions at which an alternate level is reset to a previous level (alternate or primary) are summarized in Table 6-1.

A device or file may be specified on a SELECT statement which is the primary source or an alternate system input source at a previous level. If so, a new alternate level is established, but reading resumes from the device or file at its current position. This occurs for devices only if the device is identically specified, e.g., the specification MT1000 is not identical to MT10.

CONDITION	PROCESSING
I/O Error	On alternate source: perform end-of-job processing (ENDJOB). Reset to primary source. If primary source is a device, continue reading from device. If primary source is a file, terminate reading from the file. On device or file primary source: ENDJOB and terminate reading from device or file.
End-of-Medium	On alternate source: ENDJOB and reset to primary source. If primary source is a device, continue reading from device. If primary source is a file, terminate reading from the file. On device or file primary source: ENDJOB and terminate reading from device or file.
End-of-File	On alternate or primary card device source: pseudo end-of-file (0F record) written to SYC and continue reading from device. On alternate non-card device source: revert to previous level if specified number of files have been read. On primary non-card device source: continue reading from device. On file primary source: ENDJOB and terminate reading from file.
$EOJ	On alternate or primary source: ENDJOB and continue reading from source.
$$	On alternate source: ENDJOB and reset to primary source. If primary source is a device, continue reading from device. If primary source is a file, terminate reading from the file. On device primary source: ENDJOB and terminate reading from device if Continuous Batch mode is not set. If Continuous Batch mode is set, continue reading from device. On file primary source: ENDJOB and terminate reading from the file.
$$$	On alternate source: identical to $$. On device or file primary source: terminate reading from device or file.

Table 6-1 Terminating Conditions for Spooled Input Processing

6.4 Deck Organization

Job Control statements are processed sequentially by the Job Control task. Certain statements establish preconditions for a task's execution. These consist of $USERNAME, $ASSIGN, $ALLOCATE, and $OPTION. These statements apply to the task executed via the subsequent $DEBUG, $EXECUTE or $ACTIVATE statement within the job. $ASSIGN, $ALLOCATE, and $OPTION statements remain in effect only for the task subsequently executed. A $USERNAME statement remains in effect within a job until overridden by another $USERNAME statement.

Tasks placed into execution by $DEBUG and $EXECUTE (but not by $ACTIVATE) statements may read data or directive statements from the SYC file. These statements immediately follow the $DEBUG or $EXECUTE statement. An end-of-file condition is generated if the task attempts to read a Job Control statement. $SELECT statements do not generate this condition since they are replaced during input spooling.

6.5 Job Control Statements

All Job Control statements begin in column 1 with a dollar symbol ($) prefix. By default, positions beyond column 72 on Job Control statements are not interpreted. All numeric entries are decimal unless otherwise noted. Fields are separated by one or more blank spaces.

Some Job Control statements may be abbreviated. Underlining is used in the syntax statements which follow to show the valid abbreviations.

SELECT statements are a special form of job control used to include records on the SYC file from alternate system input sources while the job is being spooled to its SYC.

SELECT statements which specify a valid device or file are replaced by data from that alternate device or file. SELECT statements are not written to the SYC file unless they contain errors.

Conditional batch processing is provided by the Job Control statements $SETF, $RESETF, $IFT, $IFF, and $DEFNAME.

Statement	Function
$ACTIVATE*	Initiates a real time task.
$ALLOCATE	Increases cataloged memory for execution.
$ASSIGN1	Associates permanent disc file (optionally unblocked) with lfc. Default is blocked.
$ASSIGN2	Associates system SBO, SLO, SYC, or SGO file with lfc.
$ASSIGN3	Associates device (optionally unblocked) with lfc. If channel/subaddress specified, denotes specific device. Default for tapes and discs is blocked. Option for unblocking applies only to these units.
$ASSIGN4	Associates lfc with another lfc.
$DEBUG	Loads task with Debugger as an overlay.
$DEFNAME	Delimits an IFT, IFF, or GOTO processing sequence and controls conditional processing.
$EOJ	Specifies end of job.
$EXECUTE	Initiates system batch processor: FORTRAN, FILEMGR, etc.
$GOTO	Causes an unconditional branch to a name in the input stream which is defined by a DEFNAME statement.
$IFF	If false, test.
$IFT	If true, test.
$JOB	Identifies the job to the system. Allows specification by job of final destination for SLO and SBO files, if desired. Owner name is specified.
$NOTE	Writes a note to the OPCOM console.
$OBJECT*	Used at the beginning of an object code "deck".
$OPTION	Specifies program numeric options (1-20), system DUMP option, or CPUONLY/IPUBIAS option. NOMAP and NOEXEC options are not supported.
$RESETF	Resets false flags for conditional processing.
$SCAN	Specifies last column to read on JCL statement (80 maximum).
$SELECTD*	Reads batch from specified device.

* Not a valid command under TSM. (See Section 5.4).

$SELECTF*	Reads batch records from specified disc file.
$SELECTLD*	Reads records in library format from device file created via Source Update.
$SELECTLF*	Reads records from library formatted disc file created via Source Update.
$SELECTS*	Resets all alternate system input source levels. Reading of batch data reverts to primary system input source.
$SETF	Sets true flags for conditional processing.
$USERNAME	Overrides default user name for files in subsequent $SELECT statements and $EXECUTE statements until overridden by another $USERNAME statement.
$$ and $$$	Follows last $EOJ in job stream. $$ terminates a batch input stream if noncontinuous. $$$ terminates if in continuous batch mode. (If continuous batch is specified via the OPCOM MODE command, $$ is ignored and processing terminates at $$$.)

* Not a valid command under TSM. (See Section 5.4).

6.5.1 $ACTIVATE Statement

The $ACTIVATE statement initiates the execution of a real-time task. Processing of Job Control statements continues immediately after the task is activated.

Syntax

$ACTIVATE loadmod

where:

loadmod — is a load module name. If the load module is cataloged as not available for multicopying and a copy is currently executing, a message is output to SLO and processing continues with the next Job Control statement.

Usage

The task is activated at its cataloged priority with its pseudonym set to the job's sequence number. The task's owner name is set to the owner name from the $JOB statement. Tasks activated via this statement may not perform I/O to SYC or SGO files.

Note that $ASSIGN and $OPTION statements can be used prior to $ACTIVATE to assign files or devices and set option bits for the real time task.

6.5.2 $ALLOCATE Statement

The $ALLOCATE statement provides for an increase in the amount of memory allocated for a task's subsequent execution.

Syntax

$ALLOCATE bytes

where:

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

Usage

This statement increases the task's memory requirements that were established when the task was cataloged. The statement applies to the task subsequently executed via $DEBUG, $EXECUTE or $ACTIVATE. If the number of bytes specified is less than the task's cataloged memory requirements, the statement is ignored.

6.5.3 $ASSIGN1 Statement

The $ASSIGN1 statement associates a logical file code with a permanent disc file.

Syntax

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

where:

lfc	is a one- to three-character logical file code contained in a File Control Block (FCB) of a task.
filename	is the one- to eight-character name of a permanent disc file.
password	is an optional one- to eight-character password associated with the file. An entry is required only if access if restricted, and the type of program usage (read or write) is permitted by password only.
U	is an indicator that the file is to be accessed in the unblocked mode. If "U" is not entered, blocked access is assumed.

Usage

Assignments establish linkage between logical file codes and peripheral devices and files for I/O operations. Assignments made by Job Control statements are temporary and exist only for the task subsequently executed via $DEBUG, $EXECUTE or $ACTIVATE. These assignments augment or replace assignments that were made when the task was cataloged. If the logical file code in the assignment matches one made at catalog time, it replaces the previous assignment. If not, it is added to cataloged assigments. Multiple assignments, separated by one or more blank spaces, may be made with $ASSIGN1, $ASSIGN2, $ASSIGN3, and $ASSIGN4 statements. If more than three characters are entered for a logical file code, only the first three are used.

6.5.4 $ASSIGN2 Statement

The $ASSIGN2 statement associates a logical file code with a system file.

Syntax

$ASSIGN2 lfc=file,size [lfc= ...]

where:

lfc	is a one- to three-character logical file code contained in a File Control Block (FCB) in a task.
file	is the three-character system file mnemonic. One of four entries may be made: SBO (System Binary Output file) SLO (System Listed Output file) SGO (System General Object file) SYC (System Control file) "SGO" or "SYC" may not be entered if the subsequent task within the job is brought into execution by a $ACTIVATE statement.
size	is the amount of file space required. An entry is made only for the punched (SBO) and listed (SLO) output files. The size is expressed for the respective files as the number of cards and the number of print lines.

Usage

Refer to the $ASSIGN1 statement.

6.5.5 $ASSIGN3 Statement

The $ASSIGN3 statement associates a logical file code with a peripheral device or a temporary disc file.

Syntax

$$\underline{\$ASSIGN3}\ lfc=devmnc\left\{\begin{array}{l},blocks\\ ,reel,[vol]\end{array}\right\}\ [,U]\ [lfc=\ ...]$$

where:

lfc	is the logical file code used in the task.
devmnc	is the device mnemonic of a configured peripheral device. See Appendix A.
blocks	number of disc blocks (192 words) to be allocated for this file.
reel	specifies a 1-4 character identifier for the reel. 'SCRA' (scratch) cannot be used for multivolume tapes.
vol	if multivolume tape, indicates volume number. Default: 0 (not multivolume)
U	specifies the tape or disc is optionally unblocked. Default: Blocked

Note: There must be no embedded blanks within a lfc assignment. Commas must be inserted for all nonspecified options (see Examples). One or more blanks are the legal separator between one lfc assignment and the next.

Examples

```
Tape:   A3 IN=M91000,SRCE,,U  OT=PT
Disc:   A3 IN=DC,20
```

Usage

Refer to the $ASSIGN1 statement.

6.5.6 $ASSIGN4 Statement

The $ASSIGN4 statement associates a logical file code with another logical file code.

Syntax

$ASSIGN4 lfc1=lfc2 [lfc= ...]

where:

lfc1 is a one- to three-character logical file code contained in a File Control Block (FCB) in a batch task. This logical file code is to be associated with a file or device indirectly through a second logical file code, lfc2.

lfc2 is a one- to three-character logical file code contained in a FCB in a task. This file code is directly associated with a file or device via a previous $ASSIGN1, $ASSIGN2, or $ASSIGN3 or indirectly via an $ASSIGN4 statement or cataloged assignment.

Usage

Refer to the $ASSIGN1 statement.

6.5.7 $DEBUG Statement

The $DEBUG statement places the specified task in execution as a batch task under control of the Debug processor.

Syntax

$DEBUG loadmod

where:

loadmod	is a load module name. If the load module is cataloged as unique (single-copied) and a copy is currently executing, a message is output to the SLO and processing continues with the next Job Control statement.

Usage

The task is activated with its psueodnym set to the job's sequence number. The task's owner name is set to the owner name from the $JOB statement. Directive or data statements to be read by the specified batch task from the SYC file follow the $DEBUG statement.

6.5.8 $DEFNAME Statement

The $DEFNAME statement is used in conjunction with the $IFT and $IFF Job Control statements to control batch input stream processing.

Syntax

$DEFNAME name

where:

name	specifies a one- to eight-character name corresponding to a name entry of a previous $IFT or $IFF Job Control statement. The name must contain at least one nonnumeric character.

Usage

This statement is used to delimit an $IFF and/or $IFT conditional processing sequence. If the specified "name" matches that given with a previous $IFF and/or $IFT statement for which the tested conditions have been satisfied, batch stream processing resumes with the next Job Control statement after the $DEFNAME statement. If the specified "name" does not match that given with the $IFF and/or $IFT statement that causes processing to be suspended, the $DEFNAME statement has no effect on batch stream processing.

6.5.9 $EOJ Statement

The $EOJ statement signals the end of a job.

Syntax

$EOJ

Usage

This is the last statement of a job.

6.5.10 $EXECUTE Statement

The $EXECUTE statement initiates the execution of a batch task.

Syntax

$ EXECUTE loadmod

where:

EXECUTE	is an optional word. If EXECUTE is not used, the $ sign is still required with no embedded blanks (i.e., $loadmod) and loadmod cannot be a keyword.
loadmod	is a load module name. A system utility (see Volume 2) or a user task can be specified. (Note: the name of a load module file is identical to the task name on MPX-32.) If the load module is cataloged as not available for multicopying and a copy is currently executing, a message is output to the SLO and processing continues with the next Job Control statement.

Usage

The task is activated with its pseudonym set to the job's sequence number. The task's owner name is set to the owner name from the $JOB statement. Directive or data statements to be read by the specified batch task from the SYC file follow the $EXECUTE statement.

$EXECUTE GO is not supported.

6.5.11 $GOTO Statement

The GOTO statement is used to unconditionally branch to a name in the batch input stream. The name to branch to must be defined by a DEFNAME statement.

Syntax

$GOTO name

where:

name	is a one- to eight-character name that appears in a subsequent DEFNAME statement.

Usage

The unconditional GOTO provides a means of suspending batch stream processing until a DEFNAME statement is encountered with the same name specified in the GOTO statement or until an EOJ statement is encountered.

6.5.12 $IFF Statement

The $IFF statement causes specified condition(s) to be tested for false values. If the condition is satisfied, batch input stream processing is altered according to parameter specifications included with the statement. The $IFF statement has three formats, as follows:

Syntax 1

$IFF [flagno] flagno ... name

where:

flagno	specifies the flag number in the range 1-16 (decimal).
name	is a one- to eight-character name that appears on a subsequent $DEFNAME statement.

Usage

This form of the $IFF statement provides for specifying up to 16 distinct flags, which, if any are false (see $RESETF), result in batch stream processing being suspended until a $DEFNAME statement is encountered that specifies a "name" matching that given in the $IFF statement; or until an $EOJ statement is encountered. If no flag tests false, batch stream processing continues with the next Job Control statement.

Syntax 2

```
$IFF ABORT name
```

where:

name	is a one- to eight-character name that appears on a subsequent $DEFNAME statement.

Usage

If the immediately preceding task within the current job was not aborted, this variation of the $IFF statement causes batch stream processing to be suspended until a $DEFNAME statement is encountered that specifies the "name" matching that in the $IFF statement; or until an $EOJ statement is encountered. If the previous batch job processing was aborted, batch stream processing continues with the next Job Control statement.

Syntax 3

```
$IFF FILE filename name
```

where:

filename	is a one- to eight-character file name.
name	is a one- to eight-character name that appears in a subsequent $DEFNAME statement.

Usage

If the specified "file" is not defined in the SMD as a system or user (based on any preceding $USERNAME statement) file, batch input stream processing is suspended until a $DEFNAME statement is encountered that specifies a "name" matching that given in the $IFF statement; or until an $EOJ statement is encountered. If the file is defined to the system, batch stream processing continues with the next Job Control statement.

6.5.13 $IFT Statement

The $IFT statement causes specified condition(s) to be tested for the true value. If the condition is satisfied, batch input stream processing is altered according to parameter specifications included with the statement. The $IFT statement has three formats, as follows:

Syntax 1

```
$IFT  flagno [flagno]...  name
```

where:

flagno	specifies a flag number in the range 1-16 (decimal).
name	is a one- to eight-character name that appears in a subsequent $DEFNAME statement.

Usage

This form of the $IFT statement provides for specifying up to 16 distinct flags, which, if true (see $SETF), result in batch stream processing being suspended until a $DEFNAME statement is encountered that specifies a "name" matching that given in the $IFT statement; or until an $EOJ statement is encountered. If any flag tests false, batch stream processing continues with the next Job Control statement.

Syntax 2

```
$IFT  ABORT  name
```

where:

name	is a one- to eight-character name that appears in a subsequent $DEFNAME statement.

Usage

If the immediately preceding batch task was aborted, this variation of the $IFT statement causes batch stream input processing to be suspended until a $DEFNAME statement is encountered that specifies a "name" matching that given in the $IFT statement; or until an $EOJ statement is encountered. If the previous batch task was not aborted, batch stream processing continues with the next Job Control statement.

Syntax 3

```
$IFT FILE filename name
```

where:

filename	is a one- to eight-character file name.
name	is a one- to eight-character name that appears in a subsequent $DEFNAME statement.

Usage

If the specified file is defined in the SMD as a system or user (based on any preceding $USERNAME statement) file, batch input stream processing is suspended until a $DEFNAME statement is encountered that specifies a "name" matching that given in the $IFT statement; or until an $EOJ statement is encountered. If the file is defined to the system, batch stream processing continues with the next Job Control statement.

6.5.14 $JOB Statement

The $JOB statement identifies the job to the system. It is required as the first statement of a job, and together with the $EOJ statement, it delimits a job.

Syntax

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

where:

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

ownername — is a one- to eight-character owner name.

key — is a one- to eight-character key associated with the owner name in the M.KEY file if any.

size — specifies the number of 192-word blocks of disc space that are to be allocated for the job's SGO file. If the SGO field is omitted, the size of the job's SGO file is determined by the SYSGEN directive SGOSIZE (see Volume 3). A size of zero is interpreted as one.

number — specifies the number of SLO or SBO files to be generated by the Batch Job. A separate SLO and SBO file is created for each job and SLO or SBO assignment within the job. A queue is created for these SBO/SLO files. The queue is large enough to contain the definitions of 768 separate SLO or SBO files. The SLO and SBO fields may be used to increase or decrease the length of their respective queues. To discard all SLO or SBO files generated by the job, enter zero. If zero is entered for SLO, listings of Job Control statements and statement error messages are also discarded.

Default: Up to 768 files are generated, as required by the job.

SLOD/F
SBOD/F

These fields specify the final destinations of SLO and SBO files generated by the job. If these fields are omitted, final destinations for SLO and SBO files are automaticaly selected from eligible devices specified by SYSGEN DEVICE directives (the SPOOLED parameter). Fields in the $JOB statement are interpreted as follows:

SLOD — Any SLO files generated by the job are to be output to the specified device.

SLOF — Any SLO files generated by the job are to be output to the specified permanent disc file.

SBOD — Any SBO files generated by the job are to be output to the specified device.

SBOF — Any SBO files generated by the job are to be output to the specified permanent disc file.

devmnc — is the six-character device mnemonic of the final destination device for SLO or SBO files. The specified device may be a card punch, paper tape punch, line printer, or magnetic tape device. The entry consists of a two-character device code followed by a four hexadecimal character device address, e.g., LP7A00. (See Appendix A.) The device address consists of a two-character device channel number followed by a two-character device subaddress. If the subaddress is omitted, zero is assumed. If the entire device address is omitted, a channel number and subaddress of zero are assumed. Reel identifiers are not entered for magnetic tape devices. For assignments to magnetic tape devices, the reel identifier on the output tape is SLO or SBO, the tape is multivolume (beginning with volume 1) and the tape is blocked.

file — is the identity of the final destination permanent disc file for SLO or SBO files in the following form:

filename [,[[username],[key]][,password]

If the specified file does not exist, a file is dynamically created to contain the job's SLO or SBO. If the specified file does exist, the job's SLO or SBO is copied to it if the file is large enough. If not large enough, the file is deleted, and a file of adequate size is dynamically created. If a password is specified, dynamically created files have password-only access. If disc space is unavailable to dynamically create a file or an invalid file name or password is specified, output is redirected to a device available for automatic selection.

S — specifies that the job is to be run sequentially. A sequential job is not run until all previously entered sequential jobs are completed.

Usage

This is the first statement of a job and must be the first statement on a primary system input source. When the job becomes active, the job sequence number, owner name, and job name are listed on the operator's console.

Fields following the "ownername" field may be entered in any order. If the fields following the "ownername" exceed the length of a single statement, the fields may be entered on continuation statements. Continuation statements must immediately follow the $JOB statement. Continuation statements do not contain a "$" symbol in column 1 but contain specification fields beginning in any column through 72.

6.5.15 $NOTE Statement

The $NOTE statement provides a means of writing a message to the operator's console.

Syntax

$NOTE [message]

where:

message is an optional field containing information to be printed with the card image.

Usage

The job's sequence number together with the contents of columns 1 through 65 are listed on the operator's console when the statement is processed by Job Control.

6.5.16 $OBJECT Statement

The $OBJECT statement serves as a precursor for a program object deck.

Syntax

$OBJECT

Usage

The program object deck which follows the statement is stored on the SGO file. More than one deck may follow the statement. The last deck is terminated by the next Job Control statement. A $SELECT statement does not terminate the deck.

6.5.17 $OPTION Statement

The $OPTION statement designates options for the task subsequently executed.

Syntax

$OPTION number [number] ...

where:

number is a numeric entry from 1 through 20

or

the system option, as follows:

DUMP	if this option is specified and the batch task aborts, the task's logical address space is dumped.
CPUONLY	if this option is specified, tasks are executed only by the CPU processor.
IPUBIAS	if this option is specified, IPU compatible tasks are executed by the IPU processor. Default: tasks are executed by the first available processor.

Usage

Options provided by tasks are specific to each task and are designated by numbers. The statement applies to the task subsequently executed via $DEBUG, $EXECUTE or $ACTIVATE. Options associated with system batch tasks are explained in their respective sections.

A bit is set in the task option word for each option specified in the $OPTION statement. Bits 12 through 31 of the option word are processor options and correspond respectively to options 20 through 1. The Task Option Word Inquiry system service provides a task with the contents of its option word. Control of any task can be accomplished through its interpretation of the word.

Note

One $OPTION statement may be used to specify multiple options. Option specifications must be separated by one or more blank spaces.

For valid OPTION numbers and their descriptions, refer to the MPX-32 Reference Manual, Volume 2, under the particular utility you are using (Assembler, Source Update, etc.).

6.5.18 $RESETF Statement

The $RESETF statement provides for specifying up to sixteen distinct flags that may be tested by subsequent $IFT and/or $IFF Job Control statements within a job. Flags set with the statement will have a FALSE (=0) Boolean value.

Syntax

$RESETF flagno [flagno]...

where:

flagno specifies a flag number in the range 1 thrugh 16 (decimal). Any number of flags may be reset with a single $RESETF statement. Flags remain reset within a job until set with a subsequent $SETF statement.

6.5.19 $SCAN Statement

The $SCAN statement specifies the last column to be scanned on all Job Control statements within a job. If no $SCAN statement is present, 72 is the last column scanned.

Syntax

$SCAN number

where:

number is a numeric entry in the range 4-80 specifying the last position to be scanned on Job Control statements.

6.5.20 $SELECTD Statement

The $SELECTD statement causes ensuing batch records to be spooled to the SYC file from the specified peripheral device.

Syntax

$SELECTD devmnc reel [mode] [density] [parity] [NORE]
[UNBLOCKED] [fs fr]

where:

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

The following fields are applicable only for magnetic tape devices and floppy discs SYSGENed as magnetic tape:

reel is a four-character tape reel identifier.

The following three fields are applicable only for 7-track magnetic tape format as follows:

mode is a character which specifies the 7-track magnetic tape format as follows:

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

If this field is omitted, the interchange mode is assumed.

density is a character which specifies the 7-track magnetic tape density as follows:

H = 800 BPI
L = 556 BPI

If this field is omitted, 800 BPI is assumed.

parity — is a character which specifies the 7-track magnetic tape parity as follows:

E = even parity
O = odd parity

If this field is omitted, even parity is assumed.

The following fields are applicable for magnetic tape only and floppy discs SYSGENed as magnetic tape.

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

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

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

fr — is the number of files to be read from the magnetic tape. If this parameter is omitted, one file is read. End-of-file marks read from the tape are not written to the SYC file.

If "fs" is entered, "fr" must also be entered, and vice versa.

6.5.21 $SELECTF Statement

The $SELECTF statement causes ensuing batch records to be spooled to the SYC file from the named, permanent disc file.

Syntax

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

where:

file — is a one- to eight-character permanent file name. If the file is the primary or alternate system input source at a previous level, the following fields are not interpreted. To select a file belonging to a username different than the user name/ownername selected on the $JOB statement, see the $USERNAME statement.

UNBLOCKED — specifies that the disc file was written in the unblocked mode. If this parameter is not entered, blocked mode is assumed.

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

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

fr — is the number of files to be read. If this parameter is omitted, one file is read. End-of-files are not copied to the SYC file. If "fs" is entered, "fr" must also be entered, and vice versa. If the UNBLOCKED option is requested, fr must equal one or not be specified.

password — is the one- to eight-character password associated with the file. An entry is required if read access to the file is permitted only by password.

6.5.22 $SELECTLD Statement

The $SELECTLD statement causes ensuing batch records to be spooled to the SYC file from the specified peripheral device whose data is in library format. Data in library format is created by the Source Update processor (UPDATE).

Syntax

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

[NORE] [UNBLOCKED] header

where:

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

The following fields through "UNBLOCKED" are applicable only for magnetic tape devices and floppy discs SYSGENed as magnetic tape.

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

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

mode — is a character which specifies the 7-track magnetic tape format as follows:

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

If this field is omitted, interchange mode is assumed.

density — is a character which specifies the 7-track magnetic tape density as follows:

H = 800 BPI
L = 556 BPI

If this field is omitted, 800 BPI is assumed.

parity — is a character which specifies the 7-track magnetic tape parity as follows:

E = even parity
O = odd parity

If this field is omitted, even parity is assumed.

The following fields are applicable only for magnetic tape devices and floppy discs SYSGENed as magnetic tape.

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

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

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

6.5.23 $SELECTLF Statement

The $SELECTLF statement causes ensuing batch records to be spooled to the SYC file from the named, permanent disc file which is in library format. Library formatted disc files are created by the Source Update (UPDATE) system processor.

Syntax

$SELECTLF file [UNBLOCKED] header [password]

where:

file — is a one- to eight-character permanent file name. If the file, is a $USERNAME statement specifying the user name must precede the $SELECT statement within the job. If the file is the primary or an alternate system input source at a previous level, the following fields are not interpreted.

UNBLOCKED — specifies that the disc file was written in the unblocked mode. If this parameter is not entered, blocked mode is assumed.

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

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

password — is the one- to eight-character password associated with the file. An entry is required only if read access to the file is permitted only by password.

6.5.24 $SELECTS Statement

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

Syntax

```
$SELECTS
```

6.5.25 $SETF Statement

The $SETF statement provides for specifying up to sixteen distinct flags that may be tested by subsequent $IFT and/or $IFF Job Control statements within a job. Initially all flags have a FALSE (=0) Boolean value. Flags set with this statement will have a TRUE (=1) Boolean value.

Syntax

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

where:

flagno — specifies a flag number in the range 1 through 16 (decimal). Any number of flags may be set with a single $SETF statement. Flags remain set within a job until reset with a subsequent $RESETF statement.

6.5.26 $USERNAME Statement

The $USERNAME statement specifies the user name associated with a set of user files. The user name applies to files specified on subsequent $SELECT statements and to all files referenced by tasks activated by subsequent $EXECUTE, $DEBUG and $ACTIVATE statements within the current job. The user name is set to the owner name specified in a $JOB statement whenever the $JOB statement is encountered. This default user name/owner name association can be overridden by a $USERNAME statement.

Syntax

$USERNAME [username [,key]]

where:

username — is the name of a valid user on the M.KEY file. Default if no name supplied is no user name, i.e., system files are created and only system files are accessed.

key — specifies a valid key if required to use this username.

6.5.27 $$ Statememt

The $$ statement terminates a batch input stream.

Syntax

$$

Usage

The $$ statement follows the $EOJ statement on the last job in the batch stream. On encountering this statement, the System Input task terminates processing of batch statements. This statement is ignored if read from a primary system input source device and if Continuous Batch mode has been requested via Operator Communications.

6.5.28 $$$ Statement

The $$$ statement terminates a batch input stream when the Continuous Batch mode has been requested.

Syntax

$$$

Usage

This statement is used to terminate a batch stream when the Continuous Batch mode has been requested via Operator Communications. If Continuous Batch mode has not been requested, this statement is treated as a $$ statement.

6.6 Job Accounting

Accounting facilities are provided for batch jobs which indicate elapsed time and CPU time used for each job. Job accounting information can be listed, saved, and purged by operator commands.

The job accounting function optionally collects job information on an accounting file which is also used by TSM. The file must be named M.ACCNT and may be created by the File Manager CREATE directive. See OPCOM LIST command.

6.7 Punched Output

Each job's punched spooled output is preceded by a banner card which contains the job's sequence number. Punched spooled output from a real-time task is preceded by a banner card which contains the first four characters of the task number. Banner cards are output to card punch devices only.

6.8 Listed Output

Each page of spooled output printed by a System Output task contains a heading line consisting of the date, identification, MPX-32 release level, and a page number. The date printed is that which is specified by the operator command, ENTER. For batch output, the identification is the job name from the $JOB statement and the job sequence number. For output from nonbatch tasks, the identification is the task number.

Each Job Control statement is written on Spooled Listed Output (SLO) when it is processed by the Job Control task. Job Control statements that are not processed as a result of conditional assembly specifications will appear on listed output with the leading $ character converted to a left parenthesis, e.g., $NOTE would be listed as (NOTE.

$SELECT statements are not written on listed output unless they contain errors. Other messages written on listed output are described below.

6.8.1 Task Aborted

TASK ABORTED PSD= BIAS= REASON=

The execution of the task specified on the preceding $EXECUTE or $DEBUG statement was aborted. The contents of the PSD, the relocation offset bias, and abort reason are listed.

6.8.2 Activity Deleted

ACTIVITY DELETED - STATEMENT ERRORS

A statement error caused the next $EXECUTE, $ACTIVATE or $DEBUG statement within the job to be bypassed. This message follows the statement that is bypassed. If a $JOB statement contains an error, the entire job is bypassed.

SINGLE COPY TASK ALREADY ACTIVE - resubmit job later

ACTIVITY DELETED - nonexistent load module file

ACTIVITY DELETED - password protected load module file

ACTIVITY DELETED - invalid load module file

ACTIVITY DELETED - dispatch queue not available

ACTIVITY DELETED - load module read error

One of the above conditions caused the preceding $EXECUTE, $ACTIVATE, or $DEBUG within the job to be bypassed.

6.8.3 End of Job

$EOJ UNRECOVERABLE I/O ERROR ON SOURCE INPUT device/file
$EOJ END-OF-MEDIUM READ ON SOURCE INPUT device/file

A System Input task encountered the condition on the indicated device or file system input source while spooling the job to the SYC file. The simulated $EOJ statement is written to SYC by the System Input Task upon encountering the abnormal condition. The portion of the job preceding the simulated $EOJ is processed normally. The remainder of the job is lost.

6.8.4 Error in Field

ERROR IN FIELD d message

The preceding statement contains an error. A decimal number is inserted at "d" to indicate the statement field that contains the error. The first field of a statement is "1". "Message" identifies the error and is one of the following:

ILLEGAL DIRECTIVE

ILLEGAL BLANK FIELD

ILLEGAL ENTRY

EXCESSIVE ASSIGNMENTS - More than 64 assignments are specified for a program.

ILLEGAL FILE NAME - The file name in an $ASSIGN1 statement contains a character whose hexadecimal ASCII equivalent is not in the range 21 through 5F.

SELECT LEVELS EXCEEDED - Previous $SELECT statements have established three levels of alternate system input sources which is the maximum.

DEVICE/FILE UNAVAILABLE - The device or file specified on a $SELECT statement does not exist or cannot be allocated. An incorrect password may have been specified for a file.

VALUE MISSING - The entry in a field is incomplete.

6.8.5 Execution Time

JOB EXECUTION TIME=nnHOURS- nnMINUTES- nn.nnSECONDS

This message lists the CPU execution time (excluding I/O wait time) for the batch job.

6.8.6 Records Ignored

RECORD(S) IGNORED

Except for $JOB continuation statements, Job Control ignores any statements that it encounters that do not have a dollar symbol ($) in column 1. These statements are not written on listed output.

6.8.7 SGO Overflow

SGO OVERFLOW

Object records following the $OBJECT statement exceed the size of the SGO file. The size of the SGO file is established at SYSGEN time and on the $JOB statement. The remainder of the job is not processed.

6.8.8 Elapsed Time

TOTAL JOB TIME=nn HOURS- nn MINUTES- nn.nn SECONDS

This message lists elapsed time (from $JOB to $EOJ statement processing) for batch jobs.

6.9 Examples

Examples of various types of batch jobs are provided below:

Example 1

The following example shows a Fortran compilation.

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

Example 2

This example illustrates program assembly from magnetic tape.

```
$JOB EXAMP2 USERB SLOF=LOFILE,USERB    All listed output from job is directed to
                                       user file LOFILE.

$ASSIGN3   SI=M9,SRCE                  Overrides cataloged assignment -
                                       Assembler normally reads source from
                                       SYC file.

$EXECUTE ASSEMBLE
$EOJ
```

Example 3

This example shows a program being cataloged.

$JOB EXAMP3 USERC	Establishes owner name USERC for the job and for files allocated during job.
$USERNAME ABC	Overrides USERC to ABC for file access.
$OBJECT (Object)	Program to be combined with assembler output.
$OPTION 5	Ouput assembled program to SGO file.
$EXECUTE ASSEMBLE (Source) $EXECUTE CATALOG CATALOG FILX U 64 $ASSIGN1 AB=FILAB	
$EXECUTE FILX	Execute the cataloged program.
$EOJ	

Example 4

Here the $SELECT statement is used to obtain a source program.

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

The following are examples of conditional batch processing.

Example 5

If file ABC exists, processing will be suspended through the $DEFNAME statement and FILEMGR will not be executed.

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

Example 6

If the UPDATE run is aborted, ASSEMBLE will not be executed.

```
$JOB EXAMP6 USERG
$ASSIGN1 SI1=SSS
$ASSIGN1 SO=CC1
OPTION 1 2
$EXECUTE UPDATE
(Source Update Directives)
$IFT ABORT NOASSEM
$ASSIGN1 SI=CC1
$EXECUTE ASSEMBLE
$DEFNAME NOASSEM
$EOJ
```

Example 7

SI1 is assigned to CC1 and SO to CC2, replacing the $SETF 1 with $RESETF 1 causes SI1 to be assigned to CC2 and SO to CC1.

```
$JOB EXAMP7 USERH
$SETF 1
$IFF 1 NOT CC1
$ASSIGN1 SI1=CC1
$ASSIGN1 SO=CC1
$DEFNAME NOTCC1
$IFT 1 NOTCC2
$ASSIGN1 SI1=CC2
$ASSIGN1 SO=CC1
$DEFNAME NOTCC2
$EXECUTE UPDATE
(Source Update Directives)
$EOJ
```

7. FILE AND DEVICE ALLOCATION AND I/O

The MPX-32 Allocator obtains the physical resources required to execute a task. The MPX-32 I/O Control System (IOCS) receives and processes device independent I/O requests from both user tasks and the MPX-32 system. This section describes:

Basic I/O linkages for tasks and users

File and device assignments

MPX-32 device addressing (duplicated in Appendix A)

Logical file codes

File Control Blocks

It provides a processing overview that describes wait I/O, no-wait I/O, direct I/O, error processing, and other major I/O concepts in context of the interface between IOCS, standard device handlers, and the system. It describes special system files, how to set up File Control Blocks and Type Control Parameter Blocks, and covers each of the I/O services available to users.

7.1 MPX-32 Logical Input/Output

MPX-32 provides the user with extremely versatile logical, device independent I/O capabilities. That is, instead of coding references to actual physical devices or disc files, and performing I/O himself, the user may code references to LOGICAL FILES and request an MPX-32 Input/Output Control System to perform I/O on his behalf.

Several important advantages are gained by performing logical file I/O:

- The user need not be aware of specific device handling requirements.
- Unprivileged tasks may perform I/O (the SEL 32/75 I/O instructions are part of the privileged instruction set).
- Tasks which perform logical I/O are easier to debug and modify.

In order to provide MPX-32 with sufficient information to create the necessary linkages between the user's logical files and the actual peripheral devices or disc files, the user must:

- Identify logical files via logical file codes (lfc's).
- Describe logical file attributes via File Control Blocks (FCB's).
- Associate logical files with their target physical devices or disc files via logical file code assignments.

7.1.1 Logical File Codes

Logical file codes (lfc's) are user defined one to three ASCII character codes which identify logical files within tasks.

Logical file codes are configured into corresponding File Control Blocks (see Section 7.6 for FCB format).

7.1.2 File Control Blocks

A File Control Block (FCB) must be set up by the user to describe each logical file within a task, and to describe certain attributes of each logical I/O operation.

In addition, certain information collected by IOCS following each I/O operation is made available to the user via the corresponding FCB.

Finally, certain space in the FCB is reserved for use by IOCS.

The FCB format is decribed in detail in Section 7.6.

7.1.2.1 Logical I/O Initiation

In order to initiate a logical I/O operation the user must code into his task a call to one of the data transfer or device access services described in Section 7.8 accompanied by the address of a corresponding FCB.

7.1.3 Logical File Code Assignment

Before executing a logical I/O request the user must associate the appropriate logical file code (lfc) with the target peripheral device or disc file. This is accomplished via logical file code assignment.

Logical file codes may be assigned to specific peripheral devices or files when a task is cataloged (static assignment), or during task execution via the M.ALOC service (dynamic assignment).

For tasks which run under TSM (interactive tasks) or Job Control (batch tasks) static assignments may also be made by the user at run time (see Section 5 and 6). For such assignments, if the logical file code matches one assigned at catalog time, it replaces the cataloged assignment. If the file code assigned at run time does not match any cataloged assignment, it is added to cataloged assignments.

Dynamic assignments cannot override cataloged or run time assignments, and any attempt to do so is treated as an error. To accomplish dynamic override, the user task must first deallocate (deassign) the static assignment via the M.DALC service.

7.1.3.1 Making Assignments

MPX-32 supports four classes of assignments as follows:

ASSIGN1 - Assigns a lfc to a permanent disc file.
ASSIGN2 - Assigns a lfc to one of four types of special system disc files (SYC, SGO, SLO, SBO).
ASSIGN3 - Assigns a lfc to a peripheral device or temporary disc file.
ASSIGN4 - Assigns a lfc to a previously assigned lfc.

For a description of Cataloger assignment directives, see Volume 2, Section 2.

For a description of static assignments which may be made at run-time via TSM and Job Control, see Sections 5 and 6.

For a description of the dynamic assignment/allocation services (M.ALOC and M.DALC), see Section 7.8.

7.1.3.2 I/O Linkages

As a result of building a logical file code into an FCB, and assigning the logical file code to a device or disc file, the following I/O linkage elements are created in the user's TSA:

- A File Assignment Table (FAT) entry which points to an entry in each of two tables describing controllers (CDT) and devices (UDT).

- A File Pointer Table (FPT) entry which contains the lfc specified during assignment, and which should therefore match the lfc contained in the FCB.

Before any service requested by the user to initiate an I/O operation can be completed, the user's logical file must be opened. The user may perform this function himself via the M.FILE service, otherwise IOCS will automatically open the users logical file when an I/O initiation service request is made, and the logical file is not open.

When the user's logical file has been opened, the I/O linkage elements are tied together as follows:

- The FPT entry points to the corresponding FCB and FAT entry.

- The FAT and corresponding FCB entries point to each other.

7.2 MPX-32 File Access

The MPX-32 File System Manager (FISE) maintains the System Master Directory (SMD), which describes the disc, starting block, length, user name, indicators (e.g., password protection) and other information for each permanent file on the system. The description of temporary files is similar, but contained in the Task Service Area (TSA) for each task.

When a file is assigned for I/O, the MPX-32 Allocator calls FISE. FISE checks to ensure that the file exists, that a password has been supplied, if applicable, that the file belongs to the user who activates the task or is a system file, etc., and passes information back to the Allocator so that all such access is verified before the file is opened by the task via IOCS. The description of a valid file is placed in the File Assignment Table (FAT); if an invalid assignment is made, the error is passed back to the user in the allocation stage via an abort code and the task is not continued.

IOCS uses the FAT entry to map to the appropriate disc and file. Access to disc files is sequential, starting with the first block in the file, unless the user has provided a random access block address in Word 2 bits 12-31 or Word 10 bits 0-31 of the FCB.

Permanent files are accessed by name; temporary files are accessed by a device address as described in Section 7.4.

7.2.1 File Management

The MPX-32 File System Executive (FISE) provides space management for all discs defined to the system at SYSGEN. Disc space is allocated to files in units, each consisting of an integral number of 192-word blocks. In no case is an allocation unit larger than four blocks (see Table 7-1).

Table 7-1 Disc Description

Unit Size	Type	Words/ Sector	Sectors/ Track	Number of Heads	Max Cylinders	Sectors/ Cylinder	Max Sectors	Sectors/ Allocation	Max Alloc. Units	Bit Map Size*	Max Byte Capacity
4MB	Fixed Head	192W	23	256	0	0	5888	1	5888	184W	4.52MB
40MB	Moving Head	192W	23	5	400	115	46,000	2	23,000	719W	35.32MB
80MB	Moving Head	192W	23	5	800	115	92,000	2	46,000	1438W	70.65MB
300MB	Moving Head	192W	23	19	800	437	349,600	4	87,400	2732W	268.49MB
Extended I/O Devices											
24MB	Fixed Media	192W	20	4	300	80	24,000	2	12,000	375W	18.43MB
40MB	Moving Head	192W	20	5	400	100	40,000	2	20,000	625W	30.72MB
80MB	Moving Head	192W	20	5	800	100	80,000	2	40,000	1250W	61.44MB
300MB	Moving Head	192W	20	19	800	380	304,000	4	76,000	2375W	233.47MB

7.2.1.1 Temporary versus Permanent Files

Temporary files are allocated to a task for use during its execution and are deallocated when the task terminates. Tasks are not allowed to share a temporary file, but any number of tasks can have several temporary files allocated to them. Definitions of temporary files are retained in the Task Service Area (TSA) for a task.

Permanent files are files which are permanently defined to the system. The definition of each permanent file and each memory partition is retained in the System Master Directory (SMD) which resides on disc. An SMD entry is added each time a new permanent file or memory partition is created and the entry is retained until the file or memory partition is deleted.

7.2.1.2 System versus User Files

Permanent files defined in the SMD are divided into two levels: system files and user files. System files are required for MPX-32 functions or are shared between users. All files created at SYSGEN are system files. User files are private and owned by individual users. File identities take the following form:

System Files:	filename
User Files:	username, filename

System files are identified by a one- to eight-character file name which may not duplicate another system file name. User files are identified by a one- to eight-character user name together with a one- to eight-character file name. The "username, filename" pair identifying a user file may not duplicate the "username, filename" identity of another user file.

A file's category (system or user) is established when the FILEMGR utility, the M.CREATE system service, the Editor, or any similar utility, is used to create the file. A file created by any method may be specified as a system file. If not explicitly specified as a system file, a file's category is based on whether or not a user name is associated with the task creating the file.

The user name associated by default when a user logs on the MPX-32 system is the logon owner name. There is no user name for files created or accessed by a user who logs on the OPCOM console (i.e., system files are assumed).

A username associated with files related to a task can be modified or changed to nothing via the USERNAME Catalog, File Manager, and Job control directives, the TSM USERNAME command, and the M.USER system service. Association of a user name with a task must be consistent with the allocation of permanent files by the task, i.e., if user files are to be allocated by a task, the user name must be associated with the task prior to allocating the files.

When permanent file allocation is attempted, the file directory search is based on whether or not a user name is associated with the task. If a user name is not present, a system file with the specified file name must exist in the directory. If a user name is present, the directory is searched for a user file with the specified file name and user name. If the user file does not exist, the directory is searched for a system file with the specified file name.

7.2.1.3 Password and Key Protection

File protection mechanisms are available to prevent unauthorized access to and deletion of permanent files. System and user files may be protected individually. User files may also be protected on a per user basis.

Individual files can be protected when the files are created. Protection is based on a one- to- eight character password; the password may or may not be unique for each file as desired. If a password is associated with the file, the file is deleted only by specifying the password. The password must also be entered on the SAVE FILE, DELETE, and EXPAND File Manager directives.

In addition, read and write access to a file can be restricted. A file can be declared as read-only. A read-only file can be read without its password, but the password is required to write the file. A password-only file requires the password to read or write the file.

The KEY utility is used to provide a file (named M.KEY) containing valid owner names/user names for an installation. A key can be specified for an owner name/user name, if desired. The key is then required to:

log on the system

change to this user name for access to associated user files

The M.KEY mechanism provides file access control on a 'per-username' basis.

7.2.1.4 System Master Directory (SMD)

Each permanent file, temporary file, or memory partition has associated with it a two word space definition which describes its location, length, etc. The space definition of permanent file or memory partition resides on the System Master Directory (SMD). The space definition together with the permanent file name, user name and password, or memory partition name, comprise an SMD entry.

The space definition of a temporary file is retained in the File Assignment Table (FAT) area of the TSA of the task that assigns the file until the task terminates or system output is complete. At that time, these definitions are returned to the File System Executive for deallocation of the defined space. Deallocation of permanent file space is performed via an M.DELETE service call, a DELETE command in the Editor, or when the File Manager utility (FILEMGR) encounters a DELETE directive.

Memory partitions, similar to permanent disc files, have a two word space definition in the associated SMD entry. The SMD entry for a statically allocated memory partition is built by SYSGEN when a PARTITION directive is specified. This type of partition can be deleted only by omitting the definition in a subsequent SYSGEN warm or cold start. The SMD entry for a dynamically allocated memory partition is built by the FILEMGR utility in response to a CREATEM directive. This type partition is deleted by the FILEMGR utility upon encountering a DELETE directive, or alternatively via M.DELETE.

7.2.1.5 System Master Directory (SMD) Entries

Figure 7-2 describes the formats of SMD entries for files and memory partitions.

Disc File SMD Entry

WORD	0 7	8 15	16 31
0 1	FILE NAME (MAX 8 CHARACTERS, LEFT-JUSTIFIED, BLANK-FILLED)		
2	FILE TYPE*	STARTING DISC ADDRESS (STARTING BLOCK NUMBER)	
3	FILE INDICATORS**	LENGTH IN 192-WORD BLOCKS	
4 5	USER NAME OR 0 FOR SYSTEM FILE (MAX 8 CHARACTERS, LEFT-JUSTIFIED, BLANK-FILLED)		
6	COMPRESSED PASSWORD		UDT INDEX
7	RESERVED		

Memory Partition SMD Entry

WORD	0 7	8 15	16 31
0 1	PARTITION NAME (GLOBALnn or DATAPOOL)		
2	STARTING LOGICAL PAGE		STARTING PHYSICAL PAGE OR 0
3	FILE INDICATORS**	MEMORY CLASS	LENGTH IN PAGES
4 5	RESERVED		
6	COMPRESSED PASSWORD		RESERVED
7	RESERVED		

* File Types (Hexadecimal digits)

ED -	EDITOR SAVE
EE -	EDITOR STORE
FE -	EDITOR Workfile
FF -	SYSGEN-created
BA -	BASIC (unused)
CA -	Cataloged Load Module
01 - 99	Available for Customer Use

**File Indicators

Bit		
	0 -	Active, permanent
	1 -	SYSGEN'd partition
	2 -	No save in response to FILEMGR SAVE directive
	3 -	Fast
	4 -	Collision Mapping
	5 -	Not SYSGEN'd partition
	6 -	Read-Only Protected
	7 -	Read/Write Protected

820616

Figure 7-2. SMD Entries

7.3 MPX-32 Device Access

When a device is assigned for I/O, the MPX-32 Allocator verifies that the device is available, allocates blocking buffers required for blocked I/O to disc or magnetic tape, and identifies the device for IOCS in the File Assignment Table (FAT). If a device is not available (e.g., not included in the SYSGEN configuration of a system, offline, etc.), the Allocator returns an abort code and does not continue with the task.

Throughout the reference manual, the generic descriptor 'devmnc' is used to indicate that a device can be specified.

Under MPX-32, device addresses are specified using a combination of three levels of identification. They are device type, device channel/controller address, and device address/subaddress.

A device can be specified using the generic device type only, which will result in allocation of the first available device of the type requested.

A second method of device specification is achieved by using the generic device type and specifying the channel/controller address. This results in allocation of the first available device of the type requested on the specified channel or controller.

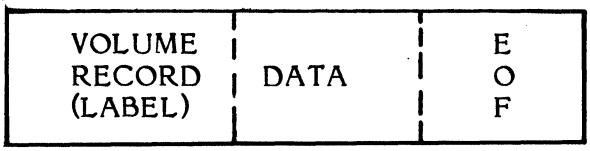
The third method of device selection requires specification of the device type, channel/controller, and device address/subaddress. This method allows specification of a particular device.

7.3.1 Special Device Specifications and Handling

7.3.1.1 Magnetic Tape

Multivolume magnetic tape processing is supported under MPX-32. A multivolume magnetic tape is defined as a set of one or more (255 maximum) physical reels of magnetic tape processed as a continuous reel.

Multivolume tape files have the following general format:



The 192-word volume record is a tape label produced as the first record for each volume of a multivolume file. The volume number and reel identifier portions of this record are verified as part of subsequent read operations. The volume record has the following format:

WORDS(S)	CONTENTS
0	Reel Identifier (ASCII)
1	Volume Number (Binary)
2-3	Date Created (ASCII)
4	Time Created (ASCII)
5-191	Unused

The reel identification is a four-character ASCII entry.

The volume number (1-255) is given in binary.

The date is Gregorian in the format.

MM/DD/YY

Time is given in the following format:

HOUR (Byte 0)
MINUTE (Byte 1)
SECOND (Byte 2)
INTS (1/100 Seconds) (Byte 3)

Multivolume magnetic tape processing mode is invoked via the Job Control statement, $ASSIGN3, and/or the Allocate File/Peripheral Device Monitor Service.

Multivolume processing is automatic for magnetic tape operations in the forward direction (i.e., READ, WRITE, ADVANCE, ERASE). For reverse direction operations (i.e., REWIND, BACKSPACE), the user is required to provide processing logic for proper reel manipulation and positioning requests.

Volume numbers in the range 1-255 are provided as an operations aid in mounting and dismounting physical reels of magnetic tape. MPX-32 treats the volume numbers in circular fashion, i.e., volume 1 follows volume 255 in the numbering scheme.

A scratch tape (SCRA) may not be assigned if the multivolume mode is specified. A unit for which multivolume magnetic tape operations are applicable may not be designated a shared (SHR) device. The mount message cannot be inhibited for multivolume magnetic tape operations.

For multivolume magnetic tape processing, MPX-32 will not pass End-of-Medium (EOM) indicators to the user via the FCB. If a READ, WRITE, or ADVANCE operation is requested when tape is at end-of-medium, the system will issue a DISMOUNT message to the console teletypewriter for the current volume (reel), followed by a MOUNT request for the next sequential volume number prior to performing the requested operation.

The system does not recognize multivolume mode specification when a magnetic tape is positioned at load point and a REWIND or BACKSPACE operation is requested.

For File Manager RESTORE operations, special processing occurs if the file being restored resides on two or more reels of a multivolume magnetic tape.

For magnetic tape, a reel identifier, multivolume number, and unblocking can be part of the device mnemonic.

Syntax:

lfc= device ,[reel] ,[volume] [,U]

where:

device — is any one of the three levels of device specification described above.

reel — specifies a one- to four-character identifier for the reel. Default: SCRA (Scratch).

volume — if multivolume tape, indicates volume number. Default: not multivolume (0).

U — the tape is optionally unblocked. Default: blocked.

Commas in this specification are significant. If an option is not specified, e.g., a reel identifier, but another option is specified, e.g., U, commas must be inserted for all non-specified options in between, e.g.,

```
MT1000,,,U
```

There must be no embedded blanks within the entire device mnemonic.

When the task is activated that has an assignment to tape, a MOUNT message indicates the name of the task and other information on the OPCOM console:

```
{TASK }
{jobno} taskname,taskno MOUNT reel VOL volume ON devmnc  DEV,R,A,H? devmnc
```

where:

jobno — if the task is part of a batch job, identifies the job by job number.

taskname — is the name of the task to which the tape is assigned.

taskno — is the number of the task.

reel — if the assignment is a multivolume tape, indicates the reel identifier specified in the assignment. SCRA is invalid if using multivolume tape.

volume — identifies the volume number to mount if multivolume tape.

devmnc — is the device mnemonic for the tape unit selected in response to the assignment. If a specific channel and subaddress are supplied in the assignment, the specific tape drive is selected and named in the message. Otherwise, a unit is selected by the system and its complete address is named in the message.

DEV,R,A,H — the device listed in the message can be allocated and the task resumed (R), a different device can be selected (DEV), the task can be aborted (A), or the task can be held with the specified device deallocated (H). If a 'R' response is given and a high speed XIO tape drive is being used, its density can be changed when the software select feature is enabled on the tape unit front panel. If specified, it will override any specification made at assignment. Values are:

N or 800	indicates 800 bpi nonreturn to zero inverted (NRZI)
P or 1600	indicates 1600 bpi phase encoded (PE)
G or 6250	indicates 6250 bpi group coded recording (GCR) Default.

Example usage: RN R1600, etc.

Note: Do not insert blanks or commas.

Response:

To indicate the drive specified in the MOUNT message is ready and proceed with the task, mount the tape on the drive and type R (Resume), optionally followed by a density specification if the drive is a high speed XIO tape unit. To abort the task, type A (Abort). To hold the task and deallocate the specified device, type H (Hold). The task can then be resumed by the OPCOM CONTINUE command, at which time a tape drive will be selected by the system and the MOUNT message redisplayed.

To select a tape drive other than the drive specified in the message, enter the mnemonic of the drive you want to use. Any of the three levels of device identification can be used. The MOUNT message is reissued. Mount the tape and type R if satisfactory, or if not satisfactory, abort, override, or hold as just described.

7.3.1.2 Temporary Disc File Size

For a temporary disc file, size must be specified and unblocking is optional.

Syntax:

lfc = device,size[,U]

where:

device	is any one of the three levels of device specification described above.
size	specifies the number of 192-word blocks required.
U	the file is optionally unblocked. Default: blocked.

7.3.2 Examples of Device Identification Levels

Examples of the three methods of device specification follow:

Type 1 - Generic Device Class

$ASSIGN3 DEV=M9,SAVE, 1

In this example, the device assigned to logical file code (lfc) "DEV" will be any 9-track tape unit on any channel. The multivolume reel number is 1. The reel identifier is SAVE.

Type 2 - Generic Device Class and Channel/Controller

```
$ASSIGN3 DEV=M910,MORK,,U
```

In this example, the device assigned to logical file code (lfc) "DEV" will be the first available 9-track tape unit on channel 10. The specification is invalid if a 9-track tape unit does not exist on the channel. The reel identifier is supplied. This is not a multivolume tape. It is, however, unblocked.

Type 3 - Specific Device Request

```
$ASSIGN3 DEV=M91001
```

In this example, the device assigned to logical file code (lfc) "DEV" will be the 9-track tape unit 01 on channel 10. The specification is invalid if unit 01 on channel 10 is not a 9-track tape. The tape reel identifier is SCRA; the tape is blocked and is not multivolume.

7.3.3 GPMC Devices

GPMC/GPDC device specifications are in keeping with the general structure just described. For instance, the terminal at subaddress 04 on GPMC 01 whose channel address is 20 would be identified as follows:

```
$ASSIGN3 DEV=TY2004
```

7.3.4 NULL Device

A special device type "NU" is available for NULL device specifications. Files accessed using this device type generate an end-of-file (EOF) upon attempt to read and normal completion upon attempt to write.

7.3.5 OPCOM Console

Logical file codes are assigned to the OPCOM console by using the device type "CT".

7.3.6 Special System Files

There are four special mnemonics provided for access to special system files: SLO, SBO, SGO and SYC. These are assigned via the $ASSIGN2 statement, as in:

```
$ASSIGN2 OUT=SLO,printlines
```

For non-batch tasks, SLO and SBO files are allocated dynamically by the system and used to disc buffer output to a device selected automatically. For batch tasks, use of SLO and SBO files is identical, except that automatic selection of a device can be overridden by assigning a specific file or device.

SGO and SYC assignments are used for batch processing. See Section 7.5.

Dev Type Code	Device	Device Description
00	CT	Operator Console (Not Assignable)
01	DC	Any Disc Unit
02	DM	Any Moving Head Disc
03	DF	Any Fixed Head Disc
04	MT	Any Magnetic Tape Unit
05	M9	Any 9-Track Magnetic Tape Unit
06	M7	Any 7-Track Magnetic Tape Unit
07	CD	Any Card Reader-Punch
08	CR	Any Card Reader
09	CP	Any Card Punch
0A	LP	Any Line Printer
0B	PT	Any Paper Tape Reader-Punch
0C	TY	Any Teletypewriter (Other than Console)
0D	CT	Operator Console (Assignable)
0E	FL	Floppy Disc
0F	NU	Null Device
10	CA	Communications Adapter (Binary Synchronous/Asynchronous)
11	U0	Available for user-defined applications
12	U1	Available for user-defined applications
13	U2	Available for user-defined applications
14	U3	Available for user-defined applications
15	U4	Available for user-defined applications
16	U5	Available for user-defined applications
17	U6	Available for user-defined applications
18	U7	Available for user-defined applications
19	U8	Available for user-defined applications
1A	U9	Available for user-defined applications
1B	LF	Line Printer/Floppy Controller (used only with SYSGEN)

Table 7-2. MPX-32 Device Type Codes

7.3.7 Floppy Discs

Floppy discs may be treated as random access devices or as sequential access devices. When a floppy disc is assigned, all space on the diskette (1334 192-word blocks) is allocated to the task. The MPX-32 file system does not support floppy discs.

The floppy disc provides a software End-of-File (EOF) capability for the unblocked, sequential mode of operation. A request to write an EOF causes the floppy to write a 192-word record beginning with X'0F00'. When reading, EOF will be reported when X'0F00' is detected as the first word of data read. EOF reporting may be optionally inhibited. EOF reporting is automatically inhibited when reading in random access mode. Only blocked EOF is reported when reading in blocked mode.

See Tables 7-5 and 7-7 for a complete description of floppy disc operations and specifications.

7.3.7.1 Generating Floppy Discs as Discs

To SYSGEN a floppy disc as a disc, the following device directive format should be used:

```
DEVICE=F0, DTC=FL, HANDLER=H.FLIOP, DISC=FL001
```

The corresponding assignment would be made as follows:

```
$ASSIGN3  LFC=FL7EF0
```

When SYSGENed as file type code "FL", all I/O is assumed to be unblocked unless bit 5 of word 2 (BL) is set prior to execution of the required operation.

7.3.7.2 Generating Floppy Discs as Magnetic Tapes

To SYSGEN a floppy disc as a magnetic tape, the following device directive format should be used:

```
DEVICE=F0, DTC=M9, HANDLER=H.FLIOP, DISC=FL001
```

The corresponding assignment would be made as follows:

```
$ASSIGN3  LFC=M97EF0, TAPE[,,U]
```

When SYSGENed as file type code "M9", all I/O is assumed to be blocked unless "U" is specified on the assignment. Standard magnetic tape mount messages will be sent to the console.

Multi-volume operation is, however, not supported for floppy discs SYSGENed as magnetic tape.

7.3.8 Samples

A description of device selection possibilities would be constructed as follows:

DISC

DC	Any Disc
DM	Any Moving Head Disc
DM08	Any Moving Head Disc on Channel 08
DM0801	Moving Head Disc 01 on Channel 08
DF	Any Fixed Head Disc
DF04	Any Fixed Head Disc on Channel 04
DF0401	Fixed Head Disc 01 on Channel 04

TAPE

MT	Any Magnetic Tape
M9	Any 9-track Magnetic Tape
M910	Any 9-track Magnetic Tape on Channel 10
M91002	9-track Magnetic Tape 02 on Channel 10
M7	Any 7-track Magnetic Tape
M712	Any 7-track Magnetic Tape on Channel 12
M71201	7-track Magnetic Tape 01 on Channel 12

CARD EQUIPMENT

CD	Any Card Reader-Punch
CR	Any CR
CR78	Any CR on Channel 78
CR7800	CR on Channel 78 Subaddress 00
CP	Any CP
CP7C	Any CP on Channel 7C
CP7C00	CP on Channel 78 Subaddress 00

LINE PRINTER

LP	Any LP
LP7A	Any LP on Channel 7A
LP7A00	LP on Channel 7A Subaddress 00

7.4 I/O Processing Overview

A task starts I/O operations (opens, reads, writes, etc.) by issuing service calls to IOCS (see Section 7.9). IOCS completes linkage of an assigned device or file and an FCB, validates the logical address of the task's data buffer (defined in the Transfer Control Word of the FCB), and links an I/O request containing the TCW information to a queue for the appropriate device handler. The I/O requests are queued by the software priority (1-64) of requesting tasks.

The handler issues appropriate instructions to the device controller (Command Device, Test Device, or Start I/O).

The controller performs the I/O. For example, it reads a record into the task's data buffer. When the requested I/O is complete, the controller issues a Service Interrupt (SI).

If the handler is passing the address of a list of commands or data (IOCL) for a controller to operate on, the SI is returned from the Controller when all operations specified in the list have been completed.

The processing that occurs when the handler receives the SI interrupt from the controller depends on whether the user has established a wait-I/O or no-wait I/O environment via the FCB.

7.4.1 Wait I/O

If wait I/O, the task has been waiting (suspended) until the I/O operation completes. IOCS returns to the task at the point following the I/O service call.

7.4.1.1 Wait I/O Errors

If an error occurs during an I/O operation to tape or disc, the handler automatically retries the operation. If retry operations fail, the handler passes the error to IOCS and IOCS posts status in the FCB. (Word 3, and optionally Words 11 or 12.) The task can take action or not as described in the next section. When an error is detected on a card reader, line printer, or other device that does not have automatic retry, the handler passes the error to IOCS and IOCS issues a message on the OPCOM console indicating the device is not operating:

```
*devmnc INOP: R,A?
```

Sample criteria for the INOP message are: the printer runs out of paper or jams, a card reader jams, etc. IOCS allows the console operator to correct the condition (or abort). If the device is fixed, the operator types R (Retry). IOCS re-establishes the entry conditions for the handler (passes the TCW with the initial transfer count, etc.) and calls the handler to retry the I/O operation that was in error. The error status is cleared and I/O proceeds normally. If the operation aborts, IOCS starts abort processing.

7.4.1.2 Wait I/O Exit and Abort Processing

If error processing is not applicable (disc or magnetic tape) or if the operator has responded A (Abort) to the INOP message, IOCS either aborts the task or transfers control to the task at the address specified in Word 6 of the FCB. If the task gains control, it can examine the contents of the Word 3 and optionally Words 11 and 12 as applicable and perform its own exit or abort processing.

If the user has not defined an error return address for wait I/O in the FCB, IOCS aborts the task and displays an abort message on the OPCOM console:

I/O ERR DEVICE: devmnc STATUS statusword

IOCS displays the status word 3 from the FCB on the OPCOM console.

7.4.1.3 Error Processing and Status Inhibit

The user can set Word 2 bit 1 of the FCB to bypass error processing by handlers and by IOCS. On error, the status word of the FCB is still set by IOCS (unless bit 3 is set as described below), and IOCS transfers control to the task normally. The task must perform any error processing.

If the user sets Word 2 bit 3 of the FCB, he inhibits handlers from checking status in any respect. No error status is returned to IOCS. All I/O appears to complete without error.

7.4.2 No-Wait I/O

If the user has indicated no-wait I/O in the FCB, IOCS returns control to the task immediately after an I/O request is queued. The task continues in execution in parallel with I/O. When the handler fields an SI interrupt for the specified I/O operation, it notifies the MPX-32 Executive. The Executive links the I/O queue entry to a software interrupt list for the task (a task interrupt). When the task is to gain control, the Executive passes control to IOCS.

7.4.2.1 No-Wait I/O Complete Without Errors

IOCS checks the address specified by the user in the FCB for no-wait I/O end action processing. If I/O has completed successfully, it routes control to the address provided by the user for normal end processing (word 13). (If the user has not specified this address in the FCB, IOCS returns control to the Executive, and the task continues in execution at the point where the task interrupt occurred.)

7.4.2.2 No-Wait I/O Complete With Errors

When IOCS gains control on a task interrupt and an error has occurred, IOCS routes control to the task at the user-supplied error return address in the FCB (Word 14.) If the user has not supplied this address, control is returned to the Executive and then to the task so that it continues in execution at the point where the task interrupt occurred.

The FCB (Word 3 and optionally Words 11 and 12) will indicate the cause of an error. The task is responsible for examining the word(s) and for any recovery procedure. IOCS makes no attempt to recover from an error condition through operator intervention when a task uses no-wait I/O.

7.4.2.3 No-Wait End Action Return to IOCS

A task using no-wait I/O end action processing should return to IOCS via an SVC 1,X'2C' after normal or error processing is complete. IOCS will return control to the executive, which returns control to the task at the point where the task interrupt occurred.

7.4.3 Direct I/O

Within a task, the user can temporarily bypass normal IOCS and handler functions by coding his own handler and attaching it to a specific channel (service interrupt level). Direct I/O is totally under the user's control and is used to acquire data at rates which prohibit IOCS overhead.

To perform direct I/O, the task basically passes a TCW directly to a device. It bypasses IOCS completely. The task is responsible for any control structures related to the I/O operation (it does not for example, have use of an FCB) and it must field interrupts on its own. The user must be familiar with the hardware, its response to software instructions, and must implement all support pertaining to I/O to the device.

Before connecting his own handler, the user issues a Reserve Channel call to IOCS. IOCS then holds all outstanding or subsequent requests for the specified channel until the task issues a Release Channel call. When IOCS receives the release request, it resumes normal processing on the channel with the standard handler.

Direct I/O is not the same function a user fulfills by developing an I/O handler and linking it into the system at SYSGEN. Direct I/O is a privileged operation.

7.4.4 Blocking

I/O to disc files and magnetic tape is blocked or not depending upon the device assignment or the setting of Word 2 bit 5 of the FCB. If the user does not set bit 5, I/O will be blocked or unblocked as determined by the device assignment. If bit 5 is set, I/O will be blocked. The FCB definition will override any assignment specification of unblocked (specified explicitly on an ASSIGN1 or ASSIGN3 directive or M.ALOC service call).

For blocked I/O, IOCS automatically allocates and uses a 192-word blocking buffer that provides intermediate buffering between a task's data buffer and a device. A block is 192 words long and records that can be packed into the block are a maximum 254 bytes long. Longer records are simply truncated.

On input, IOCS transfers a block of records from a device into the blocking buffer and moves them into the task's data buffer one logical record at a time. On output, IOCS transfers one logical record at a time from the task's data buffer into the blocking buffer and outputs the accumulated records in 192-word blocks.

Reads and writes to the same blocked file or tape by the same task should not be mixed because it interferes with the blocking buffer operations being performed by IOCS. A read attempted while IOCS is writing out a group of records to the blocking buffer (or a write when reading) is not executed and IOCS aborts the task.

7.5 I/O Via Special System Files

There are two types of temporary system files that a task or user can assign for I/O:

System Listed Output (SLO) files are used to disc buffer output for printing.

System Binary Output (SBO) files are used to disc buffer output in card format.

There are two types of temporary system files used by the MPX-32 Job Control processor and by TSM to disc buffer I/O. These files are accessed by user tasks, but such access is controlled by the system to facilitate processing.

System Control (SYC) files are used to control command processing. For batch, one SYC file is allocated, for each job submitted. The SYC file contains commands required to start, execute, and complete the job. TSM treats the user's terminal and any command files selected as an SYC file.

System General Object (SGO) files are blocked files which are passed from one job step to another without implicit end-of-file marks. Typically SGO files are used to accumulate object code from an assembly or compilation and move it through cataloging or to a library.

7.5.1 System Listed Output Files

System Listed Output (SLO) files are temporary disc files used to buffer output for printing. If assigned for output from a non-batch task, contents of the SLO file are accumulated on disc and output automatically to one of the device(s) established at SYSGEN or via OPCOM for automatic selection. Output begins only after the task deallocates the SLO file via an M.DALC service call or after the system deallocates the file during task exit or abort termination processing.

If SLO is assigned for output from a task running in the batchstream, it is output after the job is complete. The automatic selection of a device can be overridden in batch by directing SLO to a specific device or permanent file on a $JOB statement.

Tasks are allowed to READ from, WRITE to, REWIND, or WRITE an EOF to an SLO file. Writing an EOF causes IOCS to purge an SLO file that is output active. (See M.WEOF.)

An SLO file is automatically blocked and deblocked by IOCS. (See Section 7.4.4.) The maximum record size on an SLO file is 132 bytes.

With SLO files, a task can optionally fill up to three title buffers to be output at the top of each page when the contents are printed. The title buffers, which are 132 bytes each, are filled sequentially whenever a set of 1, 2 or 3 records specify minus (-) as a carriage control character.

The number of title buffers to output is initialized for each set of records. For example, if three buffers are filled and later a minus (-) carriage control character is encountered in a record, only the first buffer (the most recent record) is printed on subsequent pages. Title record(s) continue to be printed until they are replaced by another set of one or more title records or until the current SLO file has been printed.

7.5.2 System Binary Output Files

A System Binary Output (SBO) file is a type of temporary file used to disc buffer output in card image format. When SBO output is directed to a card punch, the output is punched in either ASCII or binary format as determined by the first byte of the record. A binary record is a maximum 120 bytes long (1.5 bytes per card column) and an ASCII record is a maximum 80 bytes long (1 byte per card column).

In other respects, handling of SBO files is identical to SLO files described previously.

Title buffers do not apply.

7.5.3 System General Object Files

SGO is a permanent file used to accumulate object code. The SGO file exists until a job is complete, at which time it is deleted. User tasks designed to run in batch or TSM can do I/O to the SGO file. Legal operations are: OPEN, CLOSE, READ and WRITE. REWIND and WRITE EOF are allowed on input.

Object code is collected sequentially on the SGO file. The open operation performed by the system when SGO is assigned does not set the current access address back to the base of the file. The file is positioned at the point the previous processor stopped accessing the file. This allows the accumulation of output from a compiler or assembler. In batch, object records following the $OBJECT statement on a job file can be accumulated on the SGO file for a job. In TSM, the $OBJECT capability is not available.

A task which writes data to the SGO file should not write an end-of-file, or perform a rewind operation on the file to preclude destroying the contents. To read the SGO file, a task should open the file in the read-only mode, write an end-of-file, then rewind the file before beginning the read operation. If the SGO file is opened read-only, its contents are set to null when the file is closed.

7.5.4 System Control Files

SYC is a type of temporary system file associated only with jobs processed in the batchstream, one SYC per job. SYC is used for buffering input from the devices and/or files. Tasks that are not designed to run in the batchstream or under TSM should not make assignments to SYC. TSM makes a special default to SYC for on-line users accessing batch tasks:

SYC = UT

where 'UT' has been assigned by TSM to the user's terminal.

An SYC file is created automatically for the user when the BATCH command is issued. The specified or implied file or device contents are copied by the system to SYC.

Records having a dollar sign, "$", as the first character are assumed to be Job Control statements. If a batch task attempts to read a Job Control statement, an end-of-file status is returned in the appropriate FCB of the task. A task running in batch is aborted on a second attempt to read a Job Control statement.

A Job Control $EXECUTE or $DEBUG statement activates a task in batch. Records which follow an $EXECUTE statement are read by the batch task specified on the $EXECUTE statement.

A Job Control $OBJECT statement causes subsequent records to be put on the SGO file.

Valid operations on SYC by user tasks include: OPEN, READ and CLOSE. Access is solely sequential; reads are destructive.

7.6 Setting up File Control Blocks

Section 7.1 describes the function of an FCB. Parts of the FCB must be defined by the user:

A logical file code is required.

A Transfer Control Word (TCW) indicating transfer count and data buffer address for I/O operations controlled by this FCB is required.

IOCS assumes the following if no other special I/O characteristics are defined in the FCB:

Wait I/O - IOCS returns to the calling task only when a requested operation on the file or device assigned to this FCB is complete.

Automatic retry on error by IOCS and some handlers as described in Section 7.4.1.

Device dependent output and input are handled using standard techniques.

Status information is returned in the FCB.

File and device access is sequential.

Areas of the FCB can be used to define:

No-Wait I/O - Immediate return to the calling task after I/O operation is queued. User can define address to return to in task when processing is complete (normal or error).

Error Processing Inhibit - Only status is returned by handlers. No error processing by IOCS or handlers.

Special device output characteristics.

Random access for disc files.

Expanded Transfer Control Word (TCW)

Some areas of the FCB are defined by IOCS. IOCS stores the opcode each time the task specifies a particular FCB, stores status returned by handlers, tracks actual record length in bytes for each transfer, and builds and maintains I/O queue and File Assignment Table (FAT) addresses. Figure 7-3 describes the FCB layout. In the diagram, areas which can be defined by the user are shaded. All but Words 0 and 1 are optional. The user should initialize to zero all portions of the FCB that he wants to let IOCS or handlers set up or that IOCS must handle.

Overlay Considerations

When programs are cataloged as overlays, care must be taken to preserve the FCBs used in the course of the program until the files can be closed and deallocated. One way to ensure this is to construct the FCBs within the main program where they will not be destroyed by the loading of overlays. This will require that the appropriate address labels and externals be set up so that overlay segments which do I/O may connect with the FCBs. If the FCBs are contained within overlay segments, then the files associated with the FCBs must be closed and deallocated before the segment is overlayed by another segment.

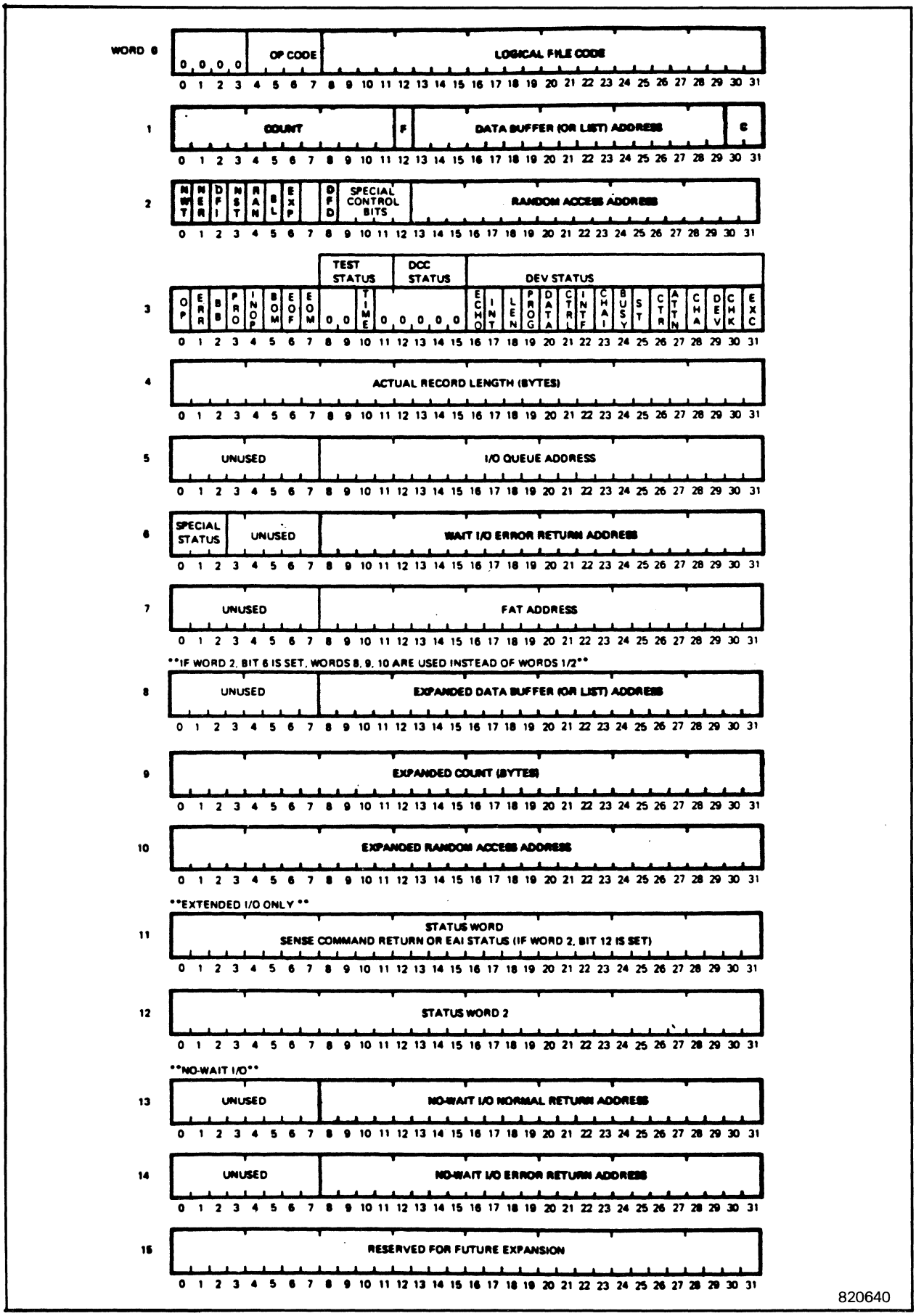


Figure 7-3.File Control Block

Word/Bit	Descriptor	Description
		Buffer Address Definitions Set by User
1/12	F	Format: 1=byte, 0=other
1/30,31	C	Code: If Format = 1: byte number. If Format = 0: 00=word, 01=left halfword, 11=right halfword
		Control Flags - Special Format Indicators Set by User
2/0	NWT*	No-Wait I/O (Words 13-14). Else: Wait I/O
2/1	NER	No error return processing. Else: see Section 7.6.1.3.
2/2	DFI*	Data Format Inhibit. Use format in Bits 8-12. Default: IOCS provides formats. See Table 7-7.
2/3	NST	No Handler Status checked. Handler status normally returned in Word 3 will not be returned, however, system status will be returned. Else: see Section 7.6.1.3.
2/4	RAN	Random Access (user supplies address in Bits 13-31). Default: Sequential. IOCS supplies address.
2/5	BL	Blocked I/O, disc/tape only. Else: based on assignment. See Section 7.5.4.
2/6	EXP	Go to Words 8, 9, 10 instead of Words 1 and 2. Default: Words 1 and 2. *For terminal I/O, see also TSM, Section 5.6.4, this volume.
2/7	Reserved.	
2/8	DFD	Device Format Definition. When set, special definitions for 7-track tapes, ALIMs, ADSs, etc. are indicated in bits 9-12. (See Table 7-7.)
		Status Flags Set by Handlers and System
3/0	OP	Operation in Progress. (I/O request has been queued.) Bit is reset after I/O post processing is complete.
3/1	ERR	Error Condition found
3/2	BB	Invalid blocking buffer control pointers encountered in blocking or deblocking
3/3	PRO	Write protect violation
3/4	INOP	Device is inoperable
3/5	BOM	BOM (load point) or illegal volume number (multi-volume) on mag tape
3/6	EOF	End of File (TSM also sets this bit when <CTRL C> typed on terminal)
3/7	EOM	End of Medium (TSM also sets if other than <CR> at bottom of screen.) (End of tape, end of disc file.)

Table 7-3
FCB Bit Interpretations (Page 1 of 2)

		Status for Extended I/O Devices Returned by Handlers
3/10	TIME	Last command exceeded timeout value and was terminated
3/16	ECHO	Echo on Channel
3/17	INT	Post-Task-Controlled interrupt on channel
3/18	LEN	Incorrect record length on channel
3/19	PROG	Channel program check
3/20	DATA	Channel data check
3/21	CTRL	Channel control check
3/22	INTF	Interface check
3/23	CHAI	Channel chaining check
3/24	BUSY	Controller/device busy
3/25	ST	Controller/device status modified
3/26	CTR	Controller end
3/27	ATTN	Attention
3/28	CHA	Channel end
3/29	DEV	Device end
3/30	CHK	Unit check
3/31	EXC	Unit exception
		Status for Non-Extended I/O Devices Returned by Handler
3/8-11	Test Status	Testing Status as received from a 8000 level test device (TD) issued by handler
3/12-15	DCC Status	Controller status (DCC) as received from a 4000 level TD instruction
3/16-31	Device Status	Device status as received from a 2000 level TD instruction. Not applicable for PT, CR, or TY. See Table 7-4.
		Special I/O Status
6/0		No-Wait I/O Normal End Action address not executed.
6/1		No-Wait I/O Error End Action address not executed.
6/2		KILL command. I/O was not issued.

Table 7-3
FCB Bit Interpretations (Page 2 of 2)

Table 7-4
Non Extended I/O Device Status
(2000 Level)

	Word 3 Bits															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Line Printer LP	CD TERM	PROG VIO	DEV INOP	0	0	0	0	0	0	BOF	0	0	0	DEV BUSY	0	0
IOP Line Printer	0	0	0	0	0	0	0	0	0	0	0	0	CHAN END	DEV END	UNIT CK	BOF
MAG TAPE MT	CD TERM	PROG VIO	DEV INOP	VRC	0	REW IN PROG	CRC LRC	DATA LOST	0	EOT	BOT	EOF	0	DEV BUSY	FILE PROT VIO	ODD REC LGT
MOVING HEAD DISC DM (EXCEPT 100MB)	CD TERM	PROG VIO	DEV INOP	CKSM ERR (HW)	CKSM ERR (SW)	FILE UN-SAFE	SEEK IN PROG	DATA LOST	0	SEC-TOR BIT 8	SEC-TOR BIT 4	SEC-TOR BIT 2	TCA ERR	SEC-TOR BIT 1	FILE PROT VIO	TRK ERR
FIXED HEAD DISC DF	CD TERM	PROG VIO	DEV INOP	CKSM ERR (HW)	CKSM ERR (SW)	0 $\overline{\text{SECTOR}}$ BIT 32	SLCT IN PROG $\overline{\text{SECTOR}}$ BIT 16	DATA LOST	0	SEC-TOR BIT 8	SEC-TOR BIT 4	SEC-TOR BIT 2	TCA ERR	SEC-TOR BIT 1	FILE PROT VIO	0
CARD READER/ PUNCH CD	CD TERM	PROG VIO	DEV INOP	ECHO CK.	0	PHOTO DIO ERR	0	0	0	0	0	0	0	DEV BUSY	0	0
CART. DISC DC	CD TERM	PROG VIO	DEV INOP	CKSM ERR (HW)	CKSM ERR (SW)	DEV ABNOR-MAL	SEEK IN PROG	DATA LOST	0	OFFSET ACTIVE	ADDR ERR	DUAL TRK	TCA ERR	EOP	FILE PROT VIO	SEEK ERR
MOVING HEAD DISC DM (100MB)	CD TERM	PROG VIO	DEV INOP	UNCOR DATA ERR	CKSM ERR (SW)	FAULT	SEEK IN PROG	CDR DATA ERR	0	0	ADR ERR	0	TCA ERR	0	0	SEEK ERR
OTHER*																
HIGH SPEED DATA INTERFACE (GENERIC HANDLER)	CD TERM		ERROR STATUS FORMAT (See Word 3 description)							EXTERNAL TERM	IOCB ADDR ERR	ERR ON TI ADDR FETCH	DEV EOB	EP5 ERR PRE-CLUDED REQUEST QUEUEING	(NONEXECUTE CHANNEL PROGRAM IOCB TYPE IN ERROR 00-DATA TRANSFER 01-DEV STATUS 10-COMMAND TRANSFER	

*TY, PT, CR not applicable

7.6.1 FCB Word Descriptions

7.6.1.1 Word 0

IOCS defines the I/O operation indicated by the task (READ, WRITE, etc.) in terms of the operation allowed on an assigned device or file; user defines the logical file code (lfc) used externally to assign a file or device for the operation.

Bits 0-3 This field is always zero.

Bits 4-7 Operation Code - IOCS uses a single hex digit to indicate the type of function for the device handler. Allowable functions and their definitions are unique to each peripheral device. (See Table 7-5.)

Bits 8-31 Logical File Code - Any combination of three ASCII characters is acceptable. Must be supplied by user.

7.6.1.2 Word 1

This word (or Words 8 and 9 if bit 6 of Word 2 is set) supplies a Transfer Control Word (TCW) used to access a data buffer or IOCL for I/O (see below). If no TCW definition is supplied, the transfer buffer defaults to location 0 of the task's logical address space (below the operating system) and is 4096 words (4KW) maximum.

Bits 0-11 Count - Three hex digits specify the number of units (bytes, halfwords, or words) to be transferred to or from a device or file. The count must include a carriage control character, if applicable. The units the count relates to are determined by the data buffer address in bits 12-31. The maximum value of this field is 4096 words. For blocked files, 254 bytes is the maximum transfer. For unblocked files, the maximum value of this field is 4096 words. A zero, negative, or greater than maximum count will default to the maximum transfer count.

or

Number of Data or Command Chain Doublewords - For data/command chaining (e.g., to a GPMC or ADI controller via the Execute Channel service) SVC 1,X'25', bits 0-11 indicate the number of data or command chain doublewords in the I/O Command List (IOCL). The address of the list is provided in bits 12-31.

The F bit (12) and C bits (30 and 31) of the data buffer address are set by IOCS according to the definitions in the following bits.

Bits 12, 30 and 31 — Format Code - These bits specify byte, halfword, or word addressing for data transfers. They are interpreted as follows:

Type of Transfer	F (Bit 12)	CC (Bits 30-31)
Byte	1	xx
Halfword	0	y1
Word	0	00

where:

xx - Byte number (00, 01, 10 or 11)

y - 0 Left Halfword
1 Right Halfword

00 - Word

If a halfword or word transfer is specified and a device accepts only bytes, IOCS adjusts the count accordingly (x2 or 4). If a byte transfer is specified and a device accepts only halfwords or words, IOCS checks to see that the number of bytes in the buffer is an even multiple of the requested transfer and that the data buffer address is on an acceptable boundary. If these conditions exist, IOCS adjusts count accordingly (times 0, 2, or 4) and initiates the transfer. If the conditions are not met, the request is treated as a specification error. Table 7-6 indicates transfers that are acceptable for various devices. Addresses that will produce specification errors are flagged with an asterisk.

Note that IOCS operations described above enable the user to specify byte transfers beginning on a word boundary or word transfers on any device, whether the device operates on bytes, words, or halfwords.

Doubleword addressing is not allowed; IOCS will abort the task.

Bits 13-29 — Data Buffer Address - Specifies the start address of a data buffer reserved by the user for reads and writes. Bounding requirements are indicated in Table 7-6.

or

Data/Command Chain Address - Specifies the address of an IOCL to use when the Execute Channel service (SVC 1,X'25') is called. The IOCL in turn supplies an IOCD entry describing the transfer count, buffer address and other control information for each command or data transfer to the device.

Table 7-5. Device Functions (Standard Devices)

Operation	IOCS Op Code	Line Printer (LP)	Card Reader (CR)	Floppy Disc (FL)	Mag Tape (MT)	Disc (DM/DF/DC)	Data (HSD) Interface
Open (M.FILE)	0	IOCS opens. (See Section 7.8)	NOP	IOCS opens (See Section 7.8.9)	IOCS opens (See Section 7.8.9)	IOCS opens (See Section 7.8.9)	NOP - intended for user device specific processing.
Rewind (M.RWND)	1	Eject; set BOM bit (3/5) in FCB.	Set BOM bit.	Set current block address to zero (FAT)	Rewind Tape.	Set current block address to zero (FAT)	Send device command
Read Record (M.READ)	2	Spec error	Read card (80 max ASCII) (120 max BIN)	Read to data buffer as described in TCW	Read to data buffer as described in TCW	Read to data buffer as described in TCW	Read to data buffer as described in expanded TCW
Write Record (M.WRIT)	3	Write from data buffer described in TCW	Spec error	Write from data buffer described in TCW; if blocked, IOCS writes 'n' data buffers to blocking buffer before output	Write from data buffer described in TCW; if blocked, writes 'n' data buffers to blocking buffer before output	Write from data buffer described in TCW; if blocked, IOCS writes 'n' data buffers to blocking buffer before output	Write from data buffer described in expanded TCW
Write EOF (M.WEOF)	4	NOP	Spec error	If blocked IOCS writes EOF. If unblocked writes X'0F00'	Write EOF	If system file or blocked, IOCS writes EOF else NOP	Send device command
Execute Channel (Extended I/O or GPMC or HSD)	5	Spec error	Spec error	Invalid	Invalid or Execute Channel if Extended I/O	Invalid or Execute Channel if Extended I/O	Logical or physical 'Start I/O' format request

Table 7-5. Device Functions (Standard Devices)

Operation	IOCS Op Code	Line Printer (LP)	Card Reader (CR)	Floppy Disc (FL)	Mag Tape (MT)	Disc (DM/DF/DC)	High Speed Data (HSD) Interface
Advance Record (M.FWRD)	6	Spec error	Skip card	If blocked, advance record; if unblocked, advance one 192W block.	Advance record	If blocked, advance record; if unblocked, advance one 192W block.	Send device command
Advance File (M.FWRD)	7	Spec error	Advance file (past X'OF' - pseudo EOF)	Advance file (past EOF)	Advance file (past EOF)	NOP	Send device command
Backspace Record (M.BACK)	8	Spec error	Spec error	If blocked, backspace record; if unblocked, backspace one 192W block	Backspace record	If blocked, backspace record; if unblocked, backspace one 192W block	Send device command
Backspace File (M.BACK)	9	Spec error	Spec error	Backspace file (past EOF)	Backspace file to previous EOF	NOP	Send device command
Upspace (M.UPSP)	A	Upspace	NOP	Format diskette. Virgin diskettes must be formatted proir to normal usage.	Multivolume: If BOT, writes volume record. If EOT, performs ERASE, writes EOF, and issues MOUNT message. Not Multi-volume: Writes a 10 byte record of ASCII blanks.	NOP	Send device command
Erase or Punch Trailer Not user IOCS/handler provide call automatically	B	NOP	NOP	Spec error	Multi-volume only. Same as upspace above. Erases 4" of tape before writing.	NOP	Send device command

Operation	IOCS Op Code	Line Printer (LP)	Card Reader (CR)	Floppy Disc (FL)	Mag Tape (MT)	Disc (DM/DF/DC)	High Speed Data (HSD) Interface
Eject/Punch Leader	C	Eject to top of form	NOP	Spec error	Write dummy record with eject control character as first character	NOP	Send device command
Close (M.CLSE)	D	IOCS closes (See Section 7.8)	IOCS closes (See Section 7.8)	IOCS closes (See Section 7.8)	IOCS closes (See Section 7.8)	IOCS closes (See Section 7.8)	NOP, complementary function to open
Reserve FHD Port	E	Spec Error	Spec Error	Spec Error	Spec Error	Reserve port - 4 MB disc only; else, spec error	Send device command
Release FHD Port	F	Spec Error	Spec Error	Spec Error	Spec Error	Release port - 4 MB disc only; else, spec error	Send device command

Note: NOP = No Operation Performed
Spec Error = Illegal Opcode

Example: M.READ on Line Printer

Table 7-5a. Device Functions (Terminals)

TERMINALS (TY) (Describes Handler Action Only)				
Operation	IOCS Op Code	Handler = H.ASMP (ALIM)	Handler = H.A8IOP (8-Line)	Handler = H.TY10 (ADS)
Open M.FILE	0	NOP	Initialize IOP Channel if Necessary	NOP
Rewind M.RWND	1	NOP	NOP	Eject; Set BOM Bit in FCB
Read Rec. M.READ	2	Read to Data Buffer	Read to Data Buffer	Read to Data Buffer
Write Rec. M.WRIT	3	Write Record to Terminal	Write Record to Terminal	Write Record to Terminal
Write EOF M.WEOF	4	NOP	NOP	Sets Ring Mode
Execute Channel	5	Execute Channel	SPEC ERROR	NOP
Advance Record M.FWRD	6	Connect Communi-cations Channel	Set Data Terminal Ready	NOP
Advance File M.FWRD	7	Disconnect Com-munications Channel	Reset Data Terminal Ready	NOP
Backspace Record M.BACK	8	Initialize Dev-ice and Set Timeout Value	Used by J.TINIT to Initialize Terminals	NOP
Backspace File M.BACK	9	Clear Break Status Flag Word	NOP	Reset Break (Dummy Service)
Upspace M.UPSP	A	SPEC ERROR	SPEC ERROR	NOP
Erase/Punch Trailer	B	Transmit Break	Set/Reset Break (depends on flags in FCB)	NOP
Eject/Punch Leader M.EJECT	C	SPEC ERROR	SPEC ERROR	NOP
Close M.CLSE	D	NOP	NOP	NOP
Reserve FHD Port	E	SPEC ERROR	SPEC ERROR	NOP
Release FHD Port	F	SPEC ERROR	SPEC ERROR	NOP

NOP = No operation performed SPEC ERROR = Illegal operation code

Table 7-6. Acceptable/Nonacceptable Device Transfers Specified in TCW Word 1, Bits 12 and 30/31

FCB Format/Bounding	Device Transfers					
	Words		Halfwords		Bytes	
	Bits		Bits		Bits	
	12	30/31	12	30/31	12	30/31
Word	0	00	0	00	0	00
Left Halfword	0	01	0	01	0	01
Right Halfword	0	11*	0	11	0	11
Byte, first even	1	00	1	00	1	00
Byte, third even	1	10*	1	10	1	10
Byte, any odd	1	x1*	1	x1*	1	x1

* Writes to/reads from device will result in IOCS abort with this FCB definition.

7.6.1.3 Word 2

Word 2 provides optional control specifications for I/O.

Bits 0-7 Operational Specifications - These eight bits enable the user to specify that special operations such as no-wait I/O be performed by IOCS. The meaning of each bit is as follows:

0 - If this bit is set, IOCS will return to the user immediately after the I/O operation is queued. If the bit is reset, IOCS will exit to the calling program only when the requested operation has been completed.

1 - If this bit is set, error processing will not be performed by either the device handler or IOCS. Normal error processing for disc and magnetic tape is automatic error retry. Error processing for unit record devices except the system console is accomplished by IOCS typing the message "INOP" to the console which allows the operator to retry or abort the I/O operation. If the operator aborts the I/O operation, or if automatic error retry for disc or magnetic tape is unsuccessful, an error status message is typed to the console. An error return address is not applicable if this bit is set, however, the device status will be posted in the FCB (unless bit 3 is set).

2 - When this bit is set, data formatting is inhibited. Otherwise, data formatting is performed by the appropriate device handler. (See Bits 8 through 12 for further explanation.)

3 - If this bit is set, the device handlers perform no status checking and no status information is returned. Hence all I/O will appear to complete without error.

4 - When this bit is set, file accessing will occur in the random mode. Otherwise, sequential accessing will be performed.

5 - If set, a blocked file is specified (disc or tape assignments only).

6 - Expanded FCB present (Words 8-15). This takes advantage of a larger I/O transfer quantity (in bytes), a 24-bit addressing field, and a 32-bit random access address. For Extended I/O operations, up to two interrupt status words are then returned after I/O complete. When this bit is set, IOCS assumes the FCB is 16 words long. The information in Words 8 and 9 is used instead of the data in Word 1. Also, the random access address in Word 10 is used instead of the data in Word 2.

7 - If this bit is set, a task will not be aborted even if an error condition occurs that would cause the task to be aborted.

Bits 8-12 Device Control Specification - These bits contain control specifications unique to certain devices. Device handlers interpret and process the specifications. A bit setting is meaningful only when a particular type of device is assigned as indicated in Table 7-7, columns 2 and 3. (Column 1 indicates default control.)

Bits 13-31 Random Access Address - This field contains a block number (zero origin) relative to the beginning of a disc file, and specifies the base address for read or write operations.

Note: If bit 6 of Word 2 is set, the expanded random access address in Word 10 is used instead of bits 13-31 above.

For High Speed Data (HSD) Interface applications, Word 2 bit meanings are as follows:

Bit 2 For Execute Channel Program, if set, physical IOCL is specified.

Bit 8 Request Device Status After Transfer - This bit indicates an IOCB should be added to the IOCL to retrieve device specific status after the data transfer has completed.

Bit 9 Send Device Command Prior to Data Transfer - This bit indicates an IOCB should prefix the data transfer to transmit a device command word to the device. The value sent is the 32-bit expanded random access address.

Bit 10 Disable Timeout for this Request - This bit indicates 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 has meaning on read operations.

Bit 11 Set UDDCMD from Least Significant Byte of Word 2 - This bit indicates that the UDDCMD byte in the data transfer IOCB should be set to the least significant byte of the random access field of the FCB. This provides the ability to pass additional control information to the device without modifying the device driver.

Bits 24-31 If bit 11 is set, these bits define the UDDCMD field of the generated IOCB, overriding the default value from a handler table.

Table 7-7. Default and Special Device Formatting

Device	Default (Bit 2=0)	Override (Bit 2=1)	Bit 8	Bit 9	Bit 10	Bit 11	Bit 12
Card Reader Punch (CR or CP)	Read/Punch Auto Select Mode. First column - Rows 2-5, all punched = binary, no translation, max. 120 bytes. Not all punched = ASCII, translate, max 80 bytes. EOF = column 1, Rows 2-5 only punched. Binary, no translation (X'0F'). Max 120 bytes.	See Bit 8.	0=ASCII read/punch 1=Binary read/punch				
Floppy Disc (FL)	Report EOF if X'0F00' encountered in word 0 during read of an unblocked record.	No EOF reporting on unblocked reads.					
Line Printer (LP)	Interpret first character as carriage control.	No carriage control interpretation					
TLC Terminal or Teletype (TY or CT)	Interpret first character as carriage control.	No carriage control					
High Speed Data (HSD) Interface (Generic Handler)	Logical IOCL	See Word 2 definition	See Word 2 definition	See Word 2 definition	See Word 2 definition	See Word 2 definition	See Word 2 definition
Paper Tape Reader (PT)	Read in formatted mode, skipping leader	Read unformatted	0=Do not skip leader 1=Skip leader				
Paper Tape Punch (PT)	Punch in formatted mode.	Punch unformatted					

Table 7-7 Default and Special Device Formatting

Device	Default (Bit 2=0)	Override (Bit 2=1)	Bit 8	Bit 9	Bit 10	Bit 11	Bit 12
Mag Tape (MT, 7-track only)	Binary, ODD parity, 800 bpi	See Bits 8-10	0=Interchange (binary coded decimal) 1=Packed (binary)	If Bit 8=0: 0=EVEN parity 1=ODD parity	0=800 bpi 1=556 bpi		
(MT 9-track only)	N/A						
Discs (DM,DF)	N/A						
ALIM (Asynchronous Line Interface Module) Terminals (TY)	Read: receive data (bytes) defined for transfer count. Write: formatted	Bit 2 Read 0 0 1 0 Write 0 0 1	Bit 8 / Bit 9 1 / 0 =Blind mode reset 0 / 1 =Echo on read N/A / N/A =Receive data 0 / 0 =Receive data N/A / 0 =Formatted write N/A / 1 =Initialize device N/A / N/A =Unformatted write		On Read: 1= Inhibit conversion of lower case characters to upper case 0= Convert		
ADS (Asynchronous Data Set) (TY)			1 = Handler to issue LCW command to ADS 0 = No LCW		Same as ALIM above	1= Handler to issue TXB command to ADS 0= No TXB	Expanded FCB only (Bit 6 set) 1=Post External Asynchronous Interrupt status in Word 11 0=Do not post status
8-Line Asynchronous Communications Multiplexer (TY)	Read: perform special character formatting. Write: first character is for form control.	Read:read n bytes with no formatting.* Write n form control. *Note: User is responsible for setting appropriate bits for his particular application.	Transmit break (erase, punch (trailer): 0=Stop transmitting break 1=Start transmitting break. Read: 1=ASCII control character detect	Read: 0=Echo by controller 1=No echo by controller Write: 0=Normal write 1=Initialize device (load UART parameters)	Read (if Bit 2=0): 0=convert lower case characters to upper case 1=Inhibit conversion of lower case characters to upper case.	H.F8IOP only Read: 0=no Special character detect 1=special character detect Write: 0=normal write 1=write with input subchannel monitoring	H.F8IOP only Read: 0=do not purge type ahead buffer 1=purge type ahead buffer

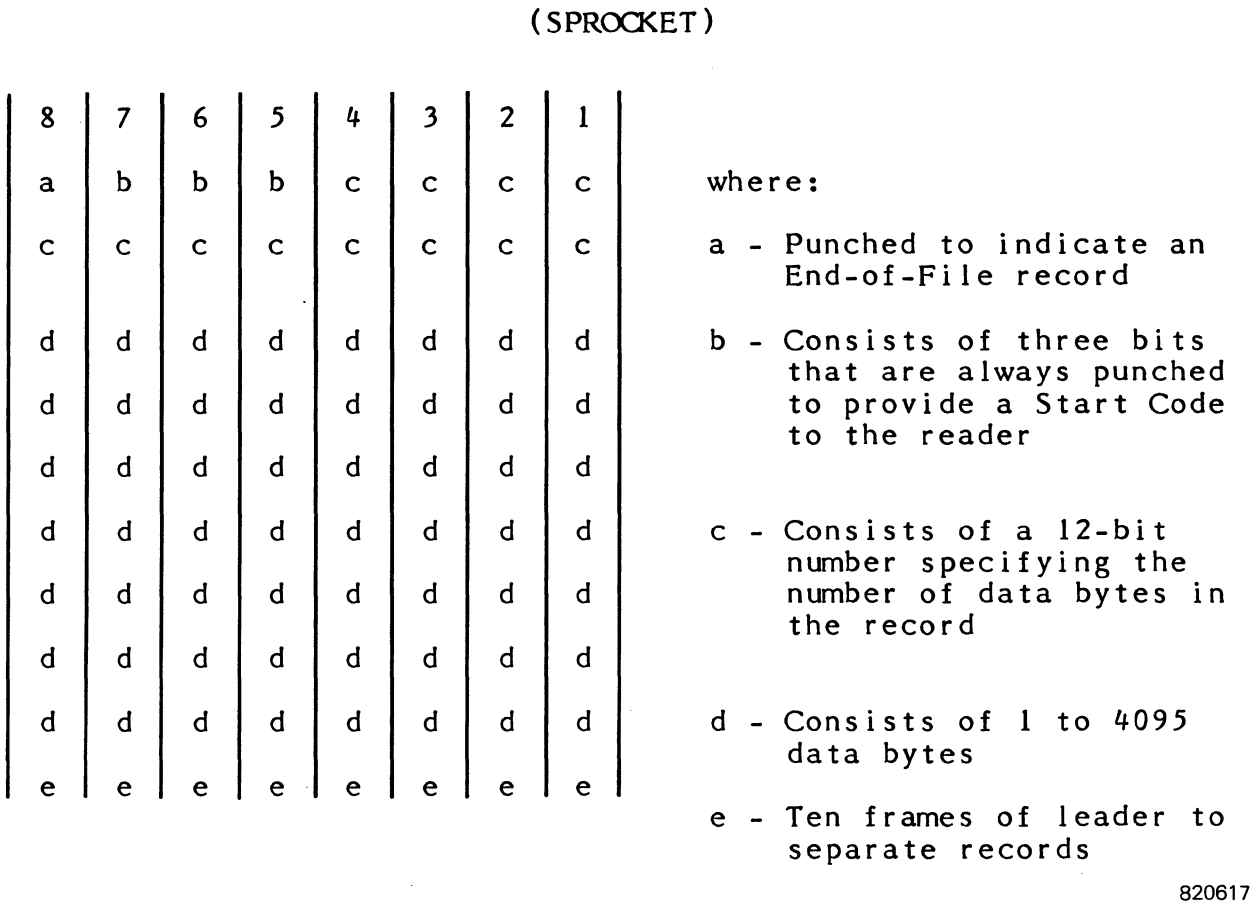


Figure 7-4 Punched Tape Format

Table 7-8 Standard Terminal and Line Printer Carriage Control Characters and Interpretation

Result

Control Character (Position 1)	Hex Value	Terminal	Line Printer
Blank	20	Linefeed/carriage return before write	Single space before print
0	30	Two linefeeds/ carriage return before write	Doublespace before print
1	31	Five linefeeds/ carriage return before write	Eject page before print
+	2B	No linefeed before write. Carriage return only.	No space before print
-	2D	Same as 1	Eject and print SLO header record*

* A task can output up to three 132-byte maximum header records at a time to an SLO file. Each record would have a minus sign in the first position. (See Section 7.6.1.)

7.6.1.4 Word 3

Word 3 returns I/O status. IOCS uses 32 indicator bits to return the status, error, and abnormal conditions detected by handlers during the previous or current device operation. The task can examine these bits as needed. Individual bit assignments for bits 0-7 apply to any device. Bits 8-31 mean different things depending on the device. For extended I/O devices, individual bits are shown in the FCB Word 3 area of Figure 7-3 and described in Table 7-3. For non-extended I/O devices, test status, controller (DCC) status, and device status are returned as described in Table 7-4, which accompanies the FCB diagram in Figure 7-3.

For High Speed Data (HSD) Interface applications, Word 3 error status bits have the following meanings:

Bits 17-18	Unused
Bits 19	Record length error
Bit 20	Parity Error
Bit 21	Nonpresent memory (NPM)
Bit 22	Invalid opcode in IOCB Word 0
Bit 23	Device inoperable
Bit 24	Data buffer overflow

7.6.1.5 Words 4 and 5

Word 4 defines record length. This word is used by IOCS to indicate the actual number of bytes transferred during a read or write.

Word 5 defines I/O queue address in bits 8-31. This field is set by IOCS to point to the I/O queue for an I/O request initiated from this FCB.

7.6.1.6 Word 6

In bits 0-7, IOCS returns special status bits as described in Table 7-3.

Word 6 defines a wait I/O error return address in bits 8-31. Specify the address to transfer control to in the case of an unrecoverable error. If this field is not defined, an unrecoverable error is detected, and the user has not set bits 1 and 3 of Word 2 to inhibit error processing, IOCS aborts the task.

7.6.1.7 Word 7

Word 7 must not be written by the user. It defines the File Assignment Table (FAT) entry associated with all I/O performed on behalf of this FCB. The FAT address is supplied by IOCS.

7.6.1.8 Word 8

Word 8 begins expanded TCW definition. This area of the FCB can be used to define transfers larger than 4096 words, e.g., for extended I/O devices.

Bits 8-31	Expanded data buffer address - specifies the start address of a data buffer reserved by the user for reads or writes. This must be a logical byte address with no format bit present. Word bounding is required for some devices if unblocked.
	or
	Expanded Data/Command Chain List Address - Word address that points to the data or command chain list (IOCL) if using Execute Channel service, SVC 1,X'25'.

7.6.1.9 Word 9

Word 9 continues the expanded TCW definition.

Bits 0-31	Expanded transfer count - eight digits specify the number of bytes to be transferred. Note that the transfer count supplied here is always in byte units. Must include the carriage control character, if applicable. A zero or negative count defaults to the maximum allowable count (254 bytes if blocked).

7.6.1.10 Word 10

Word 10 defines expanded random address. This word contains a block number (zero origin relative to the beginning of the disc file). It is the start address for the current read or write operation.

For High Speed Data (HSD) Interface requests in non-Execute Channel Program format, this word defines a device command.

7.6.1.11 Word 11

Word 11 returns status.

Bits 0-31 — Status word 1 - for extended I/O, if an error is detected during an I/O operation, these 32 bits are returned by the SENSE command.

or

Status EAI - for communications adapter (CA) interface, returns external asynchronous interrupt (EIA) status if bit 12 of Word 2 is set.

7.6.1.12 Word 12

Word 12 returns status word 2. This is the second status word returned from extended I/O hardware.

For High Speed Data (HSD) Interface applications, this word contains status sent from the user's device.

7.6.1.13 Word 13

Word 13 defines normal return address for no-wait I/O. In bits 8-31, specifies address to transfer control to when a no-wait I/O operation is complete. The code at this address must be terminated with a call to IOCS for post-processing service (SVC 1,X'2C'). See Section 7.4.2.

For High Speed Data (HSD) Interface applications, this address plus 1 word is the location to which control is transferred on asynchronous notification.

7.6.1.14 Word 14

Word 14 defines an error return address for no-wait I/O. In bits 8-31, specifies (optionally) the address to transfer control to when no-wait I/O completes with an error. The code at this address must be terminated with a call to IOCS for post processing service (SVC 1,X'2C'). See Section 7.4.2.

7.6.1.15 Word 15

Word 15 is reserved for I/O service expansion.

7.6.2 Macros

The M.DFCB or the M.DFCBE macro can be used to define an FCB or an expanded FCB, respectively.

Syntax:

```
M.DFCB [E]   label, lfc, count, buffer, [error]
             , [random] , [NWT] , [NER] , [DFI] , [NST]
             , [RAN] , [ASC|BIN] , [LDR|NLD] , [INT|PCK] , [EVN|ODD]
             , [556|800] , [nowait] , [nowaiterror]
```

where:

label	ASCII string to use as symbolic label for address of this FCB.
lfc	Logical file code; see Word 0, bits 8-31.
count	Transfer count. See Word 1, bits 0-11. Specify in number of bytes. For expanded FCB, see Word 9, bits 0-31.
buffer	Start address of data buffer. (Reserve on Word boundary.) See Word 1, bits 12-31. For expanded FCB, see Word 8, bits 8-31.
error	Error return address for wait I/O (see Word 6, bits 8-31).
random	Random access address. See Word 2, bits 12-31. For expanded FCB, see Word 10, bits 0-31.
NWT/NER/ DFI/NST/ RAN/	See Table 7-3.
ASCII/BIN	For CR or CP, see Table 7-7.
LDR/NLD	For PT (Reader), see Table 7-7.
INT/PCK	For 7-track mag tape, see Table 7-7.
EVN/ODD	Parity, see Table 7-7.
556/800	Bits per inch density, see Table 7-7.
nowait	M.DFCBE (expanded FCB) only. Specifies address for normal no-wait I/O end action return.
nowaiterror	As above. Specifies address for no-wait I/O error end action return.

7.6.3 Sample FCB Setup Non-Macro

User defines an FCB for terminal message output:

```
TERM        DATAW   G'UT '
            GEN     12/B(TY.LEN),20/B(MESSAGE)
            REZ     6W
MESSAGE     EQU     $
            DATAB   C' "M"J CREATE FAILED. ERRTYPE   '
TY.LEN      EQU     $-MESSAGE
ERRTYPE     EQU     $-1B
```

Notes: The source code uses TERM with M.WRIT to access this FCB. The logical file code is UT. The Transfer Control Word built with the GEN directive has a transfer count equal to the message (TY.LEN EQU $-MESSAGE), computed by the Assembler.

The buffer start address is at MESSAGE (address is supplied by the Assembler). In the actual message, "M indicates carriage return and "J indicates line feed before output. The last byte of the message comes from Register 5 when an error occurs as defined by:

```
C.MSG       EQU     $
            M.CONBAD
            STB     R5,ERRTYPE
            M.WRIT  TERM
```

7.6.4 Sample FCB Setup-Macro

```
            M.DFCB  TERM, UT, TY.LEN, MESSAGE
MESSAGE     EQU     $
            DATAB   C' "M"J CREATE FAILED. ERRTYPE   '
TY.LEN      EQU     $-MESSAGE
ERRTYPE     EQU     $-1B
```

7.7 Setting Up Type Control Parameter Blocks (TCPB's) for the OPCOM Console

Messages are sent from a task to the OPCOM console and a response optionally read back via a TCPB. The TCPB sets up task buffer areas for messages output by the task and reads back from the console.

The TCPB is comprised of a Write and an optional Read Transfer Control Word defined like the TCW in Word 1 of the FCB. If read back from the terminal is not desired, the Read TCW must be zero. The user must perform his own carriage return and line feed.

The size of the read buffer should include space for both an input character count preceding the message (provided by IOCS) and the carriage return (end-of-record) character typed by the user at the end of an input line.

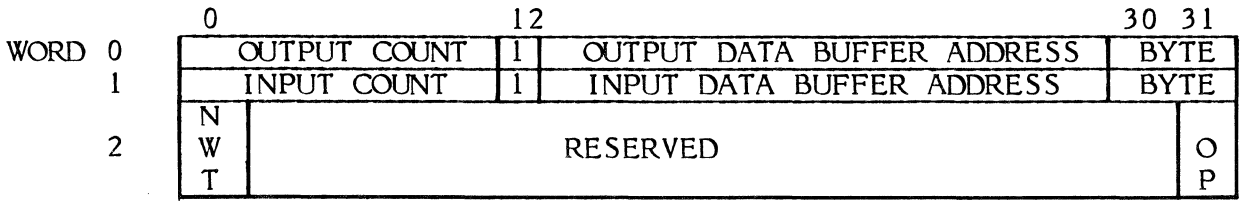
The count of input characters actually typed is placed by IOCS in the first byte of the read buffer. This input count does not include the carriage return character.

The byte transfer count is normally used by a task as maximum allowed input before termination. If the operator types in the maximum count without having typed a carriage return, the read is terminated.

The end-of-record character (carriage return) is normally allowed for in both the read and write buffer transfer count. If five characters are expected, the read transfer count would typically be for six characters. This allows the operator to type in and verify the correctness of all five characters before terminating the input message by striking a carriage return.

Message transfers are always in bytes, hence the buffer address must be a byte address (F-bit setting-bit 12 as shown in Figure 7-5).

If the NWT bit is set (Word 2, bit 0) IOCS returns immediately to the calling task after the message is queued. The task can subsequently examine the OP indicator set by IOCS in Word 2, bit 31 to see if the requested transfer is complete (0) or in process (1).



Bit Interpretations

NWT	Word 2, Bit 0 is set to define no-wait I/O.
OP	Word 2, Bit 31 is set by IOCS if an operation is in progress.

820618

Figure 7-5 Type Control Parameter Block

7.8 Services

Services pertaining to file and device allocation and I/O are summarized below and described in detail in this section. Special operations performed for a task are:

OPEN - If not issued by the task, IOCS opens the file or device read-write.

CLOSE - If not issued by the task, the file is closed automatically and a device is deallocated automatically during task termination.

Services are organized alphabetically by macro names. The services that are not macro callable, but yet available to the user, are described at the end of the section.

SVC numbers and short alpha descriptors are listed under the macro name. Appendix B provides cross reference charts.

Standard MPX-32 conventions are used in syntax statements.

7.8.1 M.ALOC - Allocate File or Peripheral Device

The M.ALOC service 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 service may also be used to equate a new logical file code with an existing logical file code.

Under memory-only MPX-32, this service dynamically allocates peripheral devices only; files are not supported.

Entry Conditions

Calling Sequence:

```
M.ALOC    retad,lfc,function,arga,argb, [MOUNT] , [UNBLOCKED]
          [,WAIT]

(or)

LA        R1,retad
LI        R5,function
SLL       R5,24
ORMW      R5,lfc
[SBR      R5,0 for MOUNT option inhibited                    ]
 SBR      R5,1 for UNBLOCKED option
[SBR      R5,2 for WAIT for resource requested               ]
LA        R6,arga
LA        R7,argb
SVC       1,X'40' (or) M.CALL  H.MONS,21
```

where:

retad — is the denial return logical address

function — is the function code as follows:

1=assign logical file code to a user or system permanent file
2=assign logical file code to a system file code
3=assign logical file code to a peripheral device
4=assign logical file code to a defined logical file code
5=assign logical file code to a system permanent file only

lfc — contains the one to three ASCII character logical file code to be assigned. The first byte contains zero to accommodate the function code. The lfc is then left-justified and blank-filled in the three remaining bytes.

arga,argb contents unique to each function are as follows:

1	arga	contains the one to eight ASCII character permanent file name. If a user name is not associated with the calling task, the system file of the name specified is allocated. If a user name is associated with the calling task, an attempt is made to allocate a user file of the name specified. If unsuccessful, a system file is allocated.
	argb	contains the one to eight character, left justified, blank filled password required to access the file. If the file is not password protected, contains 0.
2	arga	contains the character string SLO or SBO in bytes 1,2,3
	argb	contains the number of 192 word blocks required for allocation to the file
3	arga	contains the device type code (see Appendix A, Table A-1) in byte 0 and optionally the channel number in byte 2 and the device subaddress in byte 3. If the device subaddress is present, the most significant bit of byte 2 must be set. If the channel number is present, bit 0 of byte 0 must be set. For magnetic tape devices, byte 1 = volume number or 0 if single volume.
	argb	if arga defines a disc file = size of file (number of 192-word blocks required). If arga defines a magnetic tape = four-character reel identifier. For all other devices = 0.
4	arga	contains the previously defined logical file code
	argb	equals zero
5	arga	contains the one to eight character permanent file name of a system file
	argb	contains the one to eight character, left-justified, blank filled password required to access the file. If the file is not password protected, contains 0.

MOUNT is a character string indicating to the macro that the optional MOUNT message should not be sent.

UNBLOCKED is a character string indicating to the macro that the file being allocated is to be unblocked. If not specified, the file will be blocked automatically. This option should not be used when specifying function code 2 or 4.

WAIT is a character string indicating the caller wishes to be queued for the resource and will relinquish the CPU until the resource becomes available.

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 1,6 — for denial returns if the requested file or device cannot be allocated.

Registers:

R6 = 0 if file or device busy. Condition codes 1-4 are set to indicate why:

CC1 set=Permanent file is exclusively locked
CC2 set=File Lock Table (FLT) is full
CC3 set=Nonshared device is busy (already allocated)
CC4 set=Disc space is not available

=n if an error condition exists as described next. Note that when R6=n, CC1-CC4 are not applicable.

External References

Error Conditions

R6=1 - permanent file non-existent
=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 ASSIGN 1
=8 - unrecoverable I/O error to SMD
=9 - SGO assignment specified by terminal task
=10 - no 'UT' file code exists for terminal task
=11 - invalid RRS entry
=12 - LFC in ASSIGN4 non existent
=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

Output Messages: See MOUNT message description in Section 7.3.1.1.

7.8.2 M.BACK - Backspace Record or File

The M.BACK service is not applicable for system files (i.e., SYC, SGO, SLO, SBO). This service performs the following functions for backspacing records:

If a blocked file is output active, a purge is issued prior to the backspace function. After the specified number of records are backspaced, returns to the user.

Backspaces specified number of records.

M.BACK performs the following functions for files:

If a blocked file is output active, an end-of-file and purge are issued prior to performing the backspace file function. Records are backspaced until an end-of-file record is found.

The specified number of files are backspaced.

The read/write control word then points to the end-of-file just encountered.

Entry Conditions

Calling Sequence:

```
M.BACK   fcb,[R],number

(or)

LA       1,fcb
LNW      4,number
{SVC     1,X'35'  or  M.CALL  H.IOCS,9  }
{SVC     1,X'36'  or  M.CALL  H.IOCS,19 }
BIB      4,$-1W
```

where:

fcb	FCB address
number	address of word containing the number of records or files to backspace
R	backspaces by record (SVC1,X'35'); else backspaces by file (SVC1,X'36')
$-1W	branches back to SVC until reaches last word of Register 4

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO06	Invalid blocking buffer control cells for blocked file.
IO13	Illegal operation for system file

Output Messages: None

7.8.3 M.CLSE - Close File

The M.CLSE service marks a file closed in the File Pointer Table (FPT) and the count of open files (DFT.OPCT) is decremented. If any logically equivalent files (ASSIGN4) are open, i.e., if count after decrementing is not equal to zero, no further action is taken.

If the file is a system file or blocked file, purges any active output blocking buffer. The file is marked closed (open bit cleared in FAT).

For files assigned to SYC or SGO, the current disc address is used to update the Job Table for Job Control.

This service issues an EOF prior to purging system files SLO and SBO which were opened for read/write. Also issues an EOF prior to purging for blocked files which are output active.

Close requests to a file that is already closed are ignored.

Entry Conditions

Calling Sequence:

```
M.CLSE      fcb[,[EOF] [,REW] ]

(or)

 LA         1,fcb
[SVC        1,X'38'  or M.CALL  H.IOCS,5]
[SVC        1,X'37'  or M.CALL  H.IOCS,2]
 SVC        1,X'39'  or M.CALL  H.IOCS,23
```

where:

fcb is the FCB address

EOF Writes EOF (SVC 1,X'38'). See M.WEOF description.

REW Rewinds file or device (SVC 1,X'37'). See M.RWND description.

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases: None

Output Messages: None

7.8.4 M.CREATE - Create Permanent File

The M.CREATE service allocates disc space for the specified permanent file and writes a corresponding entry into the SMD. Optionally, the allocated space is zeroed.

Entry Conditions

Calling Sequence:

M.CREATE filename,blocks,devtype,[devchan],[subaddr],

[{R / P}, password], [S] ,[N] ,[F] ,[type] ,[Z]

(or)

```
  LD      R6,filename
  LW      R2,blocks
 [SBR     R2,6    if R  -  read only
  SBR     R2,7    if P  -  password only
  SBR     R2,2    if N  -  not SAVE DEVICE file
 _SBR     R2,3    if F  -  FAST file                ]
  ZR      R3
  LI      R1,devtype
  SLL     R1,16
 [ADI     R3,devchan
  SLL     R3,8
  ADI     R3,subaddr
 _SBR     R3,0    if devchan present               ]
  SBR     R3,16   if subaddr present
  ORR     R1,R3
  ZR      R4 (or) LD   R4,password
  ZR      R1
 [LI      R1,X'type'  if type present
  SBR     R1,0    if S  -  system file
 _SBR     R1,1    if Z  -  pre zero                ]
  SVC     1,X'75' (or) M.CALL   H.FISE,12
```

where:

filename — is a doubleword containing the one to eight ASCII character, left-justified, blank filled name of the file. Each character in the name must have an ASCII equivalent in the range 21 through 5F (printable) and may not contain a comma

blocks — contains the size of the file specified as a multiple of 192 word blocks

devtype — the device type code which specifies the type of disc on which the file is to reside as follows:

01 = DC
02 = DM
03 = DF

devchan — is the optional channel number of the particular device on which the file is to reside (Bit 0 of R3 is set to indicate the presence of the channel number)

subaddr — is the optional subaddress of the particular device on which the file is to reside (Bit 16 of R3 is set to indicate the presence of the subaddress)

R,P — optional character to indicate file access restrictions as follows:

R - Read only (password required to write)
P - Password only (password required to read or write)

If this parameter is not specified, the file may be read or written to without a password.

password — is a doubleword containing the optional one to eight ASCII character, left-justified, blank filled password. Each character must have an ASCII equivalent in the range 01 through 7F (printable, lowercase, or special). The password may not contain a comma. If a file access restriction (R or P parameter) is specified, a password must be entered. If no access restriction is specified, a password is ignored. If a password is entered, the file can be deleted only by specifying the password. In addition, the password must be entered on SAVE FILE, DELETE, and EXPAND File Manager directives. R4 is zero if no password is to be associated with the file.

S — is an optional character to indicate that the file is to be a system file. If not specified, the file is created as a user file if a user name is associated with the calling task, or as a system file if no user name is associated with the calling task.

N — is an optional character to indicate that the file is not to be saved in response to the SAVE DEVICE File Manager directive.

F — is an optional character to indicate that the file is a FAST file. If no parameter is entered, the file is created as a SLOW file.

type — is an optional parameter which is a one or two digit hexadecimal value specified by the user to identify the origin of the file as follows. For example, the system software utilizes the following file types:

ED - Editor Save File
EE - Editor Store File
FE - Editor Work File
FF - SYSGEN Created File
BA - BASIC File
CA - Cataloged Load Module

Z — is an optional character to indicate that the space allocated to the file is to be zeroed.

Exit Conditions

Return Sequence:

M.RTRN 6,7

Registers:

R7 zero if the file was not created. R6 contains the reason as follows:

R6
- = 1 if a file of the name specified already exists
- = 2 if a FAST file was specified and collision mapping occurred with an existing directory entry
- = 3 if restricted access (bit 6 or 7 of R2) was specified, but no password (R4,R5) was entered.
- = 4 if disc space is unavailable
- = 5 if the specified device (channel and/or subaddress) is not configured, or no device of the type specified is available.
- = 6 if the specified device is off-line.
- = 7 if the SMD is full
- = 8 if the specified device type (byte 1 of R3) is not configured.
- = 9 if the file name or password contains invalid characters or imbedded blanks.

External References

System Macros:

M.CALL H.FISE,1; H.FISE,2; H.FISE,5; H.FISE,7;
M.RTRN H.FISE,8; H.FISE,10; H.MONS,20; H.IOCS,4

System Subroutines:

S.ALOC8
S.ALOC9
S.FISE3

Abort Cases:

FS01 Unrecoverable I/O error to the System Master Directory (SMD).
FS02 Unrecoverable I/O error to a disc allocation map.

Output Messages:

None

7.8.5 M.CWAT - System Console Wait

The M.CWAT service suspends operation of the calling program until the specified I/O transfer is complete.

Entry Conditions

Calling Sequences:

```
M.CWAT  tcpb

(or)

LA      1,tcpb
SVC     1,X'3D'  (or)  M.CALL  H.IOCS,26
```

where:

tcpb is the address of a Type Control Parameter Block (TCPB). See Section 7.7.

Exit Conditions

Return Sequence:

```
M.RTRN
```

Registers: None

Abort Cases: None

Output messages: None

7.8.6 M.DALC - Deallocate File or Peripheral Device

The M.DALC service deallocates a peripheral device or disc file to which the specified logical file code is assigned. Dynamic deallocation of a peripheral or permanent disc file makes that resource available to other tasks. Deallocation of SLO and SBO files result in their definitions being passed to System Output for output to their terminal devices. If the specified logical file code has been equated to other logical file codes in the system, this service deallocates only the specified code.

Under memory-only MPX-32, this service deallocates peripheral devices only; files are not supported.

Entry Conditions

Calling Sequence:

```
M.DALC  lfc

(or)

LW      R5,lfc
SVC     1,X'41' (or) M.CALL  H.MONS,22
```

where:

lfc — contains the one to three ASCII character, left justified, blank filled logical file code for which deallocation is to be performed (bytes 1-3). Byte 0 is zero unless the device is magnetic tape. If tape, bit 0 of byte 0=1 indicates that the DISMOUNT message is to be output, but that the device is not deallocated. If bit 0 of byte 0=0 output DISMOUNT message and deallocate.

Exit Conditions

Return Sequence: M.RTRN

Registers: None

External References

System Macros: M.CALL, M.RTRN

Abort Cases: None

Output Messages:

```
*task DISMOUNT reel FROM UNIT xx
*task DISMOUNT reel, VOL volno FROM UNIT xx
```

where:

task — is the load module name of the task requesting that the magnetic tape unit be deallocated

reel — is the reel identifier of the magnetic tape to be dismounted

volno — is the volume number of the magnetic tape to be dismounted - if single volume = blank

xx — is the device number of the non-shared magnetic tape unit from which the tape is to be dismounted

7.8.7 M.DELETE - Delete Permanent File or Non-SYSGEN Memory Partition

The M.DELETE service deletes a specified permanent file or non-SYSGEN created memory partition.

Entry Conditions

Calling Sequence:

```
M.DELETE  filename,[S],[password]

(or)

LD        R6,filename
ZR        R3 (or) [LI   R3,G'S']
ZR        R4 (or) [LD   R4,password]
SVC       1,X'77' (or) M.CALL   H.FISE,14
```

where:

filename — is a doubleword containing the one to eight character name of an existing file to be deleted. The name is left-justified and blank filled.

S — is the optional character "S" if a system file only is to be deleted. If the character "S" is not specified, the file is assumed to be a user file whose user name is that which is associated with the calling task.

password — is a doubleword containing the one to eight character password which is left-justified, and blank filled. A password must be specified if the file has an associated password.

Exit Conditions

Return Sequence:

```
M.RTRN  6,7
```

Registers:

R7 — 0 if the specified file was not deleted. R6 contains the reason as follows:

R6 — = 1 if a file of the name specified does not exist or is a SYSGEN created memory partition, or the file is allocated, or File Lock Table space is not available (CC1 = 1 if file is allocated, CC2 = 1 if file is allocated and exclusively locked, CC3 = 1 if File Lock Table space is not available)
= 2 if a required password was not specified.

External References

System Macros:

```
M.CALL   H.FISE,2; H.FISE,7; H.FISE,8; H.MONS,20
M.RTRN
```

System Subroutines:

```
S.FISE1
S.FISE2
S.FISE3
```

Abort Cases:

FS01	Unrecoverable I/O error to the System Master Directory (SMD).
FS02	Unrecoverable I/O error to a disc allocation map.

Output Messages: None

7.8.8 M.FADD - Permanent File Address Inquiry

The M.FADD service is intended for the use of those who wish to issue I/O directly. It provides the word address of the beginning of a memory partition or the track, head, and sector address of the beginning of a disc file. The device address for disc files is also included in the result. Access restrictions are returned for disc files.

Entry Conditions

Calling Sequence:

```
M.FADD   name

(or)

LD       R6,name
SVC      1,X'43'  (or)  M.CALL  H.MONS,2
```

where:

name — is a doubleword containing the one to eight ASCII character, left-justified and blank-filled permanent file name. The file is assumed to be a user file if the calling task has an associated user name or a system file if the calling task has no associated user name. If the calling task has an associated user name but no user file of the specified name exists, the file is assumed to be a system file.

Exit Conditions

Case I, Denial Return:

Return Sequence: M.RTRN 7

Registers:

R7	bit 0	1 indicating that the specified file name cannot be located in the SMD
	bits 1-31	zero

Case II, Memory Partition:

Return Sequence: M.RTRN 6,7

Registers:

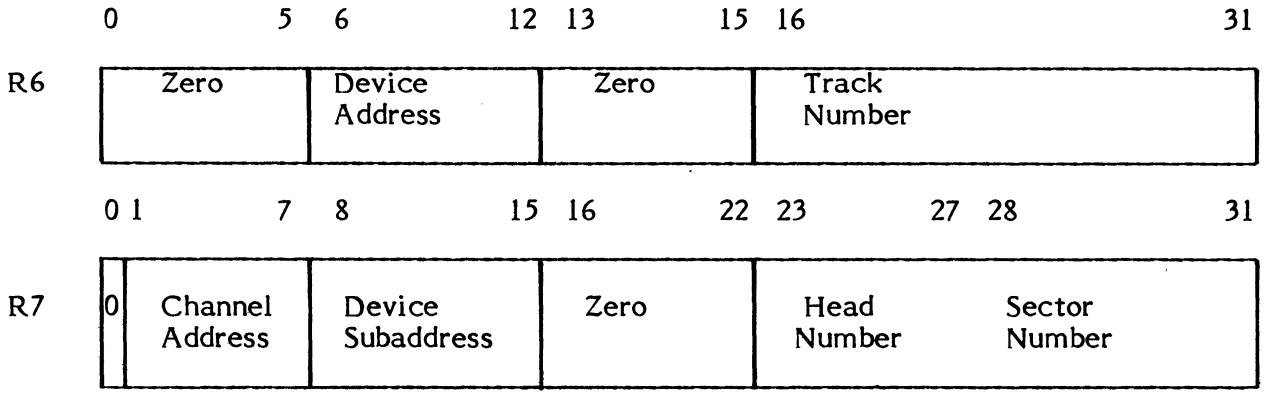
R6	bit 0	1 indicating that the specified name is a memory partition
	bits 1-31	number of 512 word pages allocated to the partition logical
R7		address of the first word of the specified partition

Case III, Disc File:

Return Sequence: M.RTRN 6,7

Registers:

R6 and R7 are returned with the address of the beginning of the disc file as follows:



For disc files, the following additional parameters are returned.

Condition code 1 is set in the program status word if a password is required to write the file (read-only file).

Condition code 2 is set if a password is required to read or write the file (password-only file).

Condition code 3 is set if the file is a system file or core partition.

External References

System Macros:

```
M.CALL H.FISE,10
M.RTRN
```

Abort Cases: None

Output Messages: None

7.8.9 M.FILE - Open File

The M.FILE service performs the following functions:

Establishes appropriate linkages between a user FCB and an assigned file or device.

Marks a file open for either read-write or read-only operations. Increments internal counts of tasks having the file open at this time.

For SYC or SGO files, completes building the FAT based on Job Control information.

For system and blocked files, initializes blocking buffer for subsequent access.

Requests the initial MOUNT message for statically allocated, non-shared magnetic tape devices.

Note: OPEN requests to a file which is already OPEN are ignored.

Entry Conditions

Calling Sequence:

```
M.FILE fcb,[RW]

(or)

LA    1,fcb
[SBR  1,1 if RW]
SVC   1,X'30' (or) M.CALL H.IOCS,1
```

where:

fcb — is the FCB address

RW — is the optional character string RW which specifies read-write. (If the task has read but not write access to a permanent disc file, the file is opened read-only.)

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO17 No logical file code assigned which matches FCB file code

7.8.10 M.FSLR - Release Synchronization File Lock

The M.FSLR service is used for disc file gating. It is implemented in conjunction with the set synchronization File Lock service (M.FSLS) to control a synchronization lock indicator. When M.FSLR is called, the synchronization lock is released, and the queue of tasks waiting to own the lock will be polled.

Entry Conditions

Calling Sequence:

```
M.FSLR   lfca

(or)

LW       R5,lfca
SVC      1,X'24' (or) M.CALL   H.FISE,25
```

where:

lfca — is the address of a word that contains an unused byte in byte 0, and a one to three ASCII character, left justified, blank filled logical file code in bytes 1, 2, and 3.

Exit Conditions

Return Sequence: M.RTRN R7

Registers:

R7
- = 0, Request accepted, synchronization lock released.
- = 1, Request denied, synchronization lock was not set.
- = 5, Request denied, specified lfc not allocated.
- = 6, Request denied, specified lfc is not assigned to a permanent disc file.

Notes:

1. A synchronization lock may not be cleared by a task other than the task which set the lock.

2. If a task owns a synchronization lock when the task terminates, the lock is automatically released.

3. A synchronization lock is automatically released when the file is deallocated.

7.8.11 M.FSLS - Set Synchronization File Lock

The M.FSLS service is used in conjunction with the Release Synchronization File Lock service (M.FSLR) for disc file gating. The M.FSLS and M.FSLR services control a synchronization lock indicator which allows synchronized access to a disc file that is concurrently allocated to multiple tasks. To use the M.FSLS service, the file must have been previously allocated to the calling task. The file is identified by logical file code (lfc).

Entry Conditions

Calling Sequence:

```
M.FSLS   lfca[,timev]

(or)

LW       R5,lfca
LI       R4,timev  (or)  ZR      R4
SVC      1,X'23'   (or)  M.CALL  H.FISE,24
```

where:

lfca — is the address of a word that contains an unused byte in byte 0, and a one to three ASCII character, left justified, blank filled logical file code in bytes 1, 2, and 3.

timev — is a numeric value interpreted as follows:

- +1 = return immediately with a denial code if the file already has a synchronization lock set.
- 0 = place the requesting task in a wait state until it has become the owner of the synchronization lock.
- -n = place the requesting task in a wait state until it owns the synchronization lock, or until the expiration of n timer units, whichever occurs first.

Exit Conditions

Return Sequence: M.RTRN R7

Registers:

R7
- = 0, Request accepted, synchronization lock set.
- = 1, Request denied, synchronization lock is already owned by another task.
- = 2, Request denied, time-out occurred while waiting to become lock owner.
- = 3, Reserved.
- = 4, Reserved.
- = 5, Request denied, lfc not allocated.
- = 6, Request denied, lfc not assigned to permanent disc file.

7.8.12 M.FWRD - Advance Record or File

The M.FWRD service is not applicable for system files (i.e., SYC, SGO, SLO, SBO). It performs the following functions for advance record:

Verifies volume record if BOT on multi-volume magnetic tape.

Advances specified number of records.

M.FWRD performs the following functions for advance file:

If a blocked file, logical records are advanced until an end-of-file is found. The read/write control word will point to the first record after the end-of-file.

Verifies volume record if BOT on multi-volume magnetic tape.

Advances specified number of files.

Entry Conditions

Calling Sequence:

```
M.FWRD  fcb,[R],number

(or)

LA      1,fcb
LNW     4,number
{SVC    1,X'33' or M.CALL H.IOCS,7}
{SVC    1,X'34' or M.CALL H.IOCS,8}
BIB     4,$-1W
```

where:

fcb	FCB address
number	address of word containing the number of records or files to be advanced
R	advance by record (SVC1,X'33'). Default: advance by file (SVC1,X'34').
$-1W	branches back to SVC until reaches last word of Register 4

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO06	Invalid blocking buffer control cells for blocked file
IO13	Illegal operation for system file
IO28	Advance record issued for a blocked file while writing
IO29	Advance file issued for a blocked file while writing
IO30	Either volume number or reel id from volume record do not match FAT information

Output Messages:

MOUNT/DISMOUNT messages if EOT on multi-volume magnetic tape

7.8.13 M.FXLR - Release Exclusive File Lock

The M.FXLR service is used in conjunction with the Set Exclusive File Lock service (M.FXLS) for disc file gating. When M.FXLR is called, the exclusive lock is released and other tasks can allocate the associated disc file. Note that another task will not be able to exclusively lock the file, however, until it is deallocated by this task. For further description of file gating, see Volume 1, Section 2.

Entry Conditions

Calling Sequence:

```
M.FXLR  lfca

(or)

LW      R5,lfca
SVC     1,X'22'  (or)  M.CALL   H.FISE,23
```

where:

lfca is the address of a word that contains an unused byte in byte 0, and a one to three ASCII character, left justified, blank filled logical file code in bytes 1, 2, and 3.

Exit Conditions

Return Sequence: M.RTRN R7

Registers:

R7
- = 0, Request accepted, exclusive file lock released.
- = 1, Request denied, an exclusive file lock was not owned by this task.
- = 5, Request denied, specified lfc is not allocated.
- = 6, Request denied, specified lfc is not assigned to a permanent disc file.

Notes:

1. An exclusive file lock may not be released by a task other than the owning task.

2. Any outstanding exclusive file locks are released on task termination or on file deallocation.

7.8.14 M.FXLS - Set Exclusive File Lock

The M.FXLS service is used for disc file gating. It allows the calling task to gain exclusive allocation of a file, as though it were an unshared resource. The file must have been previously allocated, and is identified by the address of logical file code (lfc). For further description of file gating, see Volume 1, Chapter 2.

Entry Conditions

Calling Sequence:

```
M.FXLS   lfca[,timev]

(or)

LW       R5,lfca
LI       R4,timev  (or)  ZR      R4
SVC      1,X'21'   (or)  M.CALL  H.FISE,22
```

where:

lfca — is the address of a word that contains an unused byte in byte 0, and a one to three ASCII character, left justified, blank filled logical file code in bytes 1, 2, and 3.

timev — is a numeric value interpreted as follows:

- +1 = return immediately with a denial code if the file is already allocated to another task.
- 0 = place the requesting task in a wait state until the designated file can be exclusively locked.
- -n = place the requesting task in a wait state until the designated file can be exclusively locked, or until the expiration of n timer units, whichever occurs first.

Exit Conditions

Return Sequence: M.RTRN R7

Registers:

R7
- = 0, Request accepted, file is exclusively locked.
- = 1, Request denied, file is allocated to another task, or is already exclusively locked.
- = 2, Reserved.
- = 3, Reserved.
- = 4, Request denied, time-out occurred while waiting to become lock owner.
- = 5, Request denied, lfc not allocated.
- = 6, Request denied, lfc not assigned to permanent file.

7.8.15 M.LOG - Permanent File Log

The M.LOG service provides a log of currently existing permanent files.

Entry Conditions

Calling Sequence:

```
M.LOG   [type],address[,filename]

(or)

LI      R4,type
LA      R5,address
LD      R6,filename (if TYPE =N or O)
SVC     1,X'73' (or) M.CALL  H.MONS,33
```

where:

type — is a byte-scaled value which specifies the type of log to be performed as follows:

="N"=0	specifies a single named system or user file
="A"=1	specifies all permanent files
="S"=2	specifies system files only
="U"=3	specifies user files
="O"=4	specifies a single named system file

If TYPE = "N" and a user name is associated with the calling task 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 task, the file is assumed to be a system file.

If TYPE = "U" and the calling task has an associated user name, that user's files are logged. All files are logged if the calling task has no associated user name.

Note: This service logs one SMD entry per call. Types 1, 2, and 3 are usually used to search through a set of user or system files. A standard call specifying the type is used to get the first file in the set. To get the rest of the files in the set, a call in the form M.LOG ,address is made repeatedly until the desired number of files is logged or until R5 is returned as zero, signifying all files in the set have been logged.

address — is the address of an eight-word area within the calling task where the file System Master Directory entry is to be stored.

filename — contains a one to eight-character file name if TYPE = "N" or "O".

Exit Conditions

Return Sequence: M.RTRN 4,5

The eight-word SMD entry, if any, is stored at the address specified as "address". The password field contains zero or one to indicate the absence or presence of a password respectively.

Registers:

R4	If type = "N" or "O" (R4=0 or 4), R4 is destroyed. If type = "A", "S" or "U" (R4=1,2 or 3), this service 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 type = "N" or "O" (R4=0 or 4) and the specified file could not be located or type = "A", "S" or "U" (R4=1,2 or 3) and all pertinent files have been logged. Otherwise, R5 is unchanged.

External References

System Macros:

M.CALL
M.RTRN

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

7.8.16 M.PDEV - Physical Device Inquiry

The M.PDEV service returns (to the caller) physical device information describing the unit to which a specified logical file code is assigned.

Entry Conditions

Calling Sequence:

```
M.PDEV   lfc

(or)

LW       R5,lfc
SVC      1,X'42'  (or)  M.CALL  H.MONS,1
```

where:

lfc — contains a one to three ASCII character, left-justified, blank filled logical file code in bytes 1, 2, and 3 for which physical device information is requested.

Exit Conditions

Return Sequence: M.RTRN 7

Registers:

R7 — zero, if the specified logical file code is unassigned

(or)

Return Sequence:

```
M.RTRN   4,5,6,7
```

Registers:

R4 bit 0	1 for Extended I/O (Class F) device 0 for all other device classes
bits 1-7	0
byte 1	0
bytes 2,3	device (2 ASCII characters), e.g., MT, DC, etc. See Appendix A.
R5	if disc = number of 192-word blocks in file if magnetic tape = reel identifier (4 ASCII characters) if TSM terminal: byte 2 = number of hexadecimal characters in line byte 3 = number of hexadecimal lines on screen all other devices = 0

R6 bytes 0,1	maximum number of bytes transferrable to device
byte 2	device channel number
byte 3	device subaddress
R7 byte 0	device type code (2 hex digits) See Appendix A.
byte 1	
bit 8	0 if file is unblocked, 1 if file is blocked
bits 9-12	0
bits 13-15	system file code, as follows:

0	not a system file
1	SYC file
2	SGO file
3	SLO file
4	SBO file

bytes 2,3	if disc = number of 192-word sectors per allocation unit if magnetic tape = volume number (0=single volume) all other devices = 0

Note: If the specified logical file code is assigned to SYC or SGO, and that file is not open, bits 13 through 15 of R7 are returned equal to 1 or 2. All other returned parameters are not applicable.

When doing a physical device inquiry while running from the console terminal, R7 can be returned 0 even though the lfc is assigned. When this occurs, the device type code 00 (console terminal) is in register 7.

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

7.8.17 M.PERM - Change Temporary File to Permanent

The M.PERM service 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:

```
M.PERM    filename,lfc,   [{R},password],
                          [{P}         ]
                          [S],[N],[F],[type],[Z]
```

(or)

```
LD                        R6,filename
LW                        R2,=G'lfc'
ZR                        R3
[SBR      R3,6            if R - read only
 SBR      R3,7            if P - password only
 SBR      R3,2            if N - not SAVE DEVICE file
 SBR      R3,3            if F - FAST file              ]
ZR                        R1
[LI                       R1,X'type' if type present
 SBR      R1,0            if S - system file
 SBR      R1,1            if Z - pre zero               ]
ZR                        R4
[LD                       R4,password if file is to have a password]
SVC                       1,X'76' (or) M.CALL  H.FISE,13
```

where:

filename — is a doubleword containing the one to eight ASCII character, left-justified, blank filled name of the file. Each character in the name must have an ASCII equivalent in the range 21 through 5F (printable) and may not contain a comma.

lfc — is the one to three ASCII character, left-justified, blank filled logical file code assigned to an open, temporary, SLO or SBO file. The file is marked as permanent in the calling task by this service if successful.

R,P — optional character to indicate file access restrictions as follows:

R- Read only (password required to write)
P- Password only (password required to read or write)

If this parameter is not specified, the file may be read or written without a password.

password — is a doubleword containing the optional one to eight ASCII character, left-justified, blank filled password. Each character must have an ASCII equivalent in the range 01 through 7F (printable, lowercase, or special). The password may not contain a comma. If a file access restriction (R or P parameter) is specified, a password must be entered. If no access restriction is specified, a password is optional. If a password is entered, the file can be deleted only by specifying the password. In addition, the password must be entered on SAVE FILE, DELETE, and EXPAND File Manager directives. R4 is zero if no password is to be associated with the file.

S — is an optional character to indicate that the file is to be a system file. If not specified, the file is created as a user file if a user name is associated with the calling task, or as a system file if no user name is associated with the calling task.

N — is an optional character to indicate that the file is not to be saved in response to the SAVE DEVICE File Manager directive.

F — is an optional character to indicate that the file is a FAST file. If no parameter is entered, the file is created as a SLOW file.

type — is an optional parameter which is a one or two digit hexadecimal value specified by the user to identify the origin of the file as follows. For example, the system software utilizes the following file types:

ED - Editor Save File
EE - Editor Store File
FE - Editor Work File
FF - SYSGEN Created File
BA - BASIC File
CA - Cataloged Load Module

Z — is an optional character to indicate that the space allocated to the file is to be zeroed.

Exit Conditions

Return Sequence: M.RTRN 6,7

Registers:

R7 — zero if the file was not created. R6 contains the reason as follows:

R6
= 1 if a file of the name specified already exists
= 2 if a FAST file was specified and collision mapping occurred with an existing directory entry
= 3 if restricted access (bit 6 or 7 or R2) was specified, but no password (R4,R5) was entered.
= 4 if the file associated with the specified logical file code is not an open, temporary, SLO or SBO file.
= 7 if the SMD is full
= 9 if the file name or password contains invalid characters or imbedded blanks.

External References

System Macros:

```
M.CALL   H.FISE,1; H.FISE,2; H.FISE,8; H.FISE,10;
M.RTRN   H.MONS,20; H.IOCS,4
```

System Subroutines:

```
S.ALOC12
S.FISE3
```

Abort Cases:

FS01	Unrecoverable I/O error to the System Master Directory (SMD).
FS02	Unrecoverable I/O error to a disc allocation map.

Output Messages: None

7.8.18 M.READ - Read Record

The M.READ service performs the following functions:

Provides special random access handling for disc files.

Deblocks system files and blocked files.

Reads one record into the buffer indicated by the Transfer Control Word (TCW) in the FCB.

Entry Conditions

Calling Sequence:

```
M.READ   fcb

(or)

LA       1,fcb
SVC      1,X'31'  (or)  M.CALL H.IOCS,3
```

where:

fcb is the FCB address. Appropriate transfer control parameters are defined in the TCW.

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO03	Non-privileged user attempting transfer to a logical address outside legal boundaries.
IO06	Invalid blocking buffer control cell for a system or blocked file.
IO26	Read attempted for a system or blocked file while write in process.
IO30	Illegal volume record. Either volume number or reel ID from volume record do not match FAT information.
IO32	Second attempt to read a $ statement in an SYC file.

Output Messages:

DISMOUNT/MOUNT messages if EOT and multi-volume magnetic tape

7.8.19 M.RELP - Release Dual Ported Disc

The M.RELP service applies only to dual ported extended I/O disc and allows the privileged user to release a device from its reserved state.

Entry Conditions

Calling Sequence:

```
M.RELP

(or)

LA      1,fcb
SVC     1,X'27'  (or)  M.CALL  H.IOCS,27
```

where:

fcb FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases: None

Output Messages: None

7.8.20 M.RESP - Reserve Dual Ported Disc

The M.RESP service applies only to dual ported extended I/O disc and allows the privileged user to reserve a device to the requesting CPU until such time as a release (M.RELP) issued.

Entry Conditions

Calling Sequence:

```
M.RESP

(or)

LA      1,fcb
SVC     1,X'26'  (or)  M.CALL  H.IOCS,24
```

where:

fcb FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases: None

Output Messages: None

7.8.21 M.RRES - Release Channel Reservation

If the specified channel has not been reserved by this task, the M.RRES service ignores the request to release the channel and returns to the task. If the channel has been reserved by this task, the channel reserve indication is removed from the CDT entry.

After releasing the reserved channel, if any requests had been queued while the channel was reserved, IOCS resumes I/O to the associated device.

This service is not applicable for extended I/O channels.

Entry Conditions

Calling Sequence:

```
M.RRES  channel

(or)

LW      1,channel
SVC     1,X'3B' (or) M.CALL  H.IOCS,13
```

where:

channel — specifies the channel number (hexadecimal). If LW, load bits 24-31 of Register 1.

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases: None

Output Messages: None

7.8.22 M.RSML - Resourcemark Lock

The M.RSML service is called to lock the specified resourcemark. It is used in conjunction with the unlock resourcemark service (M.RSMU) by tasks to synchronize access to a common resource. For further description, see Section 2.9.1.5.

Entry Conditions

Calling Sequence:

```
M.RSML  lockid , [timev][,P]

(or)

LI      R4,timev
ZR      R5
[SBR    R5,0]
LI      R6,lockid
SVC     1,X'19' (or) M.CALL  H.MONS,62
```

where:

lockid — is the numeric resourcemark index

timev — is a numeric value which specifies action to be taken if the lock is already set and is owned by another task:

+1	=	immediate denial return
0	=	wait until this task is the lock owner
-n	=	wait until this task is the lock owner, or until n timer units have expired, whichever occurs first

Default: 0. Wait until this task is the lock owner.

P — indicates that while this task is waiting to become lock owner, the swapping mode is to be set to swap this task only if a higher priority task is requesting memory space. Otherwise, the task will be a swap candidate if any task is requesting memory.

Exit Conditions

Return Sequence: M.RTRNR7

Registers:

R7 — zero if the request was accepted, otherwise contains a request denial code:

1= lock index exceeds maximum range
2= lock index is less than minimum range
3= lock is owned by another task (and timev=+1)
4= lock is owned by another task, timev=-n and n timer units have elapsed

Abort Cases: None

Output Messages: None

7.8.23 M.RSMU - Resourcemark Unlock

The M.RSMU service is called to unlock a resourcemark which has previously been locked by a call to the M.RSML service. If any other tasks are waiting to lock the specified resourcemark, the highest priority waiting task will become the new lock owner.

Entry Conditions

Calling Sequence:

```
M.RSMU lockid

(or)

LI     R6,lockid
SVC    1,X'1A'  (or)  M.CALL  H.MONS,63
```

where:

lockid — is the numeric resourcemark index

Exit Conditions

Return Sequence:

```
M.RTRN R7
```

Registers:

R7	zero if the request was accepted, otherwise contains a request denial code: 1=lock index exceeds maximum range 2=lock index is less than minimum range 3=lock is not owned by this task

Abort Cases: None

Output Messages: None

7.8.24 M.RSRV - Reserve Channel

M.RSRV is a privileged service. If the task is unprivileged or the channel has been reserved previously by another task, this service makes a denial return. If the channel has not previously been reserved, the task number is stored in the CDT entry to mark the reservation. If any requests are currently queued for this channel, suspend is invoked until completion of any I/O currently in progress is complete. The standard handler is then disconnected from the Service Interrupt (SI) level. After reserving a channel, the task must connect its own handler to the SI dedicated location.

This service is not applicable for extended I/O channels.

Entry Conditions

Calling Sequence:

```
M.RSRV  channel,denial

(or)

LW      1,channel
LA      7,denial
SVC     1,X'3A'  (or)  M.CALL  H.IOCS,12
```

where:

channel	specifies the channel number (hexadecimal) in bits 24-32. If using LW, load channel number in Register 1.
denial	is the user's denial return address

Exit Conditions

Return Sequence:

M.RTRN normal return

(or)

M.RTNA 7 denial return

Registers:

Normal - None
Denial - None

Abort Cases:

IO14 Unprivileged user attempting to reserve channel

Output Messages: None

7.8.25 M.RWND - Rewind File

The M.RWND service performs the following functions:

Issues an end-of-file and purge if the file is a system or blocked file which is output active.

For system and blocked files, initializes blocking buffer control cells for subsequent access.

Rewinds file or device.

Entry Conditions

Calling Sequence:

```
M.RWND fcb

(or)

LA      1,fcb
SVC     1,X'37' (or) M.CALL  H.IOCS,2
```

where:

fcb is the FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO09 Rewind attempted on SYC file

7.8.26 M.TYPE - OPCOM Console Type

The M.TYPE service types a user specified message and performs an optional read on the OPCOM console. Input message address validation is performed for the unprivileged task. Operation is wait I/O.

The maximum input or output is 80 characters.

M.TYPE builds a Type Control Parameter Block (TCPB) which defines input and output buffer addresses for console messages and reads as described in Section 7.7.

Entry Conditions

Calling Sequence:

```
M.TYPE  outmess,outcount[,inmess,incount]

(or)

LA      1,tcpb
SVC     1,X'3F' (or) M.CALL H.IOCS,14
```

where:

outmess specifies address of output message buffer

outcount specifies transfer count for output (number of bytes, 80 maximum). If not specified (transfer count is zero), defaults to maximum.

inmess specifies address of input message buffer. If not specified, TCPB Word 2 is zeroed.

incount transfer count for input (number of bytes, 80 maximum). Incount also takes into account the <CR> character typed at the end of the input line.

tcpb specifies address of Type Control Parameter Block (TCPB). See Section 7.7.

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO03	Non-privileged user attempting transfer to a logical address outside legal boundaries
IO15	Type request while operation in progress
IO25	Transfer count of zero

Output Messages: None

7.8.27 M.UPSP - Upspace

The M.UPSP service is not applicable to blocked or system files (i.e., SYC, SGO, SLO, SBO). If BOT is present on multi-volume magnetic tape, volume record (header) is written. If EOT is present on multi-volume magnetic tape, ERASE/WRITE EOF is performed.

Entry Conditions

Calling Sequence:

```
M.UPSP  fcb

(or)

SVC     1,X'10' (or) M.CALL H.IOCS,20
```

where:

fcb is the FCB address

Registers:

R1 FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO13	The user has requested an illegal operation to be performed on a system file.

Output Messages: MOUNT/DISMOUNT messages if EOT on multi-volume magnetic tape

7.8.28 M.USER - Username Specification

The M.USER service associates a user name with the calling task. Optionally, this service nullifies any user name associated with the calling task. The user name associated with the task is utilized in file create, delete, log, and allocate services called subsequently.

Entry Conditions

Calling Sequence:

```
M.USER  [username [,key] ]

(or)

ZR      R6 null username
ZR      R7
SVC     1,X'74' (or) M.CALL  H.MONS,34

(or)

LD      R6,username
LD      R4,key
SVC     1,X'74' (or) M.CALL  H.MONS,34
```

where:

username — contains the one to eight-character user name left justified and blank filled. Each character must have an ASCII equivalent in the range 01 through 7F.

To nullify any user name associated with the calling task, both parameters are omitted.

key — contains the one to eight-character left justified and blank filled user key, if any, associated with the user name.

Exit Conditions

Return Sequence: M.RTRN 6,7

Registers:

R6,R7 — zero if the service was not performed because the specified user name contains invalid characters or is not in the user name file or the required key was not furnished. Otherwise unchanged.

CC1 — set if the service was not performed

External References

System Macros: M.CALL, M.RTRN

Abort Cases:

MS28 Unrecoverable I/O error to disc.

Output Messages: None

7.8.29 M.WAIT - Wait I/O

The M.WAIT service provides return to the user when the I/O request associated with the specified FCB is complete; however if the FCB has no I/O outstanding on it, then a return to the user is made instead of linking the task to the Wait state queue. Once the task is linked to the queue, it is suspended until I/O completes.

Entry Conditions

Calling Sequence:

```
M.WAIT  fcb

(or)

LA      1,fcb
SVC     1,X'3C'  (or)  M.CALL  H.IOCS,25
```

where:

fcb is the FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases: MS31 User attempted to go to an any wait state from an end action routine.

Output Messages: None

7.8.30 M.WEOF - Write EOF

The M.WEOF service performs the following functions:

Prevents a write to a read-only file.

Issues an end-of-file and purge if the file is a system or blocked file which is output active. (If EOF is requested for an unblocked disc file, the handler ignores the request.)

Writes volume record if BOT on multi-volume magnetic tape.

Performs ERASE/WRITE EOF if EOT on multi-volume magnetic tape.

Writes one EOF.

Entry Conditions

Calling sequence:

```
M.WEOF    fcb

(or)

LA        1,fcb
SVC       1,X'38' (or) M.CALL  H.IOCS,5
```

where:

fcb is the FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO10	FAT entry marked read-only (unless SGO).
IO11	File is SYC.
IO30	Illegal volume record. Either volume number or reel ID from volume record do not match FAT information.

Output Messages:

DISMOUNT/MOUNT messages if EOT on multi-volume magnetic tape

7.8.31 M.WRIT - Write Record

The M.WRIT service performs the following functions:

Prevents a write to a read-only file.

Provides special random access handling for disc files.

Blocks records for system and blocked files.

Writes volume record if BOT of multi-volume magnetic tape.

Performs ERASE/WRITE EOF if EOT of multi-volume magnetic tape.

Writes one record from the buffer pointed to by the TCW in the FCB.

Entry Conditions

Calling Sequence:

```
M.WRIT    fcb

(or)

LA        1,fcb
SVC       1,X'32'  (or)  M.CALL  H.IOCS,4
```

where:

fcb is the FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO06	Invalid blocking buffer control cell for a system or a blocked file.
IO27	Write attempted while reading from a system or a blocked file.
IO53	Write attempted to SYC file in batch mode.

Output Messages:

DISMOUNT/MOUNT messages if EOT on multi-volume magnetic tape

7.8.32 M.XIEA - No-Wait I/O End Action Return

The M.XIEA service is required for exiting any no-wait I/O end action routine (both normal and error end action routines use this exit).

Entry Conditions

Calling Sequence:

```
        M.XIEA

        (or)

        SVC     1,X'2C' or  M.CALL  H.IOCS,34
```

Exit Conditions

Return Sequence:

```
        BL      S.EXEC6  no-wait I/O post processing complete
```

Registers: None

Abort Cases: None

Output Messages: None

7.8.33 Erase or Punch Trailer

The Erase or Punch Trailer service writes the volume record if BOT on multi-volume magnetic tape or performs ERASE/WRITE EOF if EOT on multi-volume magnetic tape.

Erase, punch trailer is not applicable to blocked or system files (i.e., SYC, SGO, SLO, SBO).

Entry Conditions

Calling Sequence:

```
        LA      1,fcb
        SVC     1,X'3E' or  M.CALL  H.IOCS,21
```

where:

fcb FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO12	File not opened for write mode.
IO13	Illegal operation for system file.

Output Messages:

MOUNT/DISMOUNT messages if EOT on multi-volume magnetic tape

7.8.34 Execute Channel Program

The Execute Channel Program service allows command and data chaining to General Purpose Multiplexor Controller (GPMC) and extended I/O devices only.

Entry Conditions

Calling Sequence:

```
LA      1,fcb
SVC     1,X'25' or M.CALL  H.IOCS,10
```

where:

fcb FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases:

IO03	Non-privileged user attempting transfer to a logical address outside legal boundaries.
IO43	IOCL list or data address not in contiguous E-memory (GPMC devices only).

7.8.35 Release FHD Port

The Release FHD Port service is available only to privileged users and is currently only supported by the four megabyte fixed head disc.

Entry Conditions

Calling Sequence:

```
        LA      1,fcb
        SVC     1,X'27' or M.CALL  H.IOCS,27
```

where:

fcb FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases: None

Output Messages: None

7.8.36 Reserve FHD Port

The Reserve FHD Port service is available only to privileged tasks and is currently only supported by the four megabyte fixed head disc.

Entry Conditions

Calling Sequence:

```
        LA      1,fcb
        SVC     1,X'26' or M.CALL  H.IOCS,24
```

where:

fcb FCB address

Exit Conditions

Return Sequence: M.RTRN

Registers: None

Abort Cases: None

Output Messages: None

8. SYSTEM SERVICES

MPX-32 offers a set of resident system service routines designed to perform frequently required operations with maximum efficiency. Using the CALM or the Supervisor Call instruction, tasks running in any environment can call these routines.

All of the system service routines are reentrant. Thus, each service routine is always available to the task which is currently active.

System service routines are provided as standard modular components of the Mapped Programming Executive. The "open-ended" design of the system, however, gives each user freedom to add whatever service routines are required to tailor MPX-32 to a specific application.

System services enable tasks to:

Activate, suspend, resume, abort, terminate and hold task execution

Change a task's priority level

Create, test, and delete timers

Interrogate system clocks

Allocate and deallocate devices and files

Obtain the characteristics of a device or file

Communicate with other tasks via messages and status words

Load and execute overlays

Obtain information on the memory assigned to a task

Connect tasks to interrupts

Interrogate the arithmetic exception and option word status for task

MPX-32 services are implemented as SVC traps. There are several ways of accessing services:

By macro calls, with parameter passing as indicated. The expansion code in the system macro library is then accessed automatically during assembly to provide Assembly language setup of appropriate registers and instructions including SVC's, in the user's code.

By setting up appropriate registers and instructions directly and using appropriate SVC's.

By following the course above but issuing an M.CALL request to the entry point of the system module that provides the service.

The first two access paths are described for each system service in this section and other sections where services are documented. The third access path is privileged, and is indicated primarily to provide the appropriate system module names and entry point numbers for cross-reference to other documentation when needed.

Documentation conventions used in service syntax descriptions are the standard conventions described at the beginning of this book.

Recognizing that the user may need to get back to documentation on a particular service either from an SVC number or alphabetically by a short descriptor (like RTM) or by macro name, cross reference charts are provided for all system services in Appendix B. The services in this section are organized alphabetically by macro name under two broad categories:

Task Execution Services

Memory Management Services

File and device allocation and I/O as well as file management services are organized alphabetically in Chapter 7. Services for interactive tasks are described in Chapter 5.

8.1 RTM System Services Under MPX-32

MPX-32 will accept call monitor (CALM) instructions which are syntactically and functionally equivalent to the RTM CALM's. These CALM's are implemented for compatibility purposes, only. The user is encouraged to use the new MPX-32 system services, instead of CALM's, because they will run faster and support the new capabilities available in MPX-32. Mixing CALM's and SVC's in the same program element is not encouraged.

Generally, RTM CALM's will 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.

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) M.DDJS or CALL M:UNLKJ

CALM X'63' Activate with Core Append (replaced by memory expansion and contraction services of MPX) 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

CALM X'59' Random Access OPEN (MPX-32 does not support DRAH)
(CALL M:OPEN)

CALM X'5A' Random Access READ (same as CALM X'59')

CALM X'5B' Write Function (same as CALM X'59')
(CALL M:WRITE)

CALM X'5C' DEFINE FUNCTION
(CALL M:DEFINE)

CALM X'5D' FIND FUNCTION
(CALL M:DEFINE)

TSS CALM'S

MPX-32 replaces TSS with TSM, a new online support package. Therefore, all TSS CALM's X'80' - X'84' have been deleted.

On a CONCEPT/32 computer, a new 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 will be generated when a doubleword operation code is used with an incorrectly bounded operand, therefore coding changes will be required when a trap occurs.

8.2 Task Execution Services

8.2.1 M.ACTV - Activate Task

The M.ACTV service is used to activate a task. The task assumes the owner name of the caller.

Entry Conditions

Calling Sequence:

```
M.ACTV    loadmod

(or)

LD        R6,loadmod
SVC       1,X'52'  (or)  M.CALL   H.MONS,15
```

where:

loadmod — is a doubleword containing the load module name for which an activation request is to be queued, one to eight ASCII characters, left-justified and blank filled. The specified load module must exist as a system file and must not be password only (PO) protected.

Exit Conditions

Return Sequence:

M.RTRN 6,7

Registers:

R6 — zero if the service could be performed

R7 — contains the task number of task activated by this service

(or)

R6 = 1 — if invalid attempt to multicopy a unique load module

R7 — task number of existing task with same name

(or)

R6
- = 2 if load module file not in SMD
- = 3 if load module file is PO password protected
- = 4 if file does not contain valid data
- = 5 if no DQE is available
- = 6 if read error on SMD
- = 7 if read error on load module
- = 8 insufficient memory
- = 10 no physical memory available

External References

System Macros:

M.RTRN

Abort Codes:

None

Output Messages:

None

8.2.2 M.ANYW - Wait for Any No-Wait Operation Complete, Message Interrupt, or Break Interrupt

The M.ANYW service is called to place the currently executing task in a state waiting for the completion of any no-wait request, for the receipt of a message, or or a break interrupt. The task is removed from the associated ready to run list, and placed in the any-wait list. A return is made to the program location following the SVC instruction only when one of the wait conditions has been satisfied or when the optional time-out value has expired.

Entry Conditions

Calling Sequence:

```
M.ANYW time1

(or)

LW      R6,time1
SVC     1,X'7C' (or) M.CALL  H.MONS,37
```

where:

time1 contains zero if wait for an indefinite period is requested. Otherwise, time1 contains the negative number of time units to elapse before the wait is terminated. (The actual time elapsed can vary by one time unit because of system design.)

Exit Conditions

Return Sequence:

M.RTRN

Registers:

None

External References

System Macros:

M.RTRN

Abort Cases:

MS31 User attempted to go to an any wait state from an end action routine.

Output Messages:

None

8.2.3 M.ASYNCH - Set Asynchronous Task Interrupt

The M.ASYNCH service resets the asynchronous task interrupt mode back to the default environment.

Entry Conditions

Calling Sequence:

M.ASYNCH

(or)

SVC 1,X'1C' (or) M.CALL H.MONS,68

Exit Conditions

Return Sequence:

M.RTRN

Registers:

CC1 set if asynchronous task interrupt already set

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.4 M.BORT - Abort Specified Task, Abort Self, or Abort with Extended Message

M.BORT - Specified Task

This service allows the caller to abort another task. If the named task has been swapped out, it will not be aborted until it regains CPU control. When the named task receives CPU control, it will be aborted (see Section 2.7, Table 2-2). If the specified task is not in execution, the request is ignored.

Entry Conditions

Calling Sequence:

```
M.BORT   abcode,task

(or)

LW       R5,abcode
ZR       R6              (or) LD R6,taskname
LW       R7,taskno
SVC      1,X'56' (or) M.CALL  H.MONS,19
```

where:

abcode — contains the abort code consisting of four ASCII characters

task — the address of a doubleword containing the name of the task or 0 in Word 0 and the task number in word 1. Task number must be used if the task is multicopied or shared.

Exit Conditions

Return Sequence:

```
M.RTRN   7
```

Registers:

R7 — zero if any of the following conditions exist:

- the specified task number or task name was not found in the dispatch queue
- the specified task name was not single copied
- the owner name of the task requesting the abort is restricted from access to tasks with a different owner name (via the M.KEY file), the task requesting the abort is not privileged, and the owner names of the requesting and target task do not match
- the task is in the process of exiting the system

Otherwise, task number.

CC1 — set if the task is in the process of exiting the system.

External References

System Macros:

M.RTRN	M.CALL
M.IOFF	M.IONN
M.OPEN	

Abort Cases:

None

Output Messages:

None

M.BORT - Self

This service aborts the calling task by issuing an abort message, optionally performing a post-mortem dump, and performing the functions common to the normal termination service as described in Chapter 2.

Entry Conditions

```
M.BORT    abcode

(or)

LW        R5,abcode
SVC       1,X'57'  (or)  M.CALL   H.MONS,20
```

where:

abcode	contains the abort code consisting of four ASCII characters.

Exit Conditions

Return Sequence:

M.RTRN

Registers:

None

External References

System Macros:

M.CALL

Abort Cases:

None

Output Messages:

task number ABORTED. PSW:xxxxxxxx BIAS:yyyyy REASON:zzzz

where:

task	is the one to eight character task load module name of the task being aborted.
number	is the task number of the task being aborted
xxxxxxxx	is the location the abort occurred
yyyy	is the beginning of the DSECT
zzzz	is the four character abort code

M.BORT - With Extended Message

A call to this service will result in an abort of the specified task. An additional eight characters are displayed in the abort message.

Entry Conditions

Calling Sequence:

```
M.BORT   abcode,task,extcode

(or)

LD       R2,extcode
LW       R5,abcode
LI       R6,0      }          (or) LD R6,taskname
LW       R7,taskno }
SVC      1,X'62'  (or)  M.CALL   H.MONS,28
```

where:

abcode	contains the abort code consisting of four ASCII characters.
task	the address of a doubleword containing the name of the task or 0 in Word 0 and the task number in word 1. Task number must be used if the task is multicopied or shared. A task number of 0 specifies the calling task.
extcode	contains the extended abort code message consisting of one to eight ASCII characters, left justified, and blank filled.

Exit Conditions

Return Sequence:

```
M.RTRN   7
```

Registers:

R7	zero if the specified task was not found in the Dispatch Queue, or the task is scheduled to leave the system. Otherwise contains the task number.
CC1	set if the task was scheduled to leave the system.

External References

System Macros:

M.RTRN	M.CALL
M.IOFF	M.IONN
M.OPEN	

Abort Cases:

None

Output Messages:

Same as Abort Self

8.2.5 M.BRK - Break/Task Interrupt Link

The M.BRK service allows the caller to establish the address of a routine to be entered whenever another task or the operator activates his task interrupt via an M.INT service.

Entry Conditions

Calling Sequence:

```
M.BRK    brkadd

(or)

LA       R7,brkadd
SVC      1,X'6E' (or) M.CALL  H.MONS,46
```

where:

brkadd — is the logical word address of the entry point of the task's break/task interrupt routine

Exit Conditions

Return Sequence:

M.RTRN

Registers:

None

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.6 M.BRKXIT - Exit from Task Interrupt Level

The M.BRKXIT service must be called at the conclusion of executing a task interrupt routine. It transfers control back to the point of interruption.

Entry Conditions

Calling Sequence:

```
M.BRKXIT

(or)

SVC     1,X'70' (or) M.CALL  H.MONS,48
```

Exit Conditions

Return Sequence:

M.RTRN

Registers:

None

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.7 M.CDJS - Submit Job from Disc File

The M.CDJS service submits a job contained on a blocked permanent or temporary disc file. Prior to calling this service, the specified file should be rewound to purge the contents of the blocking buffer if it has been dynamically built.

Entry Conditions

Calling Sequence:

```
M.CDJS    filename [,password]

(or)

LD        R2, password
LD        R6,filename
SVC       1,X'61' (or) M.CALL   H.MONS,27
```

where:

filename — contains the one to eight character name of the blocked permanent disc file which contains the job. If a user name is associated with the calling task, an attempt is made to allocate a user file of the name specified. If unsuccessful, a system file is allocated.

(or)

contains zero in the first word and the second word contains the address of a file control block which is associated with a blocked temporary file in the calling task.

Once submitted, the logical file code associated with the permanent or temporary file is deallocated and may be reassigned by the user task.

password — is the optional password which is required only for password-only (PO) permanent files.

Exit Conditions

Return Sequence:

```
M.RTRN    7
```

Registers:

R7		0 if successful
(or)		
R7	bit 0	set if specified file does not exist, invalid password specified, or FCB is not associated with a temporary file
	bits 1-31	0
(or)		
R7	bit 1	1 if unable to activate system input task
	bits 0,2-31	0

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.8 M.CONN - Connect Task to Interrupt

The M.CONN service is used to centrally connect a task to an interrupt level so that when the interrupt occurs, the specified task will be scheduled for execution (resumed).

Entry Conditions

Calling Sequence:

```
M.CONN      task,intlevel

(or)

LW          R5,intlevel
LI          R6,0          (or) LD R6,taskname
LW          R7,taskno
SVC         1,X'4B'       (or) M.CALL   H.MONS,10
```

where:

task	the address of a doubleword containing the name of the task or 0 in Word 0 and the task number in word 1. Task number must be used if the task is multicopied or shared. A task number of 0 specifies the calling task.
intlevel	is the hardware priority level to which the task is to be connected.

Exit Conditions

Return Sequence:

```
M.RTRN      6,7
```

Registers:

R6	Denial Code 1 = Task already connected to an interrupt 2 = Another task connected to the specified interrupt 3 = Interrupt not SYSGEN specified indirectly connectable 4 = Specified task number not found in Dispatch Queue
R7	zero if task not connected to interrupt. Otherwise, contains the task number.

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

If the task named is not currently in execution, it is preactivated so that the interrupt may be connected.

8.2.9 M.DATE - Date and Time Inquiry

The M.DATE service returns to the caller the date (in ASCII), calendar information (century, year, month and day), and a count of the number of real-time clock interrupts since midnight. To aid in converting the interrupt count to time-of-day, counts of the number of interrupts per second and the number of interrupts per time unit are also returned.

Entry Conditions

Calling Sequence:

```
M.DATE pbaddr

(or)

LA      R7,pbaddr
SVC     1,X'15' (or) M.CALL  H.MONS,70
```

where:

pbaddr — is the logical word address of the first location of a parameter block formatted as follows:

Words 0-1		Current Gregorian Date (ASCII MM/DD/YY)
Word 2	byte 0	Century (Binary)
	byte 1	Year (Binary)
	byte 2	Month (Binary)
	byte 3	Day (Binary)
Word 3		Number of clock interrupts since midnight
Word 4		Number of clock interrupts per second (initialized by SYSGEN)
Word 5		Number of clock interrupts per time unit (initialized by SYSGEN)

Exit Conditions

Return Sequence:

M.RTRN

Registers:

None

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.10 M.DEBUG - Load and Execute Interactive Debugger

A call to the M.DEBUG service will cause the interactive Debugger to be loaded as an overlay segment. Control is then transferred to the Debugger. If the Debugger is already loaded, then only a transfer to the Debugger will take place.

Entry Conditions

Calling Sequence:

```
M.DEBUG

(or)

SVC  1,X'63'  (or)  M.CALL   H.MONS,29
```

Exit Conditions

Return Sequence:

M.RTRN	7	R7 is the transfer address of the Debugger if the Debugger was loaded by this service call. R7 is zero if the Debugger was already loaded at the time this service was called.

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.11 M.DELTSK - Delete Task

A call to the M.DELTSK service forces I/O completion and immediately aborts the specified task (see also Task Termination Sequencing in Chapter 2).

Entry Conditions

Calling Sequence:

```
M.DELTSK        abcode,task,extcode

(or)

 LD  R2,extcode
LW   R5,abcode
LI   R6,0       }              (or) LD R6,taskname
LW   R7,taskno  }
SVC  1,X'5A'  or  M.CALL  H.MONS,31
```

where:

abcode	contains the abort code consisting of four ASCII characters.
task	the address of a doubleword containing the name of the task or 0 in Word 0 and the task number in word 1. Task number must be used if the task is multicopied or shared. A task number of 0 specifies the calling task.
extcode	contains the extended abort code message consisting of one to eight ASCII characters, left justified, and blank filled.

Exit Conditions

Return Sequence:

```
M.RTRN          7
```

Registers:

R7	zero if the specified task was not found in the Dispatch Queue. Otherwise contains the task number.

External References

System Macros:

M.RTRN	M.CALL
M.IOFF	M.IONN
M.OPEN	

Abort Cases:

MS12	Non-privileged user attempting to abort a task with a different owner name.

Output Messages:

Modifies Abort message to be:

ABORT task REASON: xxxx zzzzzzzz AT: yyyyyyyy

where:

zzzzzzzz	is the extended message code supplied with the call to this service.

8.2.12 M.DEVID - Get Device Mnemonic or Type Code

The M.DEVID service allows the user to pass either a device mnemonic or a generic device type code and receive the corresponding type code or mnemonic. See Appendix A for device mnemonic and device type codes.

Entry Conditions

Calling Sequence:

```
M.DEVID         id

(or)

LW   R2,id
SVC  1,X'14' (or)  M.CALL   H.MONS,71
```

where:

id	contains either a device mnemonic in the right halfword with the left halfword zero or a device type code in byte 3 with bytes 0-2 zero.

Exit Conditions

Return Sequence:

```
M.RTRN          2
```

Registers:

If input was a device mnemonic:

R2	bytes 0-2	zero
	byte 3	corresponding device type code

If input was a device type code:

R2	left halfword	zero
	right halfword	corresponding device mnemonic

If input was a mnemonic or device type code not in the system Device Type Table (DTT):

R2	bit 0	set
	bits 1-31	unchanged

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.13 M.DISCON - Disconnect Task from Interrupt

The M.DISCON service is used to disconnect a task which has previously been centrally connected to an interrupt level.

Entry Conditions

Calling Sequence:

```
M.DISCON task

(or)

LI    R6,0        }     (or) LD R6,taskname
LW    R7,taskno   }
SVC   1,X'5D' (or) M.CALL  H.MONS,38
```

where:

task	the address of a doubleword containing the name of the task or 0 in Word 0 and the task number in word 1. Task number must be used if the task is multicopied or shared. A task number of 0 specifies the calling task.

Exit Conditions

Return Sequence:

M.RTRN	7

Registers:

R7	zero if the task was not found in the Dispatch Queue. Otherwise contains the task number.

External References

System Macros:

M.RTRN

Abort Cases:

MS29	Non-privileged user attempting to disconnect a task with a different owner name.

Output Messages:

None

8.2.14 M.DLTT - Delete Timer Entry

The M.DLTT service is used to reset the timer for the specified task so that its specified function will no longer be performed upon time-out.

Entry Conditions

Calling Sequence:

```
M.DLTT          timer

(or)

LW   R7,timer
SVC  1,X'47' (or)  M.CALL   H.MONS,6
```

where:

timer — two-character ASCII name of a timer, right justified

Exit Conditions

Return Sequence:

M.RTRN

CC1 set if timer entry not found

Registers:

None

External References

System Macros:

M.RTRN

Output Messages:

None

NOTE: Deletion of timer entry will not delete the associated task. One-shot timers are deleted on expiration.

8.2.15 M.DSMI - Disable Message Task Interrupt

The M.DSMI service disables the task interrupts for messages sent to the calling task. M.DSMI is useful for synchronization gating of the task message interrupts.

Entry Conditions

M.DSMI

(or)

SVC 1,X'2E' (or) M.CALL H.MONS,57

Exit Conditions

Return Sequence:

M.RTRN

Registers:

CC1 set if task interrupts were already disabled

8.2.16 M.DSUB - Disable User Break Interrupt

The M.DSUB service deactivates the user break interrupt (see M.ENUB) and allows user breaks via the terminal BREAK key to be acknowledged.

Entry Conditions

Calling Sequence:

M.DSUB

(or)

SVC 1,X'12' (or) M.CALL H.MONS,73

Exit Conditions

Return Sequence:

M.RTRN

Registers:

CC1 set if user break already disabled

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.17 M.EAWAIT - End Action Wait

The M.EAWAIT service will wait for the completion of any no-wait request or I/O end action if any are queued. If there aren't any outstanding, the service returns immediately to the user. This service is similar to the M.ANYW service.

Entry Conditions

Calling Sequence:

```
M.EAWAIT time1

(or)

LW   R6,time1
SVC  1,X'1D'          (or)        M.CALL   H.EXEC,40
```

where:

time1	contains zero if wait for an indefinite period is requested. Otherwise, time 1 contains the negative number of time units to elapse before the wait is terminated. (The actual time elapsed can vary by one time unit because of system design.)

Exit Conditions

Return Sequence:

M.RTRN

Registers:

None

External References

System Macros:

M.RTRN

Abort Cases:

MS31	User attempted to go to an any wait state from an end action routine.

Output Messages:

None

8.2.18 M.ENMI - Enable Message Task Interrupt

The M.ENMI service enables task interrupts for messages sent to the calling task. It is used to remove an inhibit condition previously established by invoking the M.DSMI service.

Entry Conditions

Calling Sequence:

```
M.ENMI

(or)

SVC     1,X'2F'  (or)  M.CALL  H.MONS,58
```

Exit Conditions

Return Sequence:

```
M.RTRN
```

Registers:

CC1	set if task interrupts were already enabled

8.2.19 M.ENUB - Enable User Break Interrupt

The M.ENUB service activates the user break interrupt and causes further user breaks via the terminal BREAK key to be ignored.

Entry Conditions

Calling Sequence:

M.ENUB

(or)

SVC 1,X'13' (or) M.CALL H.MONS,72

Exit Conditions

Return Sequence:

M.RTRN

Registers:

CC1 set if user break already enabled

External References

System Macros:

M.RTRN

Abort Cases:

None

Ouput Messages:

None

8.2.20 M.EXIT - Terminate Task Execution

The M.EXIT service performs all normal termination functions required of exiting tasks (see Section 2). All devices and memory are deallocated, related table space is erased, and the task's Dispatch Queue entry is cleared.

Entry Conditions

Calling Sequence:

M.EXIT

(or)

SVC 1,X'55' (or) M.CALL H.MONS,18

Exit Conditions

Return Sequence:

M.RTRN

Registers:

None

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.21 M.GMSGP - Get Message Parameters

The M.GMSGP service is called from the message receiver routine of a task that has received a message interrupt. Its purpose is to transfer the message parameters into the designated receiver buffer, and to post the ownername and task number of the sending task into the Parameter Receive Block (PRB). (For description of the PRB, see Section 3.)

Entry Conditions

Calling Sequence:

```
M.GMSGP    prbaddr

(or)

LA   R2,prbaddr
SVC  1,X'7A'  (or)  M.CALL  H.MONS,35
```

where:

prbaddr is the logical address of the Parameter Receive Block (PRB).

Exit Conditions

Return Sequence:

```
M.RTRN     6
```

Registers:

R6 Contains the processing status error code:

Error Code	Definition
0	Normal status
1	Invalid PRB address
2	Invalid receiver buffer address
3	No active send request
4	Receiver buffer length exceeded

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.22 M.GRUNP - Get Run Parameters

The M.GRUNP service is called by a task that is executing on behalf of a run request. Its purpose is to transfer the run parameters into the designated receiver buffer, and to post the ownername and task number of the sending task into the Parameter Receive Block (PRB). (See Section 3.)

Entry Conditions

Calling Sequence:

```
M.GRUNP          prbaddr

(or)

LA   R2,prbaddr
SVC  1,X'7B'  (or)  M.CALL  H.MONS,36
```

where:

prbaddr is the logical address of the Parameter Receive Block (PRB).

Exit Conditions

Return Sequence:

M.RTRN

Registers:

R6 Contains the processing status error code:

Error Code	Definition
0	Normal status
1	Invalid PRB address
2	Invalid receiver buffer address
3	No active send request
4	Receiver buffer length exceeded

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.23 M.HOLD - Program Hold Request

The M.HOLD service makes the specified task ineligible for CPU control by setting the hold bit in the CPU Dispatch Queue. The specified task remains in the hold state until the operator issues the OPCOM CONTINUE command. If the specified task is not in the CPU Dispatch Queue, the request is ignored.

Entry Conditions

Calling Sequence:

```
        M.HOLD   task

        (or)

        LI       R6,0      }        (or) LD R6,taskname
        LW       R7,taskno }
        SVC      1,X'58'  (or) M.CALL  H.MONS,25
```

where:

task	the address of a doubleword containing the name of the task or 0 in Word 0 and the task number in word 1. Task number must be used if the task is multicopied or shared. A task number of 0 specifies the calling task.

Exit Conditions

Return Sequence:

```
        M.RTRN   7
```

Registers:

R7	zero if the specified task was not found in the Dispatch Queue, or if the task is scheduled to leave the system. Otherwise contains the task number.
CC1	set if the task is scheduled to leave the system.

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.24 M.ID - Get Task Number

The M.ID service allows the user to pass the address of a parameter block containing any of the following: (a) task number, (b) task load module name, (c) owner name, or (d) task pseudonym, and the service will provide the missing items if a matching entry is found. Initially, the caller passes zero (0) as the index value following the parameter block address. If more than one task in the Dispatch Queue satisfies the given parameters, the service returns to the caller with an index value in Register 5 for retrieval of further entries. The caller is responsible for updating the index with the contents of Register 5 and reissuing M.ID until all tasks that meet specifications have been identified or R5 = 0.

Entry Conditions

Calling Sequence:

```
M.ID        pbaddr,index

(or)

LW   R5,a variable equal to 0 or an index
LA   R7,pbaddr
SVC  1,X'64' (or) M.CALL  H.MONS,32
```

where:

pbaddr — is the logical word address of the first location of a parameter block formatted as follows:

Word 0	Task Activation Sequence Number
Words 1-2	Task Load Module Name
Words 3-4	Owner Name
Words 5-6	Pseudonym

The user supplies those items that are known and zeros the other words.

index — a variable equal to zero for initial call, then set to plus or minus value for each subsequent call. When R5 = 0, all tasks that match have been identified.

Exit Conditions

Return Sequence:

M.RTRN 5

Registers:

R5 = 0 if no entry satisfies the given parameter(s).
R5 = Bit 0 is set if more than one task satisfies the given parameters. Bits 1-31 contain the DQE address of the current task found.

R5 may be used as input for subsequent calls.

External References

System Macros:

M.RTRN
M.RTNA

Abort Cases:

MS32	Invalid entry to M.ID (H.MONS,32)

Output Messages:

None

8.2.25 M.INT - Activate Task Interrupt

The M.INT service allows the calling task to cause the previously declared break/task interrupt receiver routine of the specified task to be entered.

Entry Conditions

Calling Sequence:

```
M.INT     task

(or)

ZR        R6          }      (or) LD R6,taskname
LW        R7,taskno   }
SVC       1,X'6F' (or) M.CALL  H.MONS,47
```

where:

task	the address of a doubleword containing the name of the task or 0 in Word 0 and the task number in word 1. Task number must be used if the task is multicopied or shared. A task number of 0 specifies the calling task.

Exit Conditions

Return Sequence:

```
M.RTRN    6,7
```

Registers:

R6	bit 0	1 if the specified task was not set up to receive a pseudo interrupt
		0 Otherwise
	bits 1-31	0
R7		zero if the specified task was not found in the Dispatch Queue. Otherwise contains the task number.

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.26 M.MYID - Get Task Number

The M.MYID service allows the user to obtain status on the currently executing task.

Entry Conditions

Calling Sequence:

```
M.MYID      pbaddr

(or)

ZR          R5
SBR         R5, 0
LA          R7,pbaddr
SVC         1,X'64' (or)  M.CALL   H.MONS,32
```

where:

pbaddr is the logical word address of the first location of a parameter block formatted as follows:

Word 0	Task Activation Sequence Number
Words 1-2	Task Load Module Name
Words 3-4	Owner Name
Words 5-6	Pseudonym
Words 7-8	User Name
Word 9	User Key (right-justified, left-halfword zero filled)
Word 10	Task Scheduling Flags

Exit Conditions

Return Sequence:

M.RTRN	5

Registers:

R5, R7	unchanged

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Message:

None

8.2.27 M.OLAY - Load Overlay Segment

The M.OLAY service is provided for loading an overlay segment. The actual loading is performed by the System Loader (H.LODR) and control is returned to the caller upon completion. The named segment must have been defined to the Cataloger as an overlay load module.

If desired, can transfer control to the overlay segment at its cataloged transfer address rather than to the caller.

Entry Conditions

Calling Sequence:

```
M.OLAY    loadmod [,EXE]

(or)

LD        R6,loadmod
SVC       1,X'50'  (or)  M.CALL  H.MONS,13

(or)

[SVC      1,X'51'  (or)  M.CALL  H.MONS,14]
```

where:

loadmod	is a doubleword containing the name of the file containing the overlay segment to be loaded, one to eight ASCII characters, left-justified and blank filled.
EXE	specifies transfer control to the overlay (SVC1,X'51'). Default: loads overlay without transfer (SVC1,X'50').

Exit Conditions

Return Sequence:

```
M.RTRN    7
```

Registers:

R7	transfer address of the overlay segment

External References

System Macros:

M.CALL
M.RTRN

Abort Cases:

MS07	LD01-08	Cannot load overlay segment due to software checksum or data error.
MS08		Overlay is not in the SMD.
MS10		Overlay has an invalid preamble.
MS11		Unrecoverable I/O error during overlay loading.
MS12		Overlay is password protected.

Output Messages:

None

8.2.28 M.PGOW - Task Option Word Inquiry

The M.PGOW service provides the caller with the 32-bit task option word (same as program option word).

Entry Conditions

Calling Sequence:

```
M.PGOW

(or)

SVC     1,X'4C' (or) M.CALL  H.MONS,24
```

Exit Conditions

Return Sequence:

```
M.RTRN  7
```

Registers:

```
R7      contains the 32-bit task option word
```

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.29 M.PRIL - Change Priority Level

The M.PRIL service is provided to the privileged caller who wishes to dynamically alter the priority level of the specified task. Valid priority levels for real-time tasks are 1-54 inclusive. Valid priority levels for time distribution tasks are 55-64 inclusive. A real-time task cannot be changed to a time distribution priority level and a time distribution task cannot be changed to a real-time priority level. I/O continues to operate at base priority level of the cataloged task.

Entry Conditions

Calling Sequence:

```
M.PRIL      task,priority
(or)
LW          R5,priority
LI          R6,=0      }              (or) LD R6,taskname
LW          R7,taskno  }
SVC         1,X'4A' (or) M.CALL   H.MONS,9
```

where:

task	the address of a doubleword containing the name of the task or 0 in Word 0 and the task number in word 1. Task number must be used if the task is multicopied or shared. A task number of 0 specifies the calling task.
priority	is the priority level to be assigned to the task (1-54 for a real-time task; 55-64 for a time distribution task).

Exit Conditions

Return Sequence:

```
M.RTRN    7
```

Registers:

R7	zero if the specified task was not found in the Dispatch Queue. Otherwise contains the task number.

External References

System Macros:

M.RTRN
M.CALL
M.OPEN

Abort Cases:

MS06	Specified priority not in 1-64 range.

Output Messages:

None

8.2.30 M.PTSK - Parameter Task Activation

The M.PTSK privileged service activates a task whose load module name, optional resource requirements, and optional pseudonym are specified to the service call.

Entry Conditions

Calling Sequence:

```
M.PTSK          actaddr ,psbaddr

(or)

LA      R1,actaddr
LA      R2,psbaddr (or) ZR R2
SVC     1,X'5F' (or) M.CALL  H.MONS,40
```

where:

actaddr is the logical word address of the first location of an activation parameter block formatted as shown below

PARAMETER BLOCK

(Words 0-12 are required. The RRS entries (words 13-205) are optional as indicated by byte 1.)

Word 0	byte 0	bits	1 = Reserved 2 = Terminal task 3 = Batch task 4 = Debug overlay required 5 = RTM resident (ESTABLISH)
	byte 1		Number of resource requirements or zero if same as summary entries in Load Module Information Table (LMIT)
	byte 2		Memory requirement = number of 512 word pages exclusive of TSA or zero if memory requirements are to be taken from the LMIT
	byte 3		Memory class (ASCII E, H or S) or zero if memory class to be taken from LMIT
Word 1	byte 0		Number of blocking buffers required or zero if same as LMIT
	byte 1		Number of FAT/FPT pairs (files) to be reserved or zero if same as LMIT

byte 2	Priority level of task being activated or zero if same as LMIT. Overridden if priority is specified in the option PSB.
byte 3	Reserved for future use
Word 2,3	Load Module Name - one to eight ASCII character, left-justified, blank filled, load module name
Word 4,5	Pseudonym - one to eight ASCII character left-justified, blank filled pseudonym to be associated with task or zero if no pseudonym is desired (pseudonym's are intended to be unique - the responsibility for uniqueness rests with the caller)
Word 6,7	Owner Name - one to eight ASCII character, left-justified, blank filled, owner name to be associated with task or zero if task to default to current owner name. Default is to current owner name if a PSB is used.
Word 8,9	User Name - one to eight ASCII character, left-justified, blank filled user name to be associated with files referenced by this task or zero if same as LMIT
Word 10	User Key - right justified compressed halfword equivalent of user key. If zero, same as LMIT.
Word 11	Task Option Word - contains the initial value of the task option word or zero
Word 12	Task Status Word - contains the initial value of the task status word or zero
Words 13-205 (optional)	Resource Requirement Summary - each entry contains three words. The maximum number of entries is 64. Each entry is compared with the RRS entries in the LMIT. If the logical file code currently exists, the specified lfc assignment will override the cataloged assignment, otherwise it will be treated as an additional requirement (merged).
psbaddr	is the logical address of the Parameter Send Block (PSB) or zero, if no parameters are to be passed. See Chapter 3 for PSB description.

Exit Conditions

Return Sequence:

M.RTRN 6,7

Registers:

R0	is destroyed
R6 = 0	if the service could be performed
R7	contains the task number of task activated by this service
(or)	
R6 = 1	if invalid attempt to multicopy a unique load module
R7	task number of existing task with same name
(or)	
R6 = 2	if load module file not in SMD
= 3	if load module file is password protected
= 4	if file does not contain valid data
= 5	if no DQE is available
= 6	if read error on SMD
= 7	if read error on load module
= 8	insufficient memory
= 9	calling task is unprivileged

External References

System Macros:

M.RTRN

Abort Codes:

None

Output Messages:

None

8.2.31 M.RCVR - Receive Message Link Address

The M.RCVR service allows the caller to establish the address of a routine to be entered for the purpose of receiving messages sent by other tasks.

Entry Conditions

Calling Sequence:

```
M.RCVR    recvadd

(or)

LA        R7,recvadd
SVC       1,X'6B' (or) M.CALL  H.MONS,43
```

where:

recvadd — is the logical word address of the entry point of the receive message routine in the user's task.

Exit Conditions

Return Sequence:

```
M.RTRN    7
```

Registers:

R7 — contains zero if the receiver address was invalid, otherwise contains the receiver address

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.32 M.SETS - Set User Status Word

The M.SETS service allows the calling task to modify any task's user status word. Along with the Test User Status Word service, this is one of the means provided by MPX-32 for task-to-task communication. The user status word resides in the CPU Dispatch Queue (DQE.USW) and has a value of zero until modified by this service. The user status word is removed from the queue, modified as specified, and replaced in the queue.

Entry Conditions

Calling Sequence:

```
M.SETS   function,statusw[,task]

(or)

LD       R4,task
LI       R6,function
LW       R7,statusw
SVC      1,X'48'  (or)  M.CALL   H.MONS,7
```

where:

function — STF (1), RSF (2), STC (3), or INC (4) specify the type of modification to be performed and are Set Flag, Reset Flag, Set Counter, and Increment Counter respectively. If using the macro call, specify the alphabetic code. If loading registers, specify the corresponding numeric.

statusw — contains a function parameter specific to function codes as follows
1=bit position in the status word to be set (1-31)
2=bit position in the status word to be reset (1-31)
3=value to which the status word is to be set
4=value by which the status word is to be incremented

task — is the address of a doubleword containing the name of the task, or 0 in word 0 and the task number in word 1. Task number must be used if the task is multicopied or shared. A task number of 0 or omission of the argument specifies the calling task.

Exit Conditions

Return Sequence:

```
M.RTRN   5
```

Registers:

R5 — bit 0 set if the specified task number was not found in the Dispatch Queue. Otherwise, R5 is zero.

External References

System Macros:

M.RTRN

Abort Cases:

MS05	Invalid function code specification.

Output Messages:

None

8.2.33 M.SETT - Create Timer Entry

The M.SETT service builds an entry in the timer table so that the requested function is performed upon time-out. Timer entries may be created to activate a program, resume a program, set a bit in memory, reset a bit in memory, or request an interrupt. Any task may create a timer to activate or resume a program. Timer entries to set or reset bits may be created by any task, provided the bit is within a static memory partition. Only privileged users may set bits in the operating system. A request interrupt can only be requested by a privileged task.

Entry Conditions

Calling Sequence:

```
M.SETT      timer,t1,t2,function,arg4,arg5

(or)

LB          R3,function
SLL         R3,24
ORMW        R3,timer
LW          R4,t1
LW          R5,t2
LW (or LD)  R6,arg4
(LW         R7,arg5)
SVC         1,X'45' (or) M.CALL H.MONS,4
```

where:

timer — is a word containing zeros in bytes 0 and 1, and a two-character timer identification in bytes 2 and 3.

t1 — contains the current value to which the timer will be set in negative time units.

t2 — contains the value to which the timer will be reset upon each time-out in negative time units. If the reset value is zero, the function will be performed upon time-out and the timer entry will be deleted. This case is called a "one-shot" timer entry.

function — ACP (1), RSP or RST (2), STB (3), RSB (4), and RQI (5) specify the function to be timed and are activate program, resume program, set bit, reset bit, and request interrupt, respectively. If using the macro call, specify the alphabetic code. If loading registers, specify the corresponding numeric.

ARG4 and ARG5 and FCT, CODE contain values specific to the function being timed as follows:

FUNCTION	NUMERIC	ARG4 and ARG5
ACP	1	arg4 contains the one- to eight-character name of the program to be activated. If the task named is not currently in execution, it is preactivated to connect the interrupt to the task. This connection remains in effect until the task aborts or the timer is deleted. On normal exit, the timer table is updated to point to the next generation. If arg4 is zero, indicates the current task. arg5 is null.
RSP	2	arg4 is a doubleword containing the one- to eight-character name of the task to be resumed in R6 and R7. If arg4 is zero, indicates the current task. arg5 is null.
(or) RST	2	arg4 is the task number entered into R7 and R6 is zeroed. If arg4 is zero, indicates the current task. arg5 is null.
STB	3	arg4 contains the address of the word in which the bits are to be set. The address must be within the operating system (if privileged) or a static memory partition. arg5 contains the bit configuration of the mask word to be ORed.
RSB	4	arg4 contains the address of the word in which the bit is to be reset. The address must be within the operating system (if privileged) or a static memory partition. arg5 contains the bit configuration of the mask word to be ANDed.
RQI	5	arg4 contains the priority level of the interrupt to be requested. arg5 is null.

Exit Conditions

Returned Sequence:

Normal Return:

M.RTRN R3

R3 is non-zero and condition codes are not set.

Error Condition:

M.RTRN R3

R3 is zero if request is denied.
If no condition codes are set on a denial return, then there are no timer entries available. Otherwise, the condition codes are interpreted as follows:

CC1 set if requested load module does not exist
CC2 set if requested task is not active
CC3 set if attempting to create a duplicate timer id

Abort Codes:

MS02	Invalid function code specified for request to create a timer entry. Valid codes are ACP (1), RSP or RST (2), STB (3), RSB (4), and RQI (5).
MS03	A privileged task bit Set/Reset address is outside of the operating system or a static memory partition, or an unprivileged task bit Set/Reset address is outside of a static memory partition.
MS04	Task has attempted to create a timer entry to request an interrupt with a priority level outside the range of X'12' to X'7F', inclusive, or the requesting task is unprivileged.

8.2.34 M.SMSGR - Send Message to Specified Task

The M.SMSGR service allows a task to send up to 768 bytes to the specified destination task. Up to 768 bytes may also be accepted as return parameters. For further description, see Chapter 3.

Entry Conditions

Calling Sequence:

```
M.SMSGR   psbaddr

(or)

LA        R2,psbaddr
SVC       1,X'6C'  (or)  M.CALL  H.MONS,44
```

where:

psbaddr — is the logical address of the Parameter Send Block (PSB). See Chapter 3 for PSB description.

Exit Conditions

Return Sequence:

```
M.RTRN    6
```

Registers:

R6 — Contains the processing start (initial) error status if any:

Error Code	Definition
0	Normal initial status
1	Task not found
2-9	Reserved
10	Invalid priority
11	Invalid send buffer address or send quantity exceeds 768 bytes
12	Invalid return buffer address
13	Invalid no-wait mode end-action routine address
14	Memory pool unavailable
15	Excessive no wait run requests. Limit is 5 for unprivileged tasks and 255 for privileged tasks.
16	Invalid PSB address
17	Destination task is not a valid message receiver

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.35 M.SRUNR - Send Run Request to Specified Task

The M.SRUNR service allows a task to activate or re-execute the specified destination task with a parameter pass of up to 768 bytes. Up to 768 bytes may also be accepted as return parameters. For further description, see Chapter 3.

Entry Conditions

Calling Sequence:

```
M.SRUNR   psbaddr

(or)

LA        R2,psbaddr
SVC       1,X'6D'  (or)  M.CALL  H.MONS,45
```

where:

psbaddr — is the logical address of the Parameter Send Block (PSB). See Chapter 3.

Exit Conditions

Return Sequence:

```
M.RTRN    6,7
```

Registers:

R6 — Contains the processing start (initial) error status if any:

Error Code	Definition
0	Normal initial status
1	Reserved
2	Load module name not found in SMD
3	Load module file is password protected
4	Invalid load module file format
5	Dispatch Queue Entry (DQE) unavailable
6	I/O error on SMD read
7	I/O error on load module read
8	Memory unavailable
9	Invalid taskno for run request to multicopied load module in RUNW state
10	Invalid priority
11	Invalid send buffer address or send quantity exceeds 768 bytes
12	Invalid return buffer address
13	Invalid no-wait mode end-action routine address
14	Memory pool unavailable
15	Excessive no-wait run request. Limit is 5 for unprivileged tasks and 255 for privileged tasks.
16	Invalid PSB address
17	Reserved

R7 — Contains the task number (taskno) of the destination task, or zero if the request was not processed.

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.36 M.SUAR - Set User Abort Receiver Address

The M.SUAR service provides the user with the ability to set up an address to which control will be returned on an abort during task execution.

All files remain open prior to transferring to the user specified address. See Task Termination Sequencing in Chapter 2.

Entry Conditions

```
M.SUAR    address

(or)

LA        R7,address
SVC       1,X'60' (or) M.CALL  H.MONS,26
```

where:

address — is the logical address to which control will be transferred on task termination

Exit Conditions

Return Sequence:

```
M.RTRN    7
```

Registers:

R7 bit 0 — 1 if the request is denied because the specified address is outside the user's allocated area

bit 0 — 0 if the request is honored

bits 1-31 — 0

External References

System Macros:

M.SPAD
M.CALL
M.RTRN

Abort Cases:

MS89 — An unprivileged task has attempted to reestablish an abort receiver (other than M.IOEX).

Output Messages:

None

8.2.37 M.SUME - Resume Task Execution

The M.SUME service is used to resume a task that has been suspended. A request to resume a task which is not suspended is ignored.

Entry Conditions

Calling Sequence:

```
M.SUME   task

(or)

ZR       R6      }        (or) LD R6,taskname
LW       R7,taskno}
SVC      1,X'53' (or) M.CALL  H.MONS,16
```

where:

task — the address of a doubleword containing the name of a task or 0 in Word 1 and the task number in Word 2. Task number must be used if the task is multicopied or shared.

Exit Conditions

Return Sequence:

```
M.RTRN   7
```

Registers:

R7 — zero if any of the following conditions exist:

- the specified task number or task name was not found in the dispatch queue
- the specified task name was not single copied
- the owner name of the task requesting the abort is restricted from access to tasks with a different owner name (via the M.KEY file), the task requesting the abort is not privileged, and the owner names of the requesting and target task do not match
- the task is in the process of exiting the system

Otherwise, task number.

CC1 — set if the task is in the process of exiting the system.

External References

System Macros:

```
M.CALL   M.IOFF
M.RTRN   M.IONN
M.OPEN
```

Abort Cases:

None

Output Messages:

None

8.2.38 M.SUSP - Suspend Task Execution

The M.SUSP service 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 system service. Suspension of a task can also be ended upon receipt of a message interrupt.

Entry Conditions

Calling Sequence:

```
M.SUSP   task,time1

(or)

LW       R5,time1
LI       R6,0        }                    (or) LD R6,taskname
LW       R7,taskno   }
SVC      1,X'54' (or) M.CALL  H.MONS,17
```

where:

task — the address of a doubleword containing the name of a task or 0 in Word 1 and the task number in Word 2. Task number must be used if the task is multicopied or shared. A task number of 0 specifies the calling task.

time1 — contains zero, if suspension for an indefinite time interval is requested. Else the negative number of time units to elapse before the caller is resumed. (The actual time elapsed can vary by one time unit because of system design.)

Exit Conditions

Return Sequence:

```
M.RTRN   7
```

Registers:

R7 — task number if the service was performed.
Zero if the service could not be performed,

CC1 — set if specified task number was not found in the Dispatch Queue,

CC2 — set if the time interval was invalid.

External References

System Macros:

M.RTRN	M.IONN
M.CALL	M.IOFF
M.OPEN	

Abort Cases:

None

Output Messages:

None

8.2.39 M.SYNCH - Set Synchronous Task Interrupt

The M.SYNCH service will cause message and task interrupts to be deferred until the user makes a call to M.ANYW, M.EAWAIT, M.WAIT, or M.SYNCH. When this service is used, message interrupts will not be interrupted by End Action Interrupts. All task interrupt levels cannot be interrupted, except by break, until they voluntarily relinquish control.

Calling Sequence:

```
M.SYNCH

(or)

SVC     1,X'1B'     (or)  M.CALL   H.MONS,67
```

Exit Conditions

Return Sequence:

```
M.RTRN
```

Registers:

```
CC1     set if synchronous task interrupt was already set
```

External References

System Macros:

```
M.RTRN
```

Abort Cases:

None

Output Messages:

None

8.2.40 M.TDAY - Time-of-Day Inquiry

The M.TDAY service returns to the caller the time of day as computed from the real-time clock interrupt counter. The counter is initialized via a SYSGEN parameter and may be modified at any time by the OPCOM ENTER command.

Entry Conditions

Calling Sequence:

```
M.TDAY

(or)

SVC     1,X'4E' (or) M.CALL   H.MONS,11
```

Exit Conditions

Return Sequence:

```
M.RTRN  7
```

Registers:

R7	byte 0=hours (0-23) byte 1=minutes (0-59) byte 2=seconds (0-59) byte 3=interrupts (less than one second)

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.41 M.TSTE - Arithmetic Exception Inquiry

The M.TSTE service resets the arithmetic exception status bit in the user's TSA and returns CC1 set or reset according to the status value. The status bit is set whenever the user is in execution and an arithmetic exception trap occurs. The bit remains set until this service is requested, or the task terminates.

Entry Conditions

Calling Sequence:

```
M.TSTE

(or)

SVC     1,X'4D' (or) M.CALL  H.MONS,23
```

Exit Conditions

Return Sequence:

```
M.RTRN
```

Registers:

```
PSD     CC1 contains the value of the arithmetic exception status bit
```

External References

System Macros:

```
M.RTRN
```

Abort Cases:

None

Output Messages:

None

8.2.42 M.TSTS - Test User Status Word

The M.TSTS service returns the 32 bit user status word of any specified task in execution. The user status word resides in the CPU Dispatch Queue (DQE.USW) and is modified by the Set User Status Word system service. These two services treat the user status word as either a set of 32 flags or as a 32 bit counter. Bit 0 is used as a status flag.

Entry Conditions

Calling Sequence:

```
M.TSTS    task

(or)

LI        R6,0      }                    (or) LD R6,taskname
LW        R7,taskno }
SVC       1,X'49' (or) M.CALL   H.MONS,8
```

where:

task — the address of a doubleword containing the name of a task or 0 in Word 1 and the task number in Word 2. Task number must be used if the task is multicopied or shared. A task number of 0 specifies the calling task.

Exit Conditions

Return Sequence:

```
M.RTRN    7
```

Registers:

R7 bit 0 — set if the specified task was not found in the Dispatch Queue. Otherwise, R7 returns the user-status word.

External References

System Macros:

M.RTRN, M.CALL, M.OPEN

Abort Cases:

None

Output Messages:

None

8.2.43 M.TSTT - Test Timer Entry

The M.TSTT service returns to the caller the negative number of time units remaining until the specified timer entry time-out. If the timer has expired, the result returned is zero.

Entry Conditions

Calling Sequence:

```
M.TSTT    timer

(or)

LW        R6,timer
SVC       1,X'46'  (or)  M.CALL   H.MONS,5
```

where:

timer — two-character ASCII name of a timer, right justified.

Exit Conditions

Return Sequence:

```
M.RTRN    7
```

Registers:

R7 — negative number of time units remaining until time out

(or)

zero if the timer has expired, or does not exist

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.44 M.TURNON - Activate Program at Given Time of Day

The M.TURNON service activates or resumes a specified task at a specified time and reactivates (resumes) it at specified intervals by creating a timer table entry using a specified timer ID.

Entry Conditions

Calling Sequence:

```
M.TURNON loadmod,time,[reset],timerid

(or)

LD      R6,loadmod
LW      R4,time
LW      R5,reset
LW      R3,timerid
SVC     1,X'1E'   (or)   M.CALL   H.MONS,66
```

where:

loadmod — is the 1-8 character name of the permanent load module file where the task is cataloged. It must be a system file (no user name) and it must be a left-justified blank filled doubleword. Zero load module indicates the current task.

time — is the time of day on the 24-hour clock when the task will be activated. It is a word value with the following format:

```
Byte 0  =  binary hours
Byte 1  =  binary minutes
Byte 2  =  binary seconds
Byte 3  =  zero
```

reset — is the time interval on the 24-hour clock to elapse before resetting the clock upon each timeout. It has the same format as the 'time' argument above. The task will be reactivated at each timeout. If a reset value is not specified, the comma denoting the field must still be specified and the task will be activated only once.

timerid — is a word variable containing the right-justified zero filled 2-character ASCII name of the timer that will be created.

Exit Conditions

Return Sequence:

Normal Return:

```
M.RTRN   R3
```

R3 is non-zero.

Error Condition:

```
M.RTRN    R3
```

R3 is zero if request is denied

If no condition codes are set on a denial return, then there are no timer entries available. Otherwise, the condition codes are interpreted as follows:

CC1	set if requested load module does not exist
CC2	set if requested task not active
CC3	set if attempting to create a duplicate timer ID.

External References

System Macros:

```
M.CALL    H.MONS,4
M.RTRN
```

Abort Cases:

None

Output Messages:

None

8.2.45 M.XBRKR - Exit from Task Interrupt Level

The M.XBRKR service must be called at the conclusion of executing a task interrupt routine. It transfers control back to the point of interruption.

Entry Conditions

Calling Sequence:

```
M.XBRKR

(or)

SVC     1,X'70'  (or)  M.CALL  H.MONS,48
```

Exit Conditions

Return Sequence:

M.RTRN

Registers:

None

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.2.46 M.XMEA - Exit from Message End Action Routine

The M.XMEA service is called to exit the end action routine associated with a no-wait message send request. For further description, see Chapter 3.

Entry Conditions

Calling Sequence:

```
M.XMEA

(or)

SVC     1,X'7E'  (or)  M.CALL  H.MONS,50
```

Exit Conditions

Return Sequence:

M.RTRN (to interrupt context at message interrupt or task base level)

Registers:

None

External References

System Macros:

M.RTRN

Abort Cases:

MS99 End action interrupt was inactive when end action exit issued.

Output Messages:

None

8.2.47 M.XMSGR - Exit from Message Receiver

The M.XMSGR service must be called to exit the message receiver code of the calling task after the task has received a message from another task. For further description, see Chapter 3.

Entry Conditions

Calling Sequence:

```
M.XMSGR  rxbaddr

(or)

LA       R2,rxbaddr
SVC      1,X'5E' (or) M.CALL  H.MONS,39
```

where:

rxbaddr — is the logical address of the Receiver Exit Block (RXB).

Exit Conditions

Return Sequence:

M.RTRN (to interrupted context at task base level)

Registers:

None

External References

System Macros:

M.RTRN

Abort Cases:

MS93 — Invalid Receiver Exit Block (RXB) address was encountered during message exit.

MS94 — Invalid Receiver Exit Block (RXB) return buffer address was encountered during message exit.

MS95 — Task has made a message exit while the message interrupt was not active.

Output Messages:

None

8.2.48 M.XREA - Exit from Run Request End Action Routine

The M.XREA service is called to exit the end action routine associated with having sent a no-wait run request.

Entry Conditions

Calling Sequence:

```
M.XREA

(or)

SVC     1,X'7F'  (or)  M.CALL  H.MONS,51
```

Exit Conditions

Return Sequence:

M.RTRN (to interrupted context at message interrupt or task base level)

Registers:

None

External References

System Macros:

M.RTRN

Abort Cases:

MS99 End action interrupt was inactive when end action exit issued.

Output Messages:

None

8.2.49 M.XRUNR - Exit Run Receiver

The M.XRUNR service is called to exit a task which was executing on behalf of a run request issued from another task.

Entry Conditions

Calling Sequence:

```
M.XRUNR  rxbaddr

(or)

LA       R2,rxbaddr
SVC      1,X'7D'  (or)  M.CALL  H.MONS,49
```

where:

rxbaddr — is the logical address of the Receiver Exit Block (RXB). For further description, see Chapter 3.

Exit Conditions

Return Sequence:

The run receiver queue will be examined and if not empty, the task will be executed again on behalf of the next request. If the queue is empty, the exit options in the RXB are examined. If option byte is 0, the task will be placed in a wait state, waiting for the next run request to be received. If option byte is non-zero, the task will exit the system. Note: if the task is re-executed, control will be transferred to the instruction following the M.XRUNR call.

External References

System Macros:

None

Abort Cases:

MS96	Invalid Receiver Exit Block (RXB) address.
MS97	Invalid return parameter buffer address.
MS98	Run receiver mode not active when run receiver exit issued.

Output Messages:

None

8.2.50 M.XTIME - Task CPU Execution Time

The M.XTIME service returns to the caller a doubleword value representing the total elapsed CPU execution time (in microseconds) since initiation of the task. The CPU execution time is determined by adding the accumulated CPU and IPU interval timer ticks to the number of interval timer ticks expired in the current time quantum. This sum is multiplied by the value (in tenths of microseconds) that each interval timer tick represents. This product is divided by ten to get the final result into microseconds.

Entry Conditions

Calling Sequence:

```
M.XTIME

(or)

SVC     1,X'2D'  (or)  M.CALL H.MONS,65
```

Exit Conditions

Return Sequence:

```
M.RTRN  6,7
```

Registers:

R6,R7 CPU and IPU execution time in microseconds

External References

System Macros:

None

Abort Cases:

None

Output Messages:

None

8.2.51 Debug Link Service

The Debug Link service is intended to be used only by the interactive Debugger for the purpose of transferring control to the Debugger. The Debugger plants this SVC trap in the user's task at the desired location.

Entry Conditions

Calling Sequence:

```
SVC     1,X'66' (or) M.CALL  H.MONS,42
```

Exit Conditions

Return Sequence:

M.RTRN

Registers:

None

External References

System Macros:

M.RTNA

Abort Cases:

None

Output Messages:

None

8.3 Memory Management Services

8.3.1 M.ADRS - Memory Address Inquiry

The M.ADRS service provides the beginning and ending logical addresses of the memory allocated to a task. The beginning address is the location into which the first word was loaded and is a word address. The ending address is also a word address and defines the last word allocated to the task.

Entry Conditions

Calling Sequence:

```
M.ADRS

(or)

SVC     1,X'44' (or) M.CALL  H.MONS,3
```

Exit Conditions

Return Sequence:

```
M.RTRN  6,7
```

Registers:

R6	Logical word address of the first location of the task's DSECT. This address is always on a page boundary.
R7	Logical word address of the last location available for loading or expansion of the task's DSECT. This address is always on a map block boundary -1W.

External References

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.3.2 M.DUMP - Memory Dump Request

The M.DUMP service provides a dump of the caller's Program Status Doubleword (PSD), General Purpose Registers, and specified memory limits. The output is to a SLO file in side-by-side hexadecimal with ASCII format, with the PSD and registers preceding the specified memory limits. The PSD and registers are extracted from the first level of push-down of the calling task. Optionally, register 5 may specify the address of a ten word block containing registers 0 through 7 and the PSD to be dumped, respectively. Any task may request a memory dump.

Entry Conditions

Calling Sequence:

```
M.DUMP   start,end[,mem3]

(or)

ZR       R5 (or)  LA  R5,mem3
LW       R6,start
LW       R7,end
SVC      1,X'4F' (or) M.CALL  H.MONS,12
```

where:

start — contains the low logical word address requested in dump

end — contains the high logical word address requested in dump

mem3 — is the optional address of ten consecutive words containing R0 through R7 and a PSD, respectively. If R5=0, the registers and PSD dumped are taken from the first level of push-down.

NOTE: Start and end are truncated to the nearest 8-word boundaries and memory is dumped between the truncated limits.

Exit Conditions

Return Sequence:

```
M.RTRN   6,7
```

Registers:

R6 — Reason:

1=high dump limit less than low limit
4=no FAT or FPT space available
5=request made with insufficient levels of push-down available
6=cannot allocate SLO file
7=unrecoverable I/O error

R7 — zero if dump could not be performed

External References

System Macros:

M.SPAD
M.CALL
M.RTRN

Abort Cases:

None

Output Messages:

None

8.3.3 M.EXCL - Free Shared Memory

The M.EXCL service allows a task to dynamically deallocate any common areas it has previously shared (M.SHARE) or included (M.INCL). M.EXCL causes the allocation count (SMT.ACNT) and use count (SMT.USE) to be decremented by one. The common area will be deleted and its resources returned to the free list when the allocation count goes to zero.

M.EXCL is also called by the exit processor (H.ALOC,3). If a task begins to abort or come to any unnatural or untimely end while the task was in the process of building a shared partition, or loading data into a shared partition through the use of a shared memory data lock, the exclude service will handle any tasks that are queued to the aborting shared memory table. The user task must know if the shared memory table entry was originally built with an ownername or a tasknumber, and must know what those values are.

Entry Conditions

Calling Sequence:

```
M.EXCL partition    {,ownername      }
                    {,tasknumber,TNUM}

(or)

LD      6,partition
LD      2,ownername     (or)    LW      2,tasknumber
                                ZR      3
SVC     1,X'79'         (or)    M.CALL  H.ALOC,14
```

where:

partition	contains a doubleword bounded, left-justified blank filled memory partition name, e.g., GLOBAL01
ownername	contains the owner name of the original owner of the partition
tasknumber	contains the left-justified tasknumber of the original owner of the partition
TNUM	indicates a task number is being used instead of an ownername

Exit Conditions

Return Sequence:

M.RTRN (or) abort user with AL39

Registers:

None

Abort Cases:

AL39	Shared memory table entry not found.

8.3.4 M.FADD - Permanent File Address Inquiry

The M.FADD service returns the number of 512W pages and beginning addresses of specified static or dynamic DATAPOOL or GLOBAL nn memory partition names. See Section 7.8.8.

8.3.5 M.FD - Free Dynamic Extended Indexed Data Space

The M.FD service allows the task to deallocate the most recently acquired extended memory map block, thus contracting its address space.

Entry Conditions

Calling Sequence:

```
M.FD

(or)

SVC     1,X'6A' (or) M.CALL  H.ALOC,9
```

Exit Conditions

Return Sequence:

```
M.RTRN  R3
```

Registers:

R3	new upper limit of extended memory
	0 if no extended memory left allocated

8.3.6 M.FE - Free Dynamic Task Execution Space

The M.FE service allows the task to dynamically deallocate the most recently acquired execution space map block, thus contracting its address space.

Entry Conditions

Calling Sequence:

```
        M.FE

        (or)

        SVC     1,X'68' (or) M.CALL  H.ALOC,11
```

Exit Conditions

Return Sequence:

```
        M.RTRN  R3 (or) abort user with AL20
```

Registers:

```
        R3      new upper address of execution space
```

Abort Cases:

```
        AL20    User attempted deallocation of TSA.
```

8.3.7 M.GADRL - Get Address Limits

The M.GADRL service returns to the caller the logical addresses associated with the boundaries of his task.

Entry Conditions

Calling Sequence:

```
M.GADRL

(or)

SVC     1,X'65' (or) M.CALL  H.MONS,41
```

Exit Conditions

Return Sequence:

```
M.RTRN  3,4,5,6,7
```

Registers:

R3	contains the logical word address of the first location of the task's DSECT (always on a page boundary)
R4	contains the logical word address of the last location in the DSECT actually loaded by the loader
R5	contains the logical word address of the last location currently available in the task's DSECT (always a map block boundary - 1W)
R6	contains the logical word address of the first location of the task's CSECT or COMMON allocation (always a map block boundary)
R7	contains the logical word address of the last location currently available in the task's extended indexed data space (always a map block boundary - 1W)

External References:

System Macros:

M.RTRN

Abort Cases:

None

Output Messages:

None

8.3.8 M.GD - Get Dynamic Extended Indexed Data Space

The M.GD service allows the task to dynamically acquire an additional map block of memory in its extended area. The memory will be of the same type specified when the task was cataloged. It will be mapped in a logically contiguous manner, with the first request map starting at 128KW. The task may call this service up to 15 times on a SYSTEMS 32/7x and 190 times on a CONCEPT/32, if sufficient memory exists, to expand its extended indexed data space. Alternatively, the task may choose to deallocate this space in the reverse order via M.FD. The task will be suspended until the allocation is successful.

Memory is allocated in 2KW increments on a CONCEPT/32 and in 8KW increments on a SYSTEMS 32/7x.

Entry Conditions

Calling Sequence:

```
M.GD

(or)

SVC     1,X'69' (or) M.CALL  H.ALOC,8
```

Exit Conditions

Return Sequence:

```
M.RTRN  R3,R4
```

Registers:

R3	logical address of allocated memory
	0 if allocation conflict (R4 = error code)
R4	ending logical word address of allocated memory or error code

Error Conditions:

R3 = 0	
R4 = 1	attempted allocation of an excessive number of map blocks
= 2	attempted allocation exceeds physical memory configured

8.3.9 M.GE - Get Dynamic Task Execution Space

The M.GE service allows the task to dynamically expand its memory allocation in map block increments, starting at the end of its DSECT up to the top of its logical address space. The additional memory will be of the same type specified when the task was cataloged. The task will be mapped in a logically contiguous manner up to the start of its CSECT or GLOBAL common, or 128KW, whichever occurs first. The task will be suspended until the allocation is successful.

Memory is allocated in 2KW increments on a CONCEPT/32 and in 8KW increments on a SYSTEMS 32/7x.

Entry Conditions

Calling Sequence:

```
M.GE

(or)

SVC     1,X'67' (or) M.CALL  H.ALOC,10
```

Exit Conditions

Return Sequence:

```
M.RTRN  R3,R4
```

Registers:

R3	starting logical address of new map block
R4	ending logical address of new map block

Error Conditions

R3 = 0	
R4 = 1	excessive DSECT allocation attempted
= 2	attempted allocation exceeds physical memory configured

8.3.10 M.INCL - Get Shared Memory

The M.INCL service allows a task to dynamically include a memory partition into its address space, e.g., GLOBAL01 or DATAPOOL common. The task is suspended until the inclusion is complete. M.INCL causes the allocation count (SMT.ACNT) and the use count (SMT.UCNT) to be incremented by one. The task performing an M.INCL must know if the shared memory table entry was built with an ownername or tasknumber, and know what they are.

Under memory-only MPX-32, shared memory applies only to static memory partitions or CSECTS.

Entry Conditions

Calling Sequence:

```
M.INCL partition  {,ownername}  ,[RW] , [password], denial ,[TNUM]
                  {,tasknumber}
```

(or)

```
LD      6,partition
LD      2,ownername                    (or)   LW      2,tasknumber
                                               ZR      3
ZR      0
ZR      4                (or)          LD      4,password
ZR      5
LA      0,denial
[SBR    0,0 if R/W]
SVC     1,X'72'                        (or)   M.CALL H.ALOC,13
```

where:

partition	contains the doubleword bounded, left-justified blank filled memory partition name, e.g., GLOBAL01
ownername	contains the owner name of the original owner of the partition
tasknumber	contains the left-justified task number of the original owner of the partition
RW	specifies read/write control desired. Default is read only.
password	contains a doubleword bounded, left-justified blank filled password. This must be the same as specified in CREATEM directive for partition (see MPX-32, Volume 2, Section 6.6.2).
denial	specifies a denial return address.
TNUM	indicates a task number is being used instead of an owner name

Exit Conditions

Return Sequence:

M.RTRN	R3

Registers:

R3	starting address of shared memory partition. This is a 20-bit address.
(or)	
M.RTNA	R0,R3 for denial returns
R3	1 if entry not found in shared memory table
	2 if invalid password specified
	3 if memory requirements conflict with task's address space
	4 if entry not found in shared memory table after returning from SWGQ state chain
R0	address to return to within user task body

8.3.11 M.SHARE - Share Memory with Another Task

The M.SHARE service dynamically creates a shared memory partition from the partition definition found on the System Master Directory (SMD). This definition must have been previously defined via the File Manager utility.

This service is invalid under memory-only MPX-32.

The call results in the creation of a new common area, which will be uniquely identified by the owner name or task number of the caller, and by the memory partition name. The memory type will be specified by the SMD definition. Pre-zeroing is not performed by this service. The partition is swappable with the task if the use count equals zero. The partition is deallocated when the allocation count equals zero. The task is suspended until the Shared Memory Table entry is built and the memory allocation is complete. The shared partition can be gated from the use of other tasks to allow the initial loading of data. This is called a data lock. Any tasks attempting to include this partition while the lock is set will be queued to the SWGQ state (general queue) and will remain there until the lock is reset by the M.SMULK service.

Options:

To request read/write access, set bit 0 in R0.

To request task number instead of owner name, set bit 1 in R0.

To request 'data lock' and includers to be enqueued, set bit 2 in R0.

Entry Conditions

Calling Sequence:

```
M.SHARE partition, [RW], [password] , [TNUM] , [LOCK]
(or)
LD      6,partition
ZR      0
SBR     0,0 if read/write
SBR     0,1 if task number requested
SBR     0,2 if data lock requested
LD      4,password
ZR      4 if no password
ZR      5 if no password
SVC     1,X'71'     (or)  M.CALL     H.ALOC,12
```

where:

partition	specifies the doubleword bounded, left justified memory partition name
RW	specifies read/write control.
password	specifies the doubleword bounded, left justified password. Password must be specified if the memory partition was created as either RO/PO.
TNUM	indicates a task number is being used instead of an owner name
LOCK	specifies a data lock

Exit Conditions

Return Sequence:

M.RTRN	R3	(or) abort user with AL40, AL41, or AL42

Registers:

R0 bit 4	reset if share becomes an include
R3	starting address of memory partition if share is successful

Abort Cases:

AL27	No free entries in SMT.
AL33	Shared memory definition conflicts with callers address space.
AL40	Partition definition not found on SMD.
AL41	SMD definition not a dynamic definition.
AL42	Invalid password for this partition.
AL46	Task attempted to share memory via a dynamic memory partition in a memory-only environment.
AL47	Dynamic memory partitions cannot be greater than 1 megabyte.

8.3.12 M.SMULK - Unlock and Dequeue Shared Memory

The M.SMULK service allows a user to unlock the data lock associated with a particular shared memory partition. See M.SHARE (ALOC,12) for use of data lock.

Upon executing M.SMULK, the lock on the shared area is reset and all users that are queued to the shared area are relinked from the SWGO (general queue wait state) to their appropriate run state. At this time they have full access to the shared partition.

This service is invalid under memory-only MPX-32.

Entry Conditions

Calling Sequence:

```
M.SMULK partition, ownername ,TNUM

    (or)

LD      R2,ownername    (or)    LW      R2,tasknumber
                                ZR      R3
SVC     1,X'1F'         (or)    M.CALL  H.ALOC,19
```

where:

partition	contains a doubleword bounded, left-justified blank filled memory partition name
ownername	contains the owner name of the original owner of the partition
tasknumber	contains the left-justified task number of the original owner of the partition
TNUM	indicates a task number is being used instead of an owner name

Exit Conditions

Return Sequence:

M.RTRN

Registers:

None

Abort Cases:

None

Output Messages:

None

APPENDIX A
MPX-32 DEVICE ACCESS

Throughout the reference manual, the generic descriptor 'devmnc' is used to indicate that a device can be specified.

Under MPX-32, device addresses are specified using a combination of three levels of identification. They are device type, device channel/controller address, and device address/subaddress.

A device can be specified using the generic device type only, which will result in allocation of the first available device of the type requested.

A second method of device specification is achieved by using the generic device type and specifying the channel/controller address. This results in allocation of the first available device of the type requested on the specified channel or controller.

The third method of device selection requires specification of the device type, channel/controller, and device address/subaddress. This method allows specification of a particular device.

1. Special Device Specifications and Handling

1.1 Magnetic Tape

For magnetic tape, a reel identifier, multivolume number, and unblocking can be part of the device mnemonic.

Syntax:

```
            ┌,reel          ┐
            │,reel,volume   │
lfc= device │,reel,volume,U │
            └,reel,,U       ┘
```

where:

device	is any one of the four levels of device specification described above.
reel	specifies a one- to four-character identifier for the reel. This parameter is required in batch. This parameter is not required in TSM and if not specified, the default is SCRA (Scratch).
volume	if multivolume tape, indicates volume number. Default: not multivolume (0).
U	the tape is optionally unblocked. Default: blocked.

Commas in this specification are significant. If an option is not specified, e.g., a reel identifier, but another option is specified, e.g., U, commas must be inserted for all non-specified options in between, e.g.,

MT1000,,,U

There must be no embedded blanks within the entire device mnemonic.

When the task is activated that has an assignment to tape, a MOUNT message indicates the name of the task and other information on the OPCOM console:

```
{TASK } ,taskname,taskno MOUNT reel VOL volume ON devmnc DEV,R,A,H?
{jobno}
```

where:

jobno	if the task is part of a batch job, identifies the job by job number.
taskname	is the name of the task to which the tape is assigned.
taskno	is the number of the task.

reel — if the assignment is a multivolume tape, indicates the reel identifier specified in the assignment. This parameter is required in batch. This parameter is not required in TSM and if not specified, the default is SCRA.

volume — identifies the volume number to mount if multivolume tape.

devmnc — is the device mnemonic for the tape unit selected in response to the assignment. If a specific channel and subaddress are supplied in the assignment, the specific tape drive is selected and named in the message. Otherwise, a unit is selected by the system and its complete address is named in the message.

DEV,R,A,H — the device listed in the message can be allocated and the task resumed (R), a different device can be selected (DEV), the task can be aborted (A), or the task can be held with the specified device deallocated (H). If an 'R' response is given and a high speed XIO tape drive is being used, its density can be changed when the software select feature is enabled on the tape unit front panel. If specified, it will override any specification made at assignment. Values are:

N or 800	indicates 800 bpi nonreturn to zero inverted (NRZI)
P or 1600	indicates 1600 bpi phase encoded (PE)
G or 6250	indicates 6250 bpi group coded recording (GCR) Default.

Example usage: RN, R1600, etc.

Note: Do not insert blanks or commas.

Response:

To indicate the drive specified in the MOUNT message is ready and proceed with the task, mount the tape on the drive and type R (Resume), optionally followed by a density specification if the drive is a high speed XIO tape unit. To abort the task, type A (Abort). To hold the task and deallocate the specified device, type H (Hold). The task can then be resumed by the OPCOM CONTINUE command, at which time a tape drive will be selected by the system and the MOUNT message redisplayed.

To select a tape drive other than the drive specified in the message, enter the mnemonic of the drive you want to use. Any of the three levels of device identification can be used. The MOUNT message is reissued. Mount the tape and type R if satisfactory, or if not satisfactory, abort, override, or hold as just described.

1.2 Temporary Disc File Size

For a temporary disc file, size must be specified and unblocking is optional.

Syntax:

lfc = device,size [,U]

where:

size	specifies the number of 192-word blocks required.
U	the file is optionally unblocked. Default: blocked.

Examples of the three methods of device specification follow:

Type 1 - Generic Device Class

$ASSIGN3 DEV=M9,,1

In this example, the device assigned to logical file code (lfc) "DEV" will be any 9-track tape unit on any channel. The multivolume reel number is 1. The reel identifier is SCRA.

Type 2 - Generic Device Class and Channel/Controller

$ASSIGN3 DEV=M910,MORK,,U

In this example, the device assigned to logical file code (lfc) "DEV" will be the first available 9-track tape unit on channel 10. The specification is invalid if a 9-track tape unit does not exist on the channel. The reel identifier is supplied. This is not a multivolume tape. It is, however, unblocked.

Type 3 - Specific Device Request

$ASSIGN3 DEV=M91001

In this example, the device assigned to logical file code (lfc) "DEV" will be the 9-track tape unit 01 on channel 10. The specification is invalid if unit 01 on channel 10 is not a 9-track tape. The tape reel identifier is SCRA; the tape is blocked and is not multivolume.

2. GPMC Devices

GPMC/GPDC device specifications are in keeping with the general structure just described. For instance, the terminal at subaddress 04 on GPMC 01 whose channel address is 20 would be identified as follows:

```
$ASSIGN3 DEV=TY2004
```

3. NULL Device

A special device type "NU" is available for NULL device specifications. Files accessed using this device type generate an end-of-file (EOF) upon attempt to read and normal completion upon attempt to write.

4. OPCOM Console

Logical file codes are assigned to the OPCOM console by using the device type "CT".

5. Special System Files

There are four special mnemonics provided for access to special system files: SLO, SBO, SGO and SYC. These are assigned via the $ASSIGN2 statement, as is:

```
$ASSIGN2 OUT=SLO,printlines
```

For non-batch tasks, SLO and SBO files are allocated dynamically by the system and used to disc buffer output to a device selected automatically. For batch tasks, use of SLO and SBO files is identical, except that automatic selection of a device can be overridden by assigning a specific file or device.

SGO and SYC assignments are used for batch processing. See Section 7.6.

Dev Type Code	Device	Device Description
00	CT	Operator Console (Not Assignable)
01	DC	Any Disc Unit
02	DM	Any Moving Head Disc
03	DF	Any Fixed Head Disc
04	MT	Any Magnetic Tape Unit
05	M9	Any 9-Track Magnetic Tape Unit
06	M7	Any 7-Track Magnetic Tape Unit
07	CD	Any Card Reader-Punch
08	CR	Any Card Reader
09	CP	Any Card Punch
0A	LP	Any Line Printer
0B	PT	Any Paper Tape Reader-Punch
0C	TY	Any Teletypewriter (Other than Console)
0D	CT	Operator Console (Assignable)
0E	FL	Floppy Disc
0F	NU	Null Device
10	CA	Communications Adapter (Binary Synchronous/Asynchronous)
11	U0	Available for user-defined applications
12	U1	Available for user-defined applications
13	U2	Available for user-defined applications
14	U3	Available for user-defined applications
15	U4	Available for user-defined applications
16	U5	Available for user-defined applications
17	U6	Available for user-defined applications
18	U7	Available for user-defined applications
19	U8	Available for user-defined applications
1A	U9	Available for user-defined applications
1B	LF	Line Printer/Floppy Controller (used only with SYSGEN)

Table A-1: Device Type Codes

6. Samples

A description of device selection possibilities would be constructed as follows:

DISC

DC	Any Disc
DM	Any Moving Head Disc
DM08	Any Moving Head Disc on Channel 08
DM0801	Moving Head Disc 01 on Channel 08
DF	Any Fixed Head Disc
DF04	Any Fixed Head Disc on Channel 04
DF0401	Fixed Head Disc 01 on Channel 04

TAPE

MT	Any Magnetic Tape
M9	Any 9-track Magnetic Tape
M910	Any 9-track Magnetic Tape on Channel 10
M91002	9-track Magnetic Tape 02 on Channel 10
M7	Any 7-track Magnetic Tape
M712	Any 7-track Magnetic Tape on Channel 12
M71201	7-track Magnetic Tape 01 on Channel 12

CARD EQUIPMENT

CD	Any Card Reader-Punch
CR	Any CR
CR78	Any CR on Channel 78
CR7800	CR on Channel 78 Subaddress 00
CP	Any CP
CP7C	Any CP on Channel 7C
CP7C00	CP on Channel 7C Subaddress 00

LINE PRINTER

LP	Any LP
LP7A	Any LP on Channel 7A
LP7A00	LP on Channel 7A Subaddress 00

APPENDIX B
SYSTEM SERVICES CROSS REFERENCE CHARTS

USER LEVEL SYSTEM SERVICES - MACRO NAME

	MACRO	DESCRIPTOR	SVC 1,XX	MODULE, E.P.	REF MANUAL SECTION
	M.ACTV	ACTIVATE TASK	52	H.MONS,15	8.2.1
	M.ADRS	MEMORY ADDRESS INQUIRY	44	H.MONS,3	8.3.1
*	M.ALOC	ALLOCATE FILE OR PERIPHERAL DEVICE	40	H.MONS,21	7.8.1
	M.ANYW	WAIT FOR ANY MSG, END ACTION, OR BRK	7C	H.MONS,37	8.2.2
	M.ASYNCH	SET ASYNCHRONOUS TASK INTERRUPT	1C	H.MONS,68	8.2.3
	M.BACK	BACKSPACE RECORD	35	H.IOCS,9	7.8.2
		OR			
		FILE	36	H.IOCS,19	
	M.BORT	ABORT SPEC. TASK	56	H.MONS,19	8.2.4
		OR			
		SELF	57	H.MONS,20	
		OR			
		WITH EXT. MESSAGE	62	H.MONS,28	
	M.BRK	BREAK/TASK INTERRUPT LINK	6E	H.MONS,46	8.2.5
	M.BRKXIT	EXIT FROM TASK INTERRUPT LEVEL	70	H.MONS,48	8.2.6
**	M.CDJS	SUBMIT JOB FROM DISC FILE	61	H.MONS,27	8.2.7
	M.CLSE	CLOSE FILE	39	H.IOCS,23	7.8.3
	M.CONADB	CONVERT ASCII DECIMAL TO BINARY	28	H.TSM,7	5.6.3.1
	M.CONAHB	CONVERT ASCII HEX TO BINARY	29	H.TSM,8	5.6.3.2
	M.CONBAD	CONVERT BINARY TO ASCII DECIMAL	2A	H.TSM,9	5.6.3.3

*Reduced functionality under Memory-Only MPX-32
**Not supported under Memory-Only MPX-32

USER LEVEL SYSTEM SERVICES - MACRO NAME

(CONTINUED)

	MACRO	DESCRIPTOR	SVC 1,XX	MODULE, E.P.	REF MANUAL SECTION
	M.CONBAH	CONVERT BINARY TO ASCII HEX	2B	H.TSM,10	5.6.3.4
	M.CONN	CONNECT TASK TO INTERRUPT	4B	H.MONS,10	8.2.8
**	M.CREATE	CREATE PERM FILE	75	H.FISE,12	7.8.4
	M.CWAT	SYSTEM CONSOLE WAIT	3D	H.IOCS,26	7.8.5
*	M.DALC	DEALLOCATE FILE OR DEVICE	41	H.MONS,22	7.8.6
	M.DATE	DATE AND TIME INQUIRY	15	H.MONS,70	8.2.9
**	M.DEBUG	LOAD AND EXECUTE INTERACTIVE DEBUGGER	63	H.MONS,29	8.2.10
**	M.DELETE	DELETE PERM FILE	77	H.FISE,14	7.8.7
	M.DELTSK	DELETE TASK	5A	H.MONS,31	8.2.11
	M.DEVID	GET DEVICE MNEMONIC OR TYPE CODE	14	H.MONS,71	8.2.12
	M.DISCON	DISCONNECT TASK FROM INTERRUPT	5D	H.MONS,38	8.2.13
	M.DLTT	DELETE TIMER ENTRY	47	H.MONS,6	8.2.14
	M.DSMI	DISABLE MESSAGE TASK INTERRUPT	2E	H.MONS,57	8.2.15
	M.DSUB	DISABLE USER BREAK INTERRUPT	12	H.MONS,73	8.2.16
**	M.DUMP	MEMORY DUMP REQUEST	4F	H.MONS,12	8.3.2
	M.EAWAIT	END ACTION WAIT	1D	H.EXEC,40	8.2.17
	M.ENMI	ENABLE MESSAGE TASK INTERRUPT	2F	H.MONS,58	8.2.18

*Reduced functionality under Memory-Only MPX-32
**Not supported under Memory-Only MPX-32

USER LEVEL SYSTEM SERVICES - MACRO NAME

(CONTINUED)

	MACRO	DESCRIPTOR	SVC 1,XX	MODULE, E.P.	REF MANUAL SECTION
	M.ENUB	ENABLE USER BREAK INTERRUPT	13	H.MONS,72	8.2.19
	M.EXCL	FREE SHARED MEMORY	79	H.ALOC,14	8.3.3
	M.EXIT	TERMINATE TASK EXECUTION	55	H.MONS,18	8.2.20
**	M.FADD	PERMANENT FILE ADDRESS INQUIRY	43	H.MONS,2	7.8.8
	M.FD	FREE DYNAMIC EXTENDED INDEX DATA SPACE	6A	H.ALOC,9	8.3.5
	M.FE	FREE DYNAMIC TASK EXECUTION SPACE	68	H.ALOC,11	8.3.6
	M.FILE	OPEN FILE	30	H.IOCS,1	7.8.9
**	M.FSLR	RELEASE SYNCHRONIZATION FILE LOCK	24	H.FISE,25	7.8.10
**	M.FSLS	SET SYNCHRONIZATION FILE LOCK	23	H.FISE,24	7.8.11
	M.FWRD	ADVANCE RECORD	33	H.IOCS,7	7.8.12
		OR FILE	34	H.IOCS,8	
	M.FXLR	RELEASE EXCLUSIVE FILE LOCK	22	H.FISE,23	7.8.13
**	M.FXLS	SET EXCLUSIVE FILE LOCK	21	H.FISE,22	7.8.14
	M.GADRL	GET ADDRESS LIMITS	65	H.MONS,41	8.3.7
	M.GD	GET DYNAMIC EXTENDED INDEXED DATA SPACE	69	H.ALOC,8	8.3.8
	M.GE	GET DYNAMIC TASK EXECUTION SPACE	67	H.ALOC,10	8.3.9

*Reduced functionality under Memory-Only MPX-32
**Not supported under Memory-Only MPX-32

USER LEVEL SYSTEM SERVICES - MACRO NAME

(CONTINUED)

	MACRO	DESCRIPTOR	SVC 1,XX	MODULE, E.P.	REF MANUAL SECTION
	M.GMSGP	GET MSG PARAMETERS	7A	H.MONS,35	8.2.21
	M.GRUNP	GET RUN PARAMETERS	7B	H.MONS,36	8.2.22
	M.HOLD	PROGRAM HOLD REQUEST	58	H.MONS,25	8.2.23
	M.ID	GET TASK NUMBER	64	H.MONS,32	8.2.24
*	M.INCL	GET SHARED MEMORY	72	H.ALOC,13	8.3.10
	M.INT	ACTIVATE TASK INTERRUPT	6F	H.MONS,47	8.2.25
**	M.LOG	PERMANENT FILE LOG	73	H.MONS,33	7.8.15
	M.MYID	GET TASK NUMBER	64	H.MONS,32	8.2.26
**	M.OLAY	LOAD OVERLAY	50	H.MONS,13	8.2.27
		OR			
		LOAD AND EXECUTE OVERLAY	51	H.MONS,14	
**	M.PDEV	PHYSICAL DEVICE INQUIRY	42	H.MONS,1	7.8.16
**	M.PERM	CHANGE TEMP FILE TO PERMANENT	76	H.FISE,13	7.8.17
	M.PGOW	TASK OPTION WORD INQUIRY	4C	H.MONS,24	8.2.28
	M.PRIL	CHANGE PRIORITY LEVEL	4A	H.MONS,9	8.2.29
	M.PTSK	PARAMETER TASK ACTIVATION	5F	H.MONS,40	8.2.30
	M.RCVR	RECEIVE MESSAGE LINK ADDRESS	6B	H.MONS,43	8.2.31
	M.READ	READ RECORD	31	H.IOCS,3	7.8.18
	M.RELP	RELEASE DUAL PORTED DISC	27	H.IOCS,27	7.8.19

*Reduced functionality under Memory-Only MPX-32
**Not supported under Memory-Only MPX-32

USER LEVEL SYSTEM SERVICES - MACRO NAME

(CONTINUED)

	MACRO	DESCRIPTOR	SVC 1,XX	MODULE, E.P.	REF MANUAL SECTION
	M.RESP	RESERVE DUAL PORTED DISC	26	H.IOCS,24	7.8.20
	M.RRES	RELEASE CHANNEL	3B	H.IOCS,13	7.8.21
	M.RSML	RESOURCEMARK LOCK	19	H.MONS,62	7.8.22
	M.RSMU	RESOURCEMARK UNLOCK	1A	H.MONS,63	7.8.23
	M.RSRV	RESERVE CHANNEL	3A	H.IOCS,12	7.8.24
	M.RWND	REWIND FILE	37	H.IOCS,2	7.8.25
	M.SETS	SET USER STATUS WORD	48	H.MONS,7	8.2.32
	M.SETT	CREATE TIMER ENTRY	45	H.MONS,4	8.2.33
*	M.SHARE	SHARE MEMORY WITH ANOTHER TASK	71	H.ALOC,12	8.3.11
	M.SMSGR	SEND MESSAGE TO SPECIFIED TASK	6C	H.MONS,44	8.2.34
	M.SMULK	UNLOCK AND DEQUEUE SHARED MEMORY	1F	H.ALOC,19	8.3.12
	M.SRUNR	SEND RUN REQUEST	6D	H.MONS,45	8.2.35
	M.SUAR	SET USER ABORT RECEIVER ADDRESS	60	H.MONS,26	8.2.36
	M.SUME	RESUME TASK EXECUTION	53	H.MONS,16	8.2.37
	M.SUSP	SUSPEND TASK EXECUTION	54	H.MONS,17	8.2.38
	M.SYNCH	SET SYNCHRONOUS TASK INTERRUPT	1B	H.MONS,67	8.2.39
	M.TBRKON	TRAP ONLINE USER'S TASK	5C	H.TSM,6	5.6.2

*Reduced functionality under Memory-Only MPX-32
**Not supported under Memory-Only MPX-32

USER LEVEL SYSTEM SERVICES - MACRO NAME

(CONTINUED)

	MACRO	DESCRIPTOR	SVC 1,XX	MODULE, E.P.	REF MANUAL SECTION
	M.TDAY	TIME-OF-DAY INQUIRY	4E	H.MONS,11	8.2.40
**	M.TSCAN	SCAN TERMINAL INPUT BUFFER	5B	H.TSM,2	5.6.1
	M.TSTE	ARITHMETIC EXCEPTION INQUIRY	4D	H.MONS,23	8.2.41
	M.TSTS	TEST USER STATUS WORD	49	H.MONS,8	8.2.42
	M.TSTT	TEST TIMER ENTRY	46	H.MONS,5	8.2.43
	M.TURNON	ACTIVATE PROGRAM AT GIVEN TIME OF DAY	1E	H.MONS,66	8.2.44
	M.TYPE	CONSOLE TYPE	3F	H.IOCS,14	7.8.26
	M.UPSP	UPSPACE	10	H.IOCS,20	7.8.27
	M.USER	USERNAME SPECIFICATION	74	H.MONS,34	7.8.28
	M.WAIT	WAIT I/O	3C	H.IOCS,25	7.8.29
	M.WEOF	WRITE EOF	38	H.IOCS,5	7.8.30
	M.WRIT	WRITE RECORD	32	H.IOCS,4	7.8.31
	M.XBRKR	EXIT FROM TASK INTERRUPT LEVEL	70	H.MONS,48	8.2.45
	M.XIEA	NO-WAIT I/O END ACTION RETURN	2C	H.IOCS,34	7.8.32
	M.XMEA	EXIT FROM MESSAGE END ACTION ROUTINE	7E	H.MONS,50	8.2.46
	M.XMSGR	EXIT FROM MESSAGE RECEIVER	5E	H.MONS,39	8.2.47
	M.XREA	EXIT RUN REQUEST END ACTION ROUTINE	7F	H.MONS,51	8.2.48

**Not supported under Memory-Only MPX-32

USER LEVEL SYSTEM SERVICES - MACRO NAME

(CONTINUED)

	MACRO	DESCRIPTOR	SVC 1,XX	MODULE, E.P.	REF MANUAL SECTION
	M.XRUNR	EXIT RUN RECEIVER	7D	H.MONS,49	8.2.49
	M.XTIME	TASK CPU EXECUTION TIME	2D	H.MONS,65	8.2.50
	N/A	ERASE OR PUNCH TRAILER	3E	H.IOCS,21	7.8.33
**	N/A	DEBUG LINK SERVICE	66	H.MONS,42	8.2.51
	N/A	EXECUTE CHANNEL PROGRAM	25	H.IOCS,10	7.8.34
	N/A	RELEASE FHD PORT	27	H.IOCS,27	7.8.35
	N/A	RESERVE FHD PORT	26	H.IOCS,24	7.8.36
	N/A	SET TABS IN UDT	59	H.TSM,5	N/A

**Not supported under Memory-Only MPX-32

USER LEVEL SYSTEM SERVICES - ALPHABETIC

		SVC 1,XX	MODULE	E.P.	REF MANUAL SECTION
ABORT SELF	M.BORT	57	H.MONS	20	8.2.4
ABORT SPECIFIED TASK	M.BORT	56	H.MONS	19	8.2.4
ABORT WITH EXTENDED MESSAGE	M.BORT	62	H.MONS	28	8.2.4
ACTIVATE PROGRAM AT GIVEN TIME OF DAY	M.TURNON	1E	H.MONS	66	8.2.44
ACTIVATE TASK INTERRUPT	M.INT	6F	H.MONS	47	8.2.25
ACTIVATE TASK	M.ACTV	52	H.MONS	15	8.2.1
ADVANCE FILE OR RECORD	M.FWRD	34 33	H.IOCS H.IOCS	8 7	7.8.12 7.8.12
ALLOCATE FILE OR PERIPHERAL DEVICE	M.ALOC	40	H.MONS	21	7.8.1
ARITHMETIC EXCEPTION INQUIRY	M.TSTE	4D	H.MONS	23	8.2.41
BACKSPACE FILE OR RECORD	M.BACK	36 35	H.IOCS H.IOCS	19 9	7.8.2 7.8.2
BREAK/TASK INTERRUPT LINK	M.BRK	6E	H.MONS	46	8.2.5
CHANGE PRIORITY LEVEL (PRIV)	M.PRIL	4A	H.MONS	9	8.2.29
CHANGE TEMP FILE TO PERMANENT	M.PERM	76	H.FISE	13	7.8.17
CLOSE FILE	M.CLSE	39	H.IOCS	23	7.8.3
CONNECT TASK TO INTERRUPT	M.CONN	4B	H.MONS	10	8.2.8
CONSOLE TYPE	M.TYPE	3F	H.IOCS	14	7.8.26
CONVERT ASCII DECIMAL TO BINARY	M.CONADB	28	H.TSM	7	5.6.3.1

UNPRIVILEGED USER LEVEL SYSTEM SERVICES - ALPHABETIC

(CONTINUED)

		SVC 1,XX	MODULE	E.P.	REF MANUAL SECTION
CONVERT ASCII HEX TO BINARY	M.CONAHB	29	H.TSM	8	5.6.3.2
CONVERT BINARY TO ASCII DECIMAL	M.CONBAD	2A	H.TSM	9	5.6.3.3
CONVERT BINARY TO ASCII HEX	M.CONBAH	2B	H.TSM	10	5.6.3.4
CREATE PERM FILE	M.CREATE	75	H.FISE	12	7.8.4
CREATE TIMER ENTRY	M.SETT	45	H.MONS	4	8.2.33
*DATE AND TIME INQUIRY	M.DATE	15	H.MONS	70	8.2.9
DEALLOCATE FILE OR DEVICE	M.DALC	41	H.MONS	22	7.8.6
DEBUG LINK SERVICE	N/A	66	H.MONS	42	8.2.51
DELETE PERM FILE	M.DELETE	77	H.FISE	14	7.8.7
DELETE TASK	M.DELTSK	5A	H.MONS	31	8.2.11
DELETE TIMER ENTRY	M.DLTT	47	H.MONS	6	8.2.14
DISABLE MESSAGE TASK INTERRUPT	M.DSMI	2E	H.MONS	57	8.2.15
*DISABLE USER BREAK INTERRUPT	M.DSUB	12	H.MONS	73	8.2.16
DISCONNECT TASK FROM INTERRUPT	M.DISCON	5D	H.MONS	38	8.2.13
ENABLE MESSAGE TASK INTERRUPT	M.ENMI	2F	H.MONS	58	8.2.18
*ENABLE USER BREAK INTERRUPT	M.ENUB	13	H.MONS	13	8.2.19
END ACTION WAIT	M.EAWAIT	1D	H.EXEC	40	8.2.17
ERASE OR PUNCH TRAILER	N/A	3E	H.IOCS	21	7.8.33

*NEW

UNPRIVILEGED USER LEVEL SYSTEM SERVICES - ALPHABETIC

(CONTINUED)

		SVC 1,XX	MODULE	E.P.	REF MANUAL SECTION
EXECUTE CHANNEL PROGRAM	N/A	25	H.IOCS	10	7.8.34
EXIT FROM MSG RCVR	M.XMSGR	5E	H.MONS	39	8.2.47
EXIT FROM MSG END ACTION ROUTINE	M.XMEA	7E	H.MONS	50	8.2.46
EXIT FROM TASK INTERRUPT LEVEL	M.BRKXIT M.XBRKR	70	H.MONS	48	8.2.6 8.2.45
EXIT RUN RCVR	M.XRUNR	7D	H.MONS	49	8.2.49
EXIT FROM RUN REQUEST END ACTION ROUTINE	M.XREA	7F	H.MONS	51	8.2.48
FREE DYNAMIC EXT INDEX DATA SPACE	M.FD	6A	H.ALOC	9	8.3.5
FREE DYNAMIC TASK EXECUTION SPACE	M.FE	68	H.ALOC	11	8.3.6
FREE SHARED MEMORY	M.EXCL	79	H.ALOC	14	8.3.3
GET ADDRESS LIMITS	M.GADRL	65	H.MONS	41	8.3.7
*GET DEVICE MNEMONIC OR TYPE CODE	M.DEVID	14	H.MONS	71	8.2.12
GET DYNAMIC EXTENDED INDEXED DATA SPACE	M.GD	69	H.ALOC	8	8.3.8
GET DYNAMIC TASK EXECUTION SPACE	M.GE	67	H.ALOC	10	8.3.9
GET MSG PARAMETERS	M.GMSGP	7A	H.MONS	35	8.2.21
GET RUN PARAMETERS	M.GRUNP	7B	H.MONS	36	8.2.22
GET SHARED MEMORY	M.INCL	72	H.ALOC	13	8.3.10
GET TASK NUMBER	M.ID M.MYID	64	H.MONS	32	8.2.24 8.2.26

*NEW

UNPRIVILEGED USER LEVEL SYSTEM SERVICES - ALPHABETIC

(CONTINUED)

		SVC 1,XX	MODULE	E.P.	REF MANUAL SECTION
LOAD AND EXECUTE INTERACTIVE DEBUGGER	M.DEBUG	63	H.MONS	29	8.2.10
LOAD OVERLAY AND LOAD AND EXECUTE OVERLAY	M.OLAY	5O 51	H.MONS H.MONS	13 14	8.2.27 8.2.27
MEMORY ADDRESS INQUIRY	M.ADRS	44	H.MONS	3	8.3.1
MEMORY DUMP REQUEST	M.DUMP	4F	H.MONS	12	8.3.2
NO-WAIT I/O END ACTION RETURN	M.XIEA	2C	H.IOCS	34	7.8.32
OPEN FILE	M.FILE	30	H.IOCS	1	7.8.9
PARAMETER TASK ACTIVATION (PRIV)	M.PTSK	5F	H.MONS	40	8.2.30
PERMANENT FILE ADDRESS INQUIRY	M.FADD	43	H.MONS	2	7.8.8
PERMANENT FILE LOG	M.LOG	73	H.MONS	33	7.8.15
PHYSICAL DEVICE INQUIRY	M.PDEV	42	H.MONS	1	7.8.16
PROGRAM HOLD REQUEST	M.HOLD	58	H.MONS	25	8.2.23
READ RECORD	M.READ	31	H.IOCS	3	7.8.18
RECEIVE MESSAGE LINK ADDRESS	M.RCVR	6B	H.MONS	43	8.2.31
RELEASE CHANNEL (PRIV)	M.RRES	3B	H.IOCS	13	7.8.21
RELEASE DUAL PORTED DISC	M.RELP	27	H.IOCS	27	7.8.19
RELEASE EXCLUSIVE FILE LOCK	M.FXLR	22	H.FISE	23	7.8.13
RELEASE FHD PORT (PRIV)	N/A	27	H.IOCS	27	7.8.35

*NEW

UNPRIVILEGED USER LEVEL SYSTEM SERVICES - ALPHABETIC

(CONTINUED)

		SVC 1,XX	MODULE	E.P.	REF MANUAL SECTION
RELEASE SYNCHRONIZATION FILE LOCK	M.FSLR	24	H.FISE	25	7.8.10
RESERVE CHANNEL (PRIV)	M.RSRV	3A	H.IOCS	12	7.8.24
*RESERVE DUAL PORTED DISC	M.RESP	26	H.IOCS	24	7.8.20
RESERVE FHD PORT (PRIV)	N/A	26	H.IOCS	24	7.8.36
RESOURCEMARK LOCK	M.RSML	19	H.MONS	62	7.8.22
RESOURCEMARK UNLOCK	M.RSMU	1A	H.MONS	63	7.8.23
RESUME TASK EXECUTION	M.SUME	53	H.MONS	16	8.2.37
REWIND FILE	M.RWND	37	H.IOCS	2	7.8.25
SCAN TERMINAL INPUT BUFFER	M.TSCAN	5B	H.TSM	2	5.6.1
SEND MESSAGE TO SPECIFIED TASK	M.SMSGR	6C	H.MONS	44	8.2.34
SEND RUN REQUEST	M.SRUNR	6D	H.MONS	45	8.2.35
SET ASYNCHRONOUS TASK INTERRUPT	M.ASYNCH	1C	H.MONS	68	8.2.3
SET EXCLUSIVE FILE LOCK	M.FXLS	21	H.FISE	22	7.8.14
SET SYNCHRONIZATION FILE LOCK	M.FSLS	23	H.FISE	24	7.8.11
SET SYNCHRONOUS TASK INTERRUPT	M.SYNCH	1B	H.MONS	67	8.2.39
SET TABS IN UDT	N/A	59	H.TSM	5	N/A
SET USER STATUS WORD	M.SETS	48	H.MONS	7	8.2.32
SET USER ABORT RECEIVER ADDRESS	M.SUAR	60	H.MONS	26	8.2.36

*NEW

UNPRIVILEGED USER LEVEL SYSTEM SERVICES - ALPHABETIC

(CONTINUED)

		SVC 1,XX	MODULE	E.P.	REF MANUAL SECTION
SHARE MEMORY WITH ANOTHER TASK	M.SHARE	71	H.ALOC	12	8.3.11
SUBMIT JOB FROM DISC FILE	M.CDJS	61	H.MONS	27	8.2.7
SUSPEND TASK EXECUTION	M.SUSP	54	H.MONS	17	8.2.38
SYSTEM CONSOLE WAIT	M.CWAT	3D	H.IOCS	26	7.8.5
TASK CPU EXECUTION TIME	M.XTIME	2D	H.MONS	65	8.2.50
TASK OPTION WORD INQUIRY	M.PGOW	4C	H.MONS	24	8.2.28
TERMINATE TASK EXECUTION	M.EXIT	55	H.MONS	18	8.2.20
TEST TIMER ENTRY	M.TSTT	46	H.MONS	5	8.2.43
TEST USER STATUS WORD	M.TSTS	49	H.MONS	8	8.2.42
TIME-OF-DAY INQUIRY	M.TDAY	4E	H.MONS	11	8.2.40
TRAP ONLINE USER'S TASK	M.TBRKON	5C	H.TSM	6	5.6.2
UNLOCK AND DEQUEUE SHARED MEMORY	M.SMULK	1F	H.ALOC	19	8.3.12
*UPSPACE	M.UPSP	10	H.IOCS	20	7.8.27
USERNAME SPECIFICATION	M.USER	74	H.MONS	34	7.8.28
WAIT I/O	M.WAIT	3C	H.IOCS	25	7.8.29
WAIT FOR ANY MSG, END ACTION, OR BRK	M.ANYW	7C	H.MONS	37	8.2.2
WRITE EOF	M.WEOF	38	H.IOCS	5	7.8.30
WRITE RECORD	M.WRIT	32	H.IOCS	4	7.8.31

*NEW

USER LEVEL SYSTEM SERVICES - SVC ORDER

SVC 1	DESCRIPTION	MODULE,EP	MACRO	REF MANUAL SECTION
00-0F	RESERVED			
10	UPSPACE	H.IOCS,20	M.UPSP	7.8.27
11	RESERVED			
12	DISABLE USER BREAK INTERRUPT	H.MONS,73	M.DSUB	8.2.16
13	ENABLE USER BREAK INTERRUPT	H.MONS,72	M.ENUB	8.2.19
14	GET DEVICE MNEMONIC OR TYPE CODE	H.MONS,71	M.DEVID	8.2.12
15	DATE AND TIME INQUIRY	H.MONS,70	M.DATE	8.2.9
16	RESERVED			
17	ADI I/O	H.ADIO,8	N/A	N/A
18	ADI EAI	H.ADIO,9	N/A	N/A
19	RESOURCEMARK LOCK	H.MONS,62	M.RSML	7.8.22
1A	RESOURCEMARK UNLOCK	H.MONS,63	M.RSMU	7.8.23
1B	SET SYNCHRONOUS TASK INTERRUPT	H.MONS,67	M.SYNCH	8.2.39
1C	SET ASYNCHRONOUS TASK INTERRUPT	H.MONS,68	M.ASYNCH	8.2.3
1D	END ACTION WAIT	H.EXEC,40	M.EAWAIT	8.2.17
1E	ACTIVATE PROGRAM AT GIVEN TIME OF DAY	H.MONS,66	M.TURNON	8.2.44
1F	UNLOCK AND DEQUEUE SHARED MEMORY	H.ALOC,19	M.SMULK	8.3.12
20	RESERVED			
21	SET EXCLUSIVE FILE LOCK	H.FISE,22	M.FXLS	7.8.14

USER LEVEL SYSTEM SERVICES - SVC ORDER (CONTINUED)

SVC 1	DESCRIPTION	MODULE,EP	MACRO	REF MANUAL SECTION
22	RELEASE EXCLUSIVE FILE LOCK	H.FISE,23	M.FXLR	7.8.13
23	SET SYNCHRONIZATION FILE LOCK	H.FISE,24	M.FSLS	7.8.11
24	RELEASE SYNCHRONIZATION FILE LOCK	H.FISE,25	M.FSLR	7.8.10
25	EXECUTE CHANNEL PROGRAM	H.IOCS,10	N/A	7.8.34
26	RESERVE FHD PORT RESERVE DUAL PORTED DISC	H.IOCS,24 H.IOCS,24	N/A M.RESP	7.8.36 7.8.20
27	RELEASE FHD PORT RELEASE DUAL PORTED DISC	H.IOCS,27 H.IOCS,27	N/A M.RELP	7.8.35 7.8.19
28	CONVERT ASCII DECIMAL TO BINARY	H.TSM,7	M.CONADB	5.6.3.1
29	CONVERT ASCII HEX TO BINARY	H.TSM,8	M.CONAHB	5.6.3.2
2A	CONVERT BINARY TO ASCII DECIMAL	H.TSM,9	M.CONBAD	5.6.3.3
2B	CONVERT BINARY TO ASCII HEX	H.TSM,10	M.CONBAH	5.6.3.4
2C	NO-WAIT I/O END ACTION RETURN	H.IOCS,34	M.XIEA	7.8.32
2D	TASK CPU EXECUTION TIME	H.MONS,65	M.XTIME	8.2.50
2E	DISABLE MESSAGE TASK INTERRUPT	H.MONS,57	M.DSMI	8.2.15

USER LEVEL SYSTEM SERVICES - SVC ORDER

(CONTINUED)

SVC 1	DESCRIPTION	MODULE,EP	MACRO	REF MANUAL SECTION
2F	ENABLE MESSAGE TASK INTERRUPT	H.MONS,58	M.ENMI	8.2.18
30	OPEN FILE	H.IOCS,1	M.FILE	7.8.9
31	READ RECORD	H.IOCS,3	M.READ	7.8.18
32	WRITE RECORD	H.IOCS,4	M.WRIT	7.8.31
33	ADVANCE RECORD	H.IOCS,7	M.FWRD	7.8.12
34	ADVANCE FILE	H.IOCS,8	M.FWRD	7.8.12
35	BACKSPACE RECORD	H.IOCS,9	M.BACK	7.8.2
36	BACKSPACE FILE	H.IOCS,19	M.BACK	7.8.2
37	REWIND FILE	H.IOCS,2	M.RWND	7.8.25
38	WRITE EOF	H.IOCS,5	M.WEOF	7.8.30
39	CLOSE FILE	H.IOCS,23	M.CLSE	7.8.3
3A	RESERVE CHANNEL (PRIV)	H.IOCS,12	M.RSRV	7.8.24
3B	RELEASE CHANNEL (PRIV)	H.IOCS,13	M.RRES	7.8.21
3C	WAIT I/O	H.IOCS,25	M.WAIT	7.8.29
3D	SYSTEM CONSOLE WAIT	H.IOCS,26	M.CWAT	7.8.5
3E	ERASE OR PUNCH TRAILER	H.IOCS,21	N/A	7.8.33
3F	CONSOLE TYPE	H.IOCS,14	M.TYPE	7.8.26
40	ALLOCATE FILE OR PERIPHERAL DEVICE	H.MONS,21	M.ALOC	7.8.1
41	DEALLOCATE FILE OR DEVICE	H.MONS,22	M.DALC	7.8.6
42	PHYSICAL DEVICE INQUIRY	H.MONS,1	M.PDEV	7.8.16

USER LEVEL SYSTEM SERVICES - SVC ORDER

(CONTINUED)

SVC 1	DESCRIPTION	MODULE,EP	MACRO	REF MANUAL SECTION
43	PERMANENT FILE ADDRESS INQUIRY	H.MONS,2	M.FADD	7.8.8
44	MEMORY ADDRESS INQUIRY	H.MONS,3	M.ADRS	8.3.1
45	CREATE TIMER ENTRY	H.MONS,4	M.SETT	8.2.33
46	TEST TIMER ENTRY	H.MONS,5	M.TSTT	8.2.43
47	DELETE TIMER ENTRY	H.MONS,6	M.DLTT	8.2.14
48	SET USER STATUS WORD	H.MONS,7	M.SETS	8.2.32
49	TEST USER STATUS WORD	H.MONS,8	M.TSTS	8.2.42
4A	CHANGE PRIORITY LEVEL (PRIV)	H.MONS,9	M.PRIL	8.2.29
4B	CONNECT TASK TO INTERRUPT	H.MONS,10	M.CONN	8.2.8
4C	TASK OPTION WORD INQUIRY	H.MONS,24	M.PGOW	8.2.28
4D	ARITHMETIC EXCEPTION INQUIRY	H.MONS,23	M.TSTE	8.2.41
4E	TIME-OF-DAY INQUIRY	H.MONS,11	M.TDAY	8.2.40
4F	MEMORY DUMP REQUEST	H.MONS,12	M.DUMP	8.3.2
50	LOAD OVERLAY	H.MONS,13	M.OLAY	8.2.27
51	LOAD AND EXECUTE OVERLAY	H.MONS,14	M.OLAY	8.2.27
52	ACTIVATE TASK	H.MONS,15	M.ACTV	8.2.1
53	RESUME TASK EXECUTION	H.MONS,16	M.SUME	8.2.37

USER LEVEL SYSTEM SERVICES - SVC ORDER

(CONTINUED)

SVC 1	DESCRIPTION	MODULE,EP	MACRO	REF MANUAL SECTION
54	SUSPEND TASK EXECUTION	H.MONS,17	M.SUSP	8.2.38
55	TERMINATE TASK EXECUTION	H.MONS,18	M.EXIT	8.2.20
56	ABORT SPECIFIED TASK	H.MONS,19	M.BORT	8.2.4
57	ABORT SELF	H.MONS,20	M.BORT	8.2.4
58	PROGRAM HOLD REQUEST	H.MONS,25	M.HOLD	8.2.23
59	SET TABS IN UDT	H.TSM,5	N/A	N/A
5A	DELETE TASK	H.MONS,31	M.DELTSK	8.2.11
5B	SCAN TERMINAL INPUT BUFFER	H.TSM,2	M.TSCAN	5.6.1
5C	TRAP ONLINE USER'S TASK	H.TSM,6	M.TBRKON	5.6.2
5D	DISCONNECT TASK FROM INTERRUPT	H.MONS,38	M.DISCON	8.2.13
5E	EXIT FROM MESSAGE RECEIVER	H.MONS,39	M.XMSGR	8.2.47
5F	PARAMETER TASK ACTIVATION (PRIV)	H.MONS,40	M.PTSK	8.2.30
60	SET USER ABORT RECEIVER ADDRESS	H.MONS,26	M.SUAR	8.2.36
61	SUBMIT JOB FROM DISC FILE	H.MONS,27	M.CDJS	8.2.7
62	ABORT WITH EXTENDED MESSAGE	H.MONS,28	M.BORT	8.2.4

USER LEVEL SYSTEM SERVICES - SVC ORDER

(CONTINUED)

SVC 1	DESCRIPTION	MODULE,EP	MACRO	REF MANUAL SECTION
63	LOAD AND EXECUTE INTERACTIVE DEBUGGER	H.MONS,29	M.DEBUG	8.2.10
64	GET TASK NUMBER	H.MONS,32	M.ID M.YID	8.2.24 8.2.26
65	GET ADDRESS LIMITS	H.MONS,41	M.GADRL	8.3.7
66	DEBUG LINK SERVICE	H.MONS,42	N/A	8.2.51
67	GET DYNAMIC TASK EXECUTION SPACE	H.ALOC,10	M.GE	8.3.9
68	FREE DYNAMIC TASK EXECUTION SPACE	H.ALOC,11	M.FE	8.3.6
69	GET DYNAMIC EXTENDED INDEXED DATA SPACE	H.ALOC,8	M.GD	8.3.8
6A	FREE DYNAMIC EXTENDED INDEX DATA SPACE	H.ALOC,9	M.FD	8.3.5
6B	RECEIVE MESSAGE LINK ADDRESS	H.MONS,43	M.RCVR	8.2.31
6C	SEND MESSAGE TO SPECIFIED TASK	H.MONS,44	M.SMSGR	8.2.34
6D	SEND RUN REQUEST	H.MONS,45	M.SRUNR	8.2.35
6E	BREAK/TASK INTERRUPT LINK	H.MONS,46	M.BRK	8.2.5
6F	ACTIVATE TASK INTERRUPT	H.MONS,47	M.INT	8.2.25
70	EXIT FROM TASK INTERRUPT LEVEL	H.MONS,48	M.BRKXIT M.XBRKR	8.2.6 8.2.45
71	SHARE MEMORY WITH ANOTHER TASK	H.ALOC,12	M.SHARE	8.3.11
72	GET SHARED MEMORY	H.ALOC,13	M.INCL	8.3.10
73	PERMANENT FILE LOG	H.MONS,33	M.LOG	7.8.15

USER LEVEL SYSTEM SERVICES - SVC ORDER

(CONTINUED)

SVC 1	DESCRIPTION	MODULE,EP	MACRO	REF MANUAL SECTION
74	USERNAME SPECIFICATION	H.MONS,34	M.USER	7.8.28
75	CREATE PERM FILE	H.FISE,12	M.CREATE	7.8.4
76	CHANGE TEMP FILE TO PERMANENT	H.FISE,13	M.PERM	7.8.17
77	DELETE PERM FILE	H.FISE,14	M.DELETE	7.8.7
78	RESERVED FOR FUTURE USE			
79	FREE SHARED MEMORY	H.ALOC,14	M.EXCL	8.3.3
7A	GET MSG PARAMETERS	H.MONS,35	M.GMSGP	8.2.21
7B	GET RUN PARAMETERS	H.MONS,36	M.GRUNP	8.2.22
7C	WAIT FOR ANY MSG, END ACTION, OR BRK	H.MONS,37	M.ANYW	8.2.2
7D	EXIT RUN RECEIVER	H.MONS,49	M.XRUNR	8.2.49
7E	EXIT FROM MESSAGE END ACTION ROUTINE	H.MONS,50	M.XMEA	8.2.46
7F	EXIT FROM RUN REQUEST END ACTION ROUTINE	H.MONS,51	M.XREA	8.2.28

APPENDIX C

MPX-32 ABORT AND CRASH CODES

Address Specification Trap Handler (H.IP0C)

CODE	DESCRIPTION
AD01	Address specification occurred within the operating system.
AD02	Address specification occurred within the current task.
AD03	Trap occurred while no tasks were in active state.
AD04	Trap occurred within another interrupt trap routine.

CODE	DESCRIPTION
AL01-AL06	Reserved
AL07	The combined number of file assignments for a task exceeds number specified. The cataloged assignments are combined with those defined by $ASSIGN statements. See cataloger FILES directive and recatalog if needed.
AL08	An assigned permanent file is nonexistent.
AL09	An assigned device is not configured in the system. An assigned device is off-line.
AL10	Reserved
AL11	Reserved
AL12	Unable to load program because of I/O error or addressing inconsistencies in load module preamble.
AL13	An unrecoverable I/O error has occurred during the read of the task preamble into the TSA.
AL14	Reserved
AL15	An assigned device type is not configured in the system.
AL16	A resident request has been issued for a task requiring an SLO, SBO, SGO or SYC file. Resident tasks cannot use system files.
AL17	Reserved
AL18	Reserved
AL19	A file code to file code assignment (ASSIGN4) has been made to an undefined file code. A file code must be defined before a second file code can be equated by an ASSIGN4.
AL20	User attempted deallocation of TSA.
AL21	Destroyed task MIDL was detected while attempting to allocate dynamic execution space.
AL22	A software checksum error has occurred during task loading.
AL23	An invalid user name is cataloged with the task. The user name is either not contained in the user name file M.KEY or a correct user key is not present. Also: task has attempted to deallocate TSA.

AL24	Access to an assigned permanent file is by password only, and a valid password was not included on the cataloged assignment or Job Control statement assignment.
AL25	Undefined Resource Requirement Summary (RRS) type (internal format of an assignment statement is wrong).
AL26	The task has requested more blocking buffers than were specified during catalog. See Cataloger BUFFER directive and recatalog if needed.
AL27	There are no free entries in shared memory table for GLOBAL, DATAPOOL, CSECT, or other shared areas.
AL28	Task is attempting to share an undefined GLOBAL or DATAPOOL memory partition.
AL29	Task is attempting to exclude undefined memory partition.
AL30	The requested device is already assigned to the requesting task via another file code. Use ASSIGN4 or deallocate before reallocating.
AL31	Logical file code has already been allocated by caller (e.g., a card reader may already be assigned to lfc IN and a magnetic tape cannot be assigned to the same file code). Use ASSIGN4 or deallocate before reallocating.
AL32	Dynamic common block may not be assigned via ASSIGN1 directive.
AL33	Shared memory definition conflicts with caller's address space.
AL34	Shared memory partition not defined in SMD.
AL35	Attempt to share an SMD entry that is not a memory partition.
AL36	Invalid password specified for shared memory partition.
AL37	Attempt to exclude undefined shared memory partition.
AL38	Attempt to activate a privileged task by unauthorized owner.
AL39	Shared memory entry not found.
AL40	Partition definition not found on SMD.
AL41	SMD definition not a dynamic definition.
AL42	Invalid password for this partition.
AL43	Task has attempted to allocate an unshared resource that was not available during task activation in a memory-only environment.

AL44	Unable to resume 'SYSBUILD' task during initial task activation in a memory-only environment.
AL45	Unable to deallocate input device after dynamic task activation in a memory-only environment.
AL46	Task has attempted to share memory via a dynamic memory partition in a memory-only environment.
AL47	Dynamic memory partitions cannot be greater than 1 megabyte.
AL48	The user has attempted to exclude a shared partition whose associated map blocks are not designated as being shared in the task's TSA.
AL49	The task's DSECT space requirements overlap the task's TSA space requirements.
AL50	The task's DSECT space requirements overlap the task's CSECT space requirements, or if no CSECT, load module is too large to fit in user's address space.
AL51-AL54	Reserved
AL55	The sum of the CSECT, DSECT, and the operating system sizes is greater than the total amount of memory configured.
AL56	Unrecoverable I/O error to the SMD.
AL57	File Lock Table (FLT) is full.

CODE	DESCRIPTION
AS01	Physical end-of-file encountered on write to the General Object (GO) file.
AS02	Physical end-of-file encountered on write to the Binary Output (BO) file.
AS03	Physical end-of-file encountered on write to the Listed Output (LO) file.
AS04	Physical end-of-file encountered on write to the scratch (UT1) file (i.e., $ASSIGN3 UT1 = DC, ????).
AS05	Physical end-of-file encountered on write to the cross-reference (UT2) file (i.e., $ASSIGN3 UT2 = DC,????).
AS06	There does not exist a prime number of three-word entries in the allocated core for the symbol table.
AS07	Unrecoverable I/O error on the Binary Output (BO) file.
AS08	Unrecoverable I/O error on the General Object (GO) file.
AS09	Unrecoverable I/O error on the Listed Output (LO) file.
AS10	Unrecoverable I/O error on the Source Input (SI) file.
AS11	Unrecoverable I/O error on the intermediate compressed source (UT1) file.
AS12	Physical end-of-file encountered on write to the Compressed Source Output (CS) file.
AS13	Checksum error on compressed source input either during pass 1 while reading compressed source from the Source Input (SI) file or during pass 2 while reading the intermediate scratch compressed source (UT1) file.
AS14	The file the Assembler is using as the macro library was not successfully created by the macro library generator. The file is invalid.
AS15	Unrecoverable I/O error on the Macro Library (MAC) file.
AS16	Unrecoverable I/O error on the cross-reference (UT2) file.
AS17	Unrecoverable I/O error on the Compressed Source Output (CS) file.

AS18	Invalid blocking buffer control pointer encountered on the Binary Output (BO) file.
AS19	Invalid blocking buffer control pointer encountered on the General Object (GO) file.
AS20	Invalid blocking buffer control pointer encountered on the Listed Output (LO) file.
AS21	Invalid blocking buffer control pointer encountered on the Source Input (SI) file.
AS22	Invalid blocking buffer control pointer encountered on the Scratch Compressed Source (UT1) file.
AS23	Invalid blocking buffer control pointer encountered on the Compressed Source Output (CS) file.
AS24	Invalid blocking buffer control pointer encountered on the cross-reference (UT2) file.
AS25	The macro library (MAC) file is unblocked.
AS26	End of file on MA2 file.
AS27	Unrecoverable I/O error on MA2 file.
AS28	Invalid blocking buffer control pointer on MA2 file.
AS29	MAC assigned to illegal device.
AS30	MA2 assigned to illegal device.
AS31	Potential abort conditions have been detected during program assembly. An abort status flag within the job's TSA has been set during assembler termination processing. Conditional job control directives may be used to test status prior to job continuation.
AS32	Unrecoverable I/O error on the prefix (MPXPRE) file.
AS33	Invalid blocking buffer control pointer encountered on the prefix (MPXPRE) file.

Auto-Start Trap Processor (H.IPAS)

CODE	DESCRIPTION
AU01	Trap occurred on auto-start.
AU02	Trap occurred in another interrupt trap routine.
AU03	Reserved
AU04	Reserved
AU05	User was unmapped when trap occurred.

Debugger

CODE	DESCRIPTION
DB01	End of medium error on lfc #OT in batch mode.
DB02	Fatal I/O error on lfc contained in the abort message.

Call Monitor Interrupt Processor (H.IP27 and H.IP0A)

CODE	DESCRIPTION
CM01	Physical end-of-file encountered on write to the compressed source output (CS) file. Call monitor interrupt processor cannot locate the CALM instruction.
CM02	Expected CALM instruction does not have CALM (X'30') opcode.
CMO3	Invalid CALM number.
CM04	CALM number too low (out of bounds).
CM05	CALM number too big (out of bounds).

CODE	DESCRIPTION
CT01	Physical end-of-file encountered on subroutine library. The lfc of the library in question is displayed. This results from the library being updated by another user while it is allocated by the Cataloger.
CT02	Load module file specified with CATALOG cannot be allocated.
CT03	Unrecoverable I/O error encountered on the DATAPOOL dictionary file assigned to DPD.
CT04	Listed output space is deleted and additional SLO space cannot be allocated.
CT05	Unrecoverable I/O error on file or device assigned to SBM for SYMTAB output.
CT06	An error occurred during the cataloging process and the reason is described in the SLO.

CODE	DESCRIPTION
DP01	Unrecoverable I/O error while reading or writing the DATAPOOL dictionary.
DP02	Dictionary file code (DPD) unassigned.
DP03	Unrecoverable error on error audit trail (ER) file.
DP04	Unrecoverable error on audit trail (LO) file.
DP05	Unable to allocate additional SLO space for audit trail after initial file is filled.
DP06	Unable to allocate additional SLO space for error audit trail after initial file is filled.
DP07	Invalid directive.
DP08	The name 'DATAPOOL' is not defined as a core partition.
DP09	Dictionary overflow.
DP10	Unable to reassign the DPD file.
DP11	End-of-tape or illegal end-of-file encountered on IN.
DP12	Physical end-of-media encountered on OT.
DP13	Unrecoverable error on IN.
DP14	Unrecoverable error on OT.
DP15	File code OT unassigned and the SAVE function requested.
DP16	File code IN unassigned and the REMAP function requested.
DP17	Sequence error on dictionary entry record (accessed through file code IN).
DP18	Checksum error on dictionary entry record (accessed through file code OT).
DP19	Invalid specification on REMAP directive.
DP20	Invalid specification on DPD directive.
DP21	Unrecoverable error on directive input (SYC) file.
DP22	Dictionary size is less than the required minimum (five records).
DP99	A non-fatal error occurred.

Error Condition Codes for DPEDIT

CODE	DESCRIPTION
EC11	Attempt to delete a symbol not found in the dictionary.
EC12	Attempt to delete a symbol that is used as a base for another variable.
EC13	A change is requested for a symbol used as a base that may result in a change in the relative address.
EC14	The calculated relative address does not fall on the specified boundary (precision).
EC15	The referenced base symbol is not in the datapool dictionary.
EC16	Attempt to add a symbol that is already defined in the dictionary.
EC17	The calculated relative address is not within the range of the datapool core partition.
EC18	The datapool variable does not reside in the dictionary at the location computed by the hash coding scheme.
EC19	Invalid specification on directive.
EC20	Log function deleted, not enough memory to sort data.
EC21	Log function deleted, the scratch sort file is not enough to contain the necessary data.
EC22	Log function deleted, unrecoverable I/O error on the scratch sort file.
EC23	Attempted to change a symbol not found in the dictionary.
EC24	Computed relative address does not agree with actual.
EC25	Entries are multiply defined.
ERnn	Error encountered in processing data card fields. The column number in which the error was detected is specified by "nn".

CODE	DESCRIPTION
ED01	User terminal I/O hardware error.
ED02	Internal line linkage invalid.
ED03	Reserved (RTM Only).
ED04	Internal logic error.

CODE	DESCRIPTION
FM01	Invalid command verb on directive.
FM02	Required argument(s) absent from a directive.
FM03	Create requested for an existing file.
FM04	Device specified is invalid for this command.
FM05	Decimal number specification contains non-decimal digits.
FM06	Hexadecimal number specification contains non-hexadecimal characters.
FM07	Specified file is password protected and correct password not specified.
FM08	Attempt to expand a core partition.
FM09	Cannot create or expand file due to unavailability of disc space for allocation.
FM10	Attempt to create a fast file which mapped into an existing fast file in the SMD.
FM11	DELETE, SAVE, or EXPAND requested for a disc file which does not exist.
FM12	Insufficient file assignment table space for I/O to the SMD.
FM13	Unrecoverable I/O error to the SMD.
FM14	Unrecoverable I/O error to the SYC file.
FM15	Unrecoverable I/O error to the SLO file.
FM16	Unrecoverable I/O error to the IN file.
FM17	Unrecoverable I/O error to the OUT file.
FM18	Invalid argument.
FM19	Cannot allocate required file.
FM20	Unrecoverable I/O error on SAVE/RESTORE file.
FM21	Unexpected EOF on IN file.
FM24	File specified for RESTORE not found.
FM25	Too many prototypes specified.

FM26	EOF expected on IN file not found.
FM41	End of medium on lfc SLO.
FM42	Invalid username or key.
FM99	Directive errors have been detected during execution of the File Manager. An abort status flag within the job's TSA has been set during File Manager termination processing. Conditional job control directives may be used to test status prior to job continuation.

File System

CODE	DESCRIPTION
FS01	Unrecoverable I/O error to the System Master Directory (SMD).
FS02	Unrecoverable I/O error to a disc allocation map.
FS03	Attempt to add a new file, but the System Master Directory (SMD) is full.
FS04	A disc allocation map checksum error was detected.
FS05	Attempt to allocate disc space that is already allocated.
FS06	Attempt to deallocate disc space that is not allocated.
FS07	Reserved.
FS08	Unrecoverable I/O error occurred while zeroing a file during creation.

Fortran

CODE	DESCRIPTION
FT01	Fortran scratch file *U1 must be expanded (i.e., $ASSIGN3 *U1=DC,????).
FT02	Fortran scratch file *U2 must be expanded (i.e., $ASSIGN3 *U2=DC,????).
FT03	Binary output (BO) file must be expanded if a disc file (i.e., $ASSIGN2 BO=SLO,???? or a direct assignment to the card punch - $ASSIGN3 BO=CP).
FT04	The compiled program caused the SGO file to overflow. The size of the SGO file can be increased via SYSGEN or may be assigned to tape via Operator Communications.
FT05	End of medium on nonspooled SLO file. File must be expanded if a disc file or a direct assignment made to the line printer - $ASSIGN3 LO=LP.
FT06	Potential abort conditions have been detected during program compilation. An abort status flag within the job's TSA has been set during compiler termination processing. Conditional job control directives may be used to test status prior to job continuation.

CODE	DESCRIPTION
HT01	Trap occurred in user's map area.
HT02	Trap occurred in another interrupt trap routine.
HT03	Trap occurred while no other tasks were in the active state.
HT04	Reserved.
HT05	User was unmapped when trap occurred.

CODE	DESCRIPTION
IO01	An I/O operation has been attempted for which the FCB is not properly linked to a File Assignment Table (FAT) entry. Since this linkage is established by IOCS when the file is opened, either the user task has not properly opened the file or the FCB has been inadvertently destroyed subsequent to the time the open file operation was performed.
IO02	An I/O operation has been attempted on an unopened file. This abort code will normally be issued when a user has opened a file, subsequently closed the file, then attempted an I/O operation on the file.
IO03	An unprivileged task is attempting to read data into an area of core which is not allocated for its use. This type of abort is usually caused by an invalid TCW in the task's FCB.
IO04	The control specifications in the FCB specify random access. However, the random access address contained in the FCB does not fall within the limits of the file.
IO06	Invalid blocking buffer control cells have been encountered during a read operation performed on a blocked file. This type error is normally caused in one of the three ways: (1) The user's blocking buffer has been inadvertently destroyed. (2) The file being read is not a properly blocked file. (3) A data transfer error has occurred on input of data from the file.
IO07	The task has attempted to perform an operation which is not valid for the device to which the user's file is assigned (e.g., a read operation specified for a file assigned to the line printer).
IO08	Reserved
IO09	The task has attempted to perform a rewind operation on the system SYC file.
IO10	The task has attempted a write End-of-File operation on a file which has been opened in the Read-Only access mode.
IO11	The task has attempted a write End-of-File operation on the system SYC file.

IO12	The task has attempted an erase or punch trailer operation on a file which has been opened in the Read-Only access mode.
IO13	The task has requested an illegal operation to be performed on a system file (backspace file, upspace, erase or punch trailer, eject, advance record, advance file, or backspace record).
IO14	A task running in the unprivileged mode has attempted to reserve an I/O channel.
IO15	A task has requested a type operation and the Type Control Parameter Block (TCPB) specified indicates that an operation associated with that TCPB is already in progress.
IO17	The task has attempted an open operation on a file, and no File Pointer Table (FPT) entry exists with a matching file code. This type of abort is most often caused by an improper or missing file assignment directive at catalog or linking load time. This type abort may also occur if the logical file code portion of the task's FCB has been inadvertently destroyed.
IO18	Reserved
IO21	IOCS has encountered an unrecoverable I/O error in attempting to process an I/O request on behalf of a task.
IO22	An illegal IOCS entry point has been entered by a task.
IO24	A task has specified an illegal address or transfer count in the FCB TCW. This type of error is usually the result of trying to output to a halfword device from a data area which is not on a halfword boundary. This error may also occur if the task attempts to transfer other than an even multiple of halfwords to or from a halfword device.
IO25	The task has requested a data transfer operation (read or write) with a Transfer Control Word (TCW) which specifies a quantity of zero.
IO26	Illegal sequence of operations while in read mode on either a system file or a blocked file.
IO27	Illegal sequence of operations while in write mode on either a system file or a blocked file.
IO28	Attempt to advance a record while in the write mode on a blocked file.
IO29	Attempt to advance a file while in the write mode on a blocked file.
IO30	Illegal or unexpectecd volume number encountered on magnetic tape.

IO32	Calling task has attempted to perform a second read on a $ statement through the SYC file.
IO33	An Invalid Device Address has been specified in the Task's Input/Output Control Header (IOCH).
IO34	An unprivileged task has requested the link service.
IO35	An unprivileged task has specified an IOCB list greater than 30 IOCB's in length.
IO36	A SYSGEN error has occurred, and the handler HAT address is not in the Controller Definition Table (CDT).
IO37	Job sequence number not found in the job table for task attempting to open SYC or SGO file.
IO38	The task has requested a write operation to be performed on a file which has been opened in the read-only access mode. Permanent files to which a task has read but not write access are opened read-only even though read-write is specified when the file is opened.
IO39	Blocked file indicated in FCB (or implied via assignment to a system file) but no blocking buffers available.
IO40	User TCW is in error due to one or more of the following conditions: 1. Unable to construct a valid TCW because the transfer count is too large. 2. Transfer count not an even multiple of transfer type. 3. Data address not bounded for transfer type (types = W, HW, B).
IO43	Input/Output Control List (IOCL) or data address not in contiguous 'E' memory (ASYNC,BSYNC).
IO46	Dynamic storage space for IOCDs within IOQ exhausted.
IO47	Class 'E' device TCW is not in class 'E' memory. This type of error indicates a map failure.
IO48	Reserved
IO49	Device access failure on OPEN.
IO53	The user has attempted to write to SYC file in Batch mode.
IO54	An attempt has been made to use the same logical file code in two or more File Control Blocks.

CODE	DESCRIPTION
JC01*	Unrecoverable read error from job's SYC file.
JC02*	Unrecoverable write error on SLO file.
JC03*	Unrecoverable write error on job's SGO file.
JC04*	Unable to build FAT/FPT for SLO or for SBO link file which indicates a program error.
JC05*	Unable to allocate disc space for SLO file.
JC06*	An entry is not available in the System Output Directory (M.SOD) for the definition of the job's SLO or SBO link file.
JC07	This job's Job Table entry has been destroyed which indicates a program error.
JC08*	Unable to allocate the job's SYC file.
JC09	Unrecoverable I/O error to SMD returned on call to File System Executive (H.FISE,10).
JC10*	Unrecoverable I/O error to disc allocation map returned by File System Executive (H.FISE,3 or H.FISE,4).
JC11*	Unable to allocate job's SGO file.

* Whenever a Job Control task aborts with one of these codes, the associated job is deleted.

CODE	DESCRIPTION
LD01	Load code section error.
LD02	Code section checksum error.
LD03	Bias code error.
LD04	Code matrix checksum error.
LD05	Load data section error.
LD06	Data section checksum error.
LD07	Bias data error.
LD08	Data matrix checksum error.

CODE	DESCRIPTION
LE01	Directive error. The last directive printed on the directive list is in error.
LE02	Object record sequence error. The name of the module with the error will be the last line printed on the log.
LE03	Object record checksum error. The name of the module with the error will be printed as the last line of output on the log.
LE04	Object module format error. The module whose name is the module following that one has an invalid record code in record 1.
LE05	Incomplete object module. An object module, whose name is the last printed line of the log, has no terminal record (Hex DF).
LE06	Unrecoverable error on the SYC file.
LE07	Unrecoverable error on the LGO file.
LE08	Unrecoverable error on the LLO file.
LE09	Unrecoverable error on the LIB file.
LE10	Unrecoverable error on the DIR file.
LE11	Unrecoverable error on either the dynamically assigned temporary library file or temporary directory file.
LE12	Allocation denial on request for temporary disc space used to perform update function (i.e., disc space unavailable).
LE13	Delete table overflow. A combined maximum of 255 modules may be deleted/replaced/added in any one LIBED run.
LE14	End-of-medium encountered on temporary library. This error indicates the need to expand the subroutine library assigned to the LIB file code. The FILEMGR may be used to perform this function prior to another LIBED run.
LE15	End-of-medium encountered on temporary directory. This error indicates the need to expand the subroutine library directory assigned to the DIR file code. The FILEMGR may be used to perform this function prior to another LIBED run.
LE16	End-of-medium encountered on LIB file. The assigned file must be reallocated.
LE17	End-of-medium encountered on DIR file. The assigned file must be reallocated.

LE18	End-of-medium encountered on either LIB or DIR file. This error may occur on either a log, statistics, or update run and indicates that a previous CREATE run terminated prior to completion with an uncorrectable I/O error or a LE16/LE17 error.

MEDIA

CODE	DESCRIPTION
MD01	Potential abort conditions have been detected during media conversion operation. An abort status flag within the job's TSA has been set during compilation or execution processing. Conditional job control directives may be used to test status prior to job continuation. See output on logical file Code *OT for details about the abort condition.
MD02	At EOF on a SLO file.

MACLIBR

CODE	DESCRIPTION
ME99	Potential abort conditions have been detected during library editing operation. An abort status flag within the job's TSA has been set during editing processing. Conditional job control directives may be used to test status prior to job continuation.

Memory Parity Trap (H.IP02)

CODE	DESCRIPTION
MP01	Memory error occurred in a task's logical address space.
MP02	Memory error occurred in another interrupt trap routine (nested traps, context lost).
MP03	Memory error occurred while no tasks were in the active state.
MP04	Memory error occurred in a map block reserved for the O/S.
MP05	Error occurred while current task was in the unmapped mode.

CODE	DESCRIPTION
MS01	Permanent file address inquiry service found a number of allocation units in the Unit Definition Table that do not correspond to any known disc.
MS02	Invalid function code specified for request to create a timer entry. Valid codes are ACP (1), RSP or RST (2), STB (3), RSB (4), and RQI (5).
MS03	A privileged task bit Set/Reset address is outside of the operating system or a static memory partition, or an unprivileged task bit Set/Reset address is outside of a static memory partition.
MS04	Task has attempted to create a timer entry to request an interrupt with a priority level outside the range of X'12' to X'7F', inclusive, or the requesting task is unprivileged.
MS05	Invalid function code has been specified for request to set user status word.
MS06	Unprivileged task has attempted to reset a task priority level or a privileged task has attempted to reset a task priority to a level outside the range of 1 to 64, inclusively.
MS07	Cannot load overlay segment due to software checksum or data error.
MS08	Overlay is not in the SMD.
MS09	Task has attempted to connect a task to an interrupt level not defined for indirectly connected tasks.
MS10	Overlay has an invalid preamble.
MS11	An unrecoverable I/O error has occurred during overlay loading.
MS12	Overlay is password protected.
MS16	Task has requested dynamic allocation with an invalid function code.
MS17	File name contains characters outside range of X'20' to X'5F', inclusively.
MS21	Multi-volume magnetic tape allocation request made to scratch (SCRA) tape.
MS22	Multi-volume magnetic tape allocation request made on shared tape drive.

MS23	Task has issued a MOUNT MESSAGE ONLY allocation request to a non-allocated drive or to a device which is not a magnetic tape.
MS24	Task has specified an illegal volume number (0 if tape is multivolume; non-zero if tape is single volume).
MS25	Operator has aborted task in response to mount message.
MS28	A permanent file log has been requested, but the address specified for storage of the directory entry is not contained within the calling task's logical address space.
MS29	Task has attempted to load the interactive Task Debugger overlay in a memory-only environment.
MS30	Task has attempted to obtain a permanent file log in a memory-only environment.
MS31	User attempted to go to an any wait state from an end action routine.
MS32	Invalid register set-up detected in M.ID.
MS89	An unprivileged task has attempted to reestablish an abort receiver (other than M.IOEX).
MS90	Task has made a run request end action routine exit while the run request interrupt was not active.
MS91	Task has attempted normal exit with a task interrupt still active.
MS92	Task has attempted to queue a message during its exit sequence.
MS93	An invalid Receiver Exit Block (RXB) address was encountered during message exit.
MS94	An invalid Receiver Exit Block (RXB) return buffer address was encountered during message exit.
MS95	Task has made a message exit while the message interrupt was not active.
MS96	An invalid Receiver Exit Block (RXB) address was encountered during run receiver exit.
MS97	An invalid Receiver Exit Block (RXB) return buffer address was encountered during run receiver exit.
MS98	Task has made a run receiver exit while the run receiver interrupt was not active.
MS99	Task has made a message end action routine exit while the message interrupt was not active.

CODE	DESCRIPTION
RS47	Invalid time interval request.
RS48	Invalid activation request.
RS49	Invalid run request.
RS53	Invalid task number.
RS60	Invalid address specified.
RS65	Invalid delete request.
RS66	Invalid abort request.
RS67	Invalid resource mark request.
RS68	Invalid disconnect request.
RS69	Skip file or record operation requested on non-existent FCB.
RS70	Allocation error (appears only if IOSTAT and $n parameters have been omitted).
RT01	Unformatted read I/O error.
RT02	Formatted read I/O error.
RT03	Unformatted write I/O error.
RT04	Formatted write I/O error.
RT05	Reference made to non-existent device type or address.
RT06	Unit out of 0-999 range.
RT07	No left parenthesis on format.
RT08	Transfer index out of range (option 7 or M:ERRFLG can be used to avoid an abort).
RT09	Format error.
RT10	The I/O transfer requirements for the data buffer are incompatible with the amount of available data.
RT11	Format parenthesis level in excess of two.
RT13	Argument list exceeds logical read record.

RT14	Incorrect descriptor in format.
RT15	Integer descriptor but non-integer argument (option 7 or M:ERRFLG can be used to avoid an abort).
RT16	Hexadecimal descriptor but non-hexadecimal argument (option 7 or M:ERRFLG can be used to avoid an abort).
RT17	D, E, F, G descriptor, not real or complex argument (option 7 or M:ERRFLG can be used to avoid an abort).
RT18	Logical descriptor but non-logical argument (option 7 or M:ERRFLG can be used to avoid an abort).
RT19	Attempt to read past EOF/EOM.
RT20	Attempt to write past EOF/EOM.
RT21	Attempt to read past EOF/EOM.
RT22	Attempt to write past EOF/EOM.
RT23	Attempt to backspace following EOF/EOM.
RT24	Rewind after EOF/EOM.
RT25	Formatted record read.
RT26	Unformatted record read.
RT27	Doubleword integer overflow (option 7 or M:ERRFLG can be used to avoid an abort).
RT28	Byte integer input with negative sign (option 7 or M:ERRFLG can be used to avoid an abort).
RT29	Byte integer overflow (option 7 or M:ERRFLG can be used to avoid an abort).
RT30	Halfword integer overflow (option 7 or M:ERRFLG can be used to avoid an abort).
RT31	Full word integer overflow (option 7 or M:ERRFLG can be used to avoid an abort).
RT32	Illegal character in D, E, F, G input (option 7 or M:ERRFLG can be used to avoid an abort).
RT33	Underflow in floating conversion (option 7 or M:ERRFLG can be used to avoid an abort).
RT34	Overflow in floating conversion (option 7 or M:ERRFLG can be used to avoid an abort).

RT35	Argument list overflow (option 7 or M:ERRFLG can be used to avoid an abort).
RT36	Argument list overflow (option 7 or M:ERRFLG can be used to avoid an abort).
RT40	Attempt to free busy IOCH/IOCB entry.
RT41	Attempt to link busy IOCH/IOCB entry.
RT42	IOCH/IOCB table overflow.
RT43	Wait I/O returned before I/O termination.
RT44	Status parameter not linked to ADI device prior to I/O request.
RT46	ADI table address not on halfword boundary.
RT50	Missing or omitted parameter.
RT51	Parameter out of range.
RT52	End of search list reached.
RT55	Error found in math library routine.
RT61	List-directed I/O (input) encountered, character string split between two records.
RT62	Internal file read/write past EOF/EOM with no END option specified.
RT63	Block number exceeds maximum block number in file.
RT64	Record overflow.
RT65	Record length exceeds maximum allowable.
RT66	Record length not specified for random access or specified for sequential file.
RT67	Implicit open not allowed for or random access I/O.
RT68	Reference to sequential operation on a file opened for direct access not allowed.
RT69	Error(s) encountered on open.
RT80	Subscript error (i.e., subscript not a decimal number, illegal punctuation, excessive subscripts, or subscript out of range).
RT81	NAMELIST identifier error (i.e., column 1 non-blank, ampersand character not present, name does not immediately follow ampersand character, or non-blank following name).

RT82	Symbolic name error (no equal sign after variable/array name).
RT83	Data item error (i.e., excessive values for symbol or expected to find symbol).
RT84	Illegal value (i.e., illegal punctuation, missing comma, zero Hollerith count, or illegal character in value).
RT85	Attempt to read past EOF/EOM.
RT86	Attempt to write past EOF/EOM.
RT87	Symbolic name not defined in NAMELIST statement.
RT88	Repeat count error.
RT89	Symbolic name exceeds eight characters.
RT90	Invalid read/write operation.
RT91	End-of-file status return pursuant to random access record.
RT92	Random access partition number out-of-range (i.e., partition number not between 1 and 95, inclusive).
RT93	Random access number out-of-range (i.e., record number not between 1 and 65,535, inclusive).
RT94	Random access transfer length (write/read) or record size definition (define) out-of-range (i.e., transfer record length not between 1 and 65,535 bytes, inclusive).
RT95	Invalid random access argument list length.
RT96	FCB table overflow (16 or more files for RTM; 31 or more files for MPX-32).
RT97	Diagnostic output message exceeds 100 lines. To allow more diagnostic messages, statically assign the DO file (i.e., $ASSIGN2 DO=SLO,500).
RT98	Denial return when attempting to allocate file for diagnostic output message.
RT99	Insufficient blocking buffer space (each unit assignment to a system file requires one blocking buffer unless one file is assigned to another, i.e., via $ASSIGN4).

System Binary Output

CODE	DESCRIPTION
SB01	An I/O error has been encountered on the device assigned as the system binary (punched) output device.
SB02	The system output program has encountered an unrecoverable I/O error in attempting to read a punched output file from disc.
SB03	Denial of file code to file code allocation for J.SOUT2 indicates loss of system integrity.
SB04	System binary output aborted by operator.
SB05	No timer entry for system binary output (system fault).
SB06	Five echo check errors detected while attempting to punch a single card.

System Check Trap Processor

CODE	DEFINITION
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).

CODE	DESCRIPTION
SG01	Invalid loader function code in binary object module from the System Resident Module (OBJ) file.
SG02	Invalid binary record read from System Resident Module (OBJ) file (byte 0 must be X'FF' or X'DF').
SG03	Sequence error in module being read from temporary file.
SG04	CHECKSUM error in module being read from temporary file.
SG05	Unable to find CDT and/or UDT for I/O module load.
SG06	Unable to obtain additional memory required for resident system image module loading.
SG07	Unable to obtain memory required for resident system image construction.
SG08	Non-relocatable byte string encountered in binary module being processed from temporary file.
SG09	Unable to allocate temporary file space.
SG10	Overrun of SYSGEN address space by system being generated. Probable erroneous size specification in PATCH or POOL directive.
SG11	Sequence error while reading object module from file assigned to 'OBJ'.
SG12	CHECKSUM error while reading object module from file assigned to 'OBJ'.
SG13	Unable to allocate disc space for SYMTAB file.
SG14	Unable to allocate disc space for SYSTEM IMAGE file.
SG15	Maximum number (240) of symbol table/patch file entries exceeded.
SG16	Missing SYSTEM or SYMTAB directive.
SG17	Invalid IPU interval timer priority. Must not be between X'78' and X'7F'.
SG18	Maximum size of 80K for target system has been exceeded.

SG19	Attempt to define interrupt vectoring routine as system reentrant. Only device handlers may be system reentrant.
SG20	Unable to find "link" device in UDT.
SG21	Insufficient room in memory pool for download file list.
SG23	Insufficient shared memory table entries specified with SHARE directive. Number of entries must be equal to or greater than the number of partitions specified with /PARTITION NAME directives.
SG24	Attempt to define partition starting mapblock number in operating system area.
SG25	Attempt to define partition starting mapblock number in non-configured physical memory.
SG26	Attempt to use a module incompatible with the target machine type. The offending module name is the last entry on the listing followed by three asterisks (***).
SG27	The device specified in either the SMD, SWP, SID, LOD or POD directives is not included in the configuration being built.
SG28	The null device specification which is required to be included in every configuration is missing.
SG99	Directive errors encountered.

CODE	DESCRIPTION
SM01	The System Input Directory (M.SID) which is created at SYSGEN, does not exist. The directory may be created with the File Manager, but the file must be zeroed during creation.
SM02	The Systems Output Directory (M.SOD) which is created at SYSGEN, does not exist. The directory may be created with the File Manager, but the file must be zeroed during creation.
SM03	Unable to build a FAT/FPT for a system output task which is attempting to allocate an SLO or SBO file. Indicates a program error.
SM04	The Job Table entry associated with a job for which end-of-job processing is being performed has been destroyed. Indicates a program error.
SM05	Entry linkage is not consistent on the System Output Directory (M.SOD). The contents of M.SOD have been destroyed or a program error exists.
SM06	Entry linkage has been destroyed on the System Input Directory (M.SID).
SM07	Entry linkage has been destroyed on the System Output Directory (M.SOD).
SM08	Unrecoverable I/O error on spooled link file.
SM09	Unrecoverable I/O error on System Input Directory (M.SID).
SM10	Unrecoverable I/O error on System Output Directory (M.SOD).
SM11	Unrecoverable I/O error to a disc allocation map returned on call to File System Executive (H.FISE,4).
SM12	Attempt to activate System Output task unsuccessful.
SM13	Unrecoverable I/O error to the SMD returned on call to File System Executive (H.FISE,1).
SM14	Attempt to access a system input or output file in a memory-only environment.

System Input Task (J.SSIN)

CODE	DESCRIPTION
SN01	Blocking buffer or FAT space is not available.
SN02	Unrecoverable I/O error from the disc file being used as the SYC file.
SN03	System Input Directory (M.SID) does not exist or an unrecoverable I/O error was encountered in attempting to access it.
SN04	Job Sequence Number has been duplicated. Indicates a program error.
SN05	Spooled Input Directory (M.SID) is full.
SN06	A permanent file specified on the OPCOM BATCH command does not exist.
SN07	Unrecoverable I/O error to the SMD returned on call to the File System Executive (H.FISE,1 or H.FISE,10).
SN08	Unrecoverable I/O error to the allocation map returned on call to the File System Executive (H.FISE.3 or H.FISE,4).

System Output Task (J.SOUT)

CODE	DESCRIPTION
ST01	Unrecoverable write error on destination device for SLO or SBO records.
ST02	Unable to perform file code to file code allocation for separator file code.
ST03	Unable to issue magnetic tape mount message via allocation service.

Whenever a System Output task aborts, the task may be restarted with the OPCOM/REPRINT or REPUNCH commands.

SVC Trap Processor (H.IP06)

CODE	DESCRIPTION
SV01	Abort of unprivileged task using M.CALL.
SV02	Invalid SVC number abort.
SV03	Abort of unprivileged task attempting use of a "privileged-only" service.
SV04	Invalid SVC type abort.
SV05	Abort of unprivileged task attempting M.RTRN.

Swap Scheduler Task (J.SWAPR)

CODE	DESCRIPTION
SW01	Unrecoverable I/O error.
SW02	Reserved
SW03	Reserved
SW04	No 'E' memory available for SWAPR's buffer file.
SW05	No FAT or FPT to allocate.
SW06	Task has requested inswap but was never outswapped.
SW07	EOM detected on swap file.

'SYSBUILD'

CODE	DESCRIPTION
SY01	Unable to allocate or open input device during initial task loading. (Memory only MPX-32)
SY02	Unable to activate task. (Memory only MPX-32)
SY03	Unable to deallocate or close input device after initial task loading.
SY04	IPL device is undefined.
SY05	File is too small for the tape contents.
SY06	Transfer count on read is zero.
SY07	Unable to create a permanent file.
SY08	Unable to allocate file.

CODE	DESCRIPTION
UD01	Potential abort conditions have been detected during update processing. An abort status flag within the job's TSA has been set during execution processing. Conditional job control directives may be used to test status prior to job continuation.
UD02	User requested abort from mount prompt.

Miscellaneous Abort Codes

CODE	DESCRIPTIONS
BT01	Block mode timeout trap.
EX01	An abort has occurred in the task exit sequence.
EX02	An abort has occurred during the task abort sequence and has been changed to a delete (kill) task request.
MC01	Machine check trap.
MF01	A map fault trap has occurred. This is the result of a bad memory reference outside of the user's addressable space.
MP01	Memory error occurred in a task's logical address space. This is an internal or CPU failure. Rerun task.
NM01	Indicates a CPU failure.
OC01	The operator has requested that the task be aborted.
PV01	Privilege violation trap.
TS01	User requested removal from a BREAK request.
TS02	User requested removal from a Wait State queue.
TS03	Task running from specified terminal was aborted when the terminal disconnected.
UI01	Undefined instruction trap.

When system crash occurs as a result of a trap handler entry, the CPU halts with the registers containing the following information:

R0=PSD Word 0 (when trap generated)
R1=PSD Word 1 (when trap generated)
R2=Real address of instruction causing trap
R3=Instruction causing trap
R4=CPU status word (from trap handler)
R5=Crash code:

MP01=X'4D503031'	(See H.IP02 Codes)
NM01=X'4E4D3031'	(Non-Present Memory - H.IP03)
UI01=X'55493031'	(Undefined Instruction - H.IP04)
PV01=X'50563031'	(Privilege Violation - H.IP05)
MC01=X'4D433031'	(Machine Check - H.IP07)
SC01=X'53433031'	(System Check - H.IP08)
MF01=X'4D463031'	(Map Fault - H.IP09)
CP01=X'42543031'	(Cache Parity - H.IP10) 32/87 only
BT01=X'42543031'	(Block Mode Timeout - H.IP0E)
HT01=X'48543031	(Privileged Halt Trap - H.IPHT) CONCEPT/32 only
SW01=X'53573031'	(See SWAPR codes)

R6=Real address of register save block
R7=C'TRAP'=X'54524150'

For further description, see Volume 1, Section 2.10.

APPENDIX D

NUMERICAL INFORMATION

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 884 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125

APPENDIX E

POWERS OF INTEGERS

TABLE OF POWERS OF SIXTEEN

16^n	n	16^{-n}
1	0	0.10000 00000 00000 00000 x 10
16	1	0.62500 00000 00000 00000 x 10^{-1}
256	2	0.39062 50000 00000 00000 x 10^{-2}
4 096	3	0.24414 06250 00000 00000 x 10^{-3}
65 536	4	0.15258 78906 25000 00000 x 10^{-4}
1 048 576	5	0.95367 43164 06250 00000 x 10^{-6}
16 777 216	6	0.59604 64477 53906 25000 x 10^{-7}
268 435 456	7	0.37252 90298 46191 40625 x 10^{-8}
4 294 967 296	8	0.23283 06436 53869 62891 x 10^{-9}
68 719 476 736	9	0.14551 91522 83668 51807 x 10^{-10}
1 099 511 627 776	10	0.90949 47017 72928 23792 x 10^{-12}
17 592 186 044 416	11	0.56843 41886 08080 14870 x 10^{-13}
281 474 976 710 656	12	0.35527 13678 80050 09294 x 10^{-14}
4 503 599 627 370 496	13	0.22204 46049 25031 30808 x 10^{-15}
72 057 594 037 927 936	14	0.13877 78780 78144 56755 x 10^{-16}
1 152 921 504 606 846 976	15	0.86736 17379 88403 54721 x 10^{-18}

TABLE OF POWERS OF TEN

10^n	n	10^{-n}
1	0	1.0000 0000 0000 0000
A	1	0.1999 9999 9999 999A
64	2	0.28F5 C28F 5C28 F5C3 x 16^{-1}
3E8	3	0.4189 374B C6A7 EF9E x 16^{-2}
2710	4	0.68DB 8BAC 710C B296 x 16^{-3}
1 86A0	5	0.A7C5 AC47 1847 8423 x 16^{-4}
F 4240	6	0.10C6 F7A0 B5ED 8D37 x 16^{-4}
98 9680	7	0.1AD7 F29A BCAF 4858 x 16^{-5}
5F5 E100	8	0.2AF3 1DC4 6118 73BF x 16^{-6}
3B9A CA00	9	0.4488 2FA0 9B5A 52CC x 16^{-7}
2 5408 E400	10	0.6DF3 7F67 5EF6 EADF x 16^{-8}
17 4876 E800	11	0.AFEB FF0B CB24 AAFF x 16^{-9}
E8 D4A5 1000	12	0.1197 9981 2DEA 1119 x 16^{-9}
918 4E72 A000	13	0.1C25 C268 4976 81C2 x 16^{-10}
5AF3 107A 4000	14	0.2D09 370D 4257 3604 x 16^{-11}
3 8D7E A4C6 8000	15	0.480E BE7B 9D58 566D x 16^{-12}
23 86F2 6FC1 0000	16	0.734A CA5F 6226 F0AE x 16^{-13}
163 4578 5D8A 0000	17	0.B877 AA32 36A4 B449 x 16^{-14}
DF0 B683 A764 0000	18	0.1272 5DD1 D243 ABA1 x 16^{-14}
8AC7 2304 89E8 0000	19	0.1D83 C94F B6D2 AC35 x 16^{-15}

APPENDIX F

ASCII INTERCHANGE CODE SET

Row	Col	0	1	2	3	4	5	6	7
Bit Positions 4 5 6 7	0 1 2 3	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
0000	0	NUL 12-0-9-8-1	DLE 12-11-9-8-1	SP No punch	0 0	@ 8-4	P 11-7	. 8-1	p 12-11-7
0001	1	SOH 12-9-1	DCI 11-9-1	! 12-8-7	1 1	A 12-1	Q 11-8	a 12-0-1	q 12-11-8
0010	2	STX 12-9-2	DC2 11-9-2	" 8-7	2 2	B 12-2	R 11-9	b 12-0-2	r 12-11-9
0011	3	ETX 12-9-3	DC3 11-9-3	# 8-3	3 3	C 12-3	S 0-2	c 12-0-3	s 11-0-2
0100	4	EOT 9-7	DC4 9-8-4	$ 11-8-3	4 4	D 12-4	T 0-3	d 12-0-4	t 11-0-3
0101	5	ENQ 0-9-8-5	NAK 9-8-5	% 0-8-4	5 5	E 12-5	U 0-4	e 12-0-5	u 11-0-4
0110	6	ACK 0-9-8-6	SYN 9-2	& 12	6 6	F 12-6	V 0-5	f 12-0-6	v 11-0-5
0111	7	BEL 0-9-8-7	ETB 0-9-6	' 8-5	7 7	G 12-7	W 0-6	g 12-0-7	w 11-0-6
1000	8	BS 11-9-6	CAN 11-9-8	(12-8-5	8 8	H 12-8	X 0-7	h 12-0-8	x 11-0-7
1001	9	HT 12-9-5	EM 11-9-8-1	) 11-8-5	9 9	I 12-9	Y 0-8	i 12-0-9	y 11-0-8
1010	A	LF 0-9-5	SUB 9-8-7	. 11-8-4	: 8-2	J 11-1	Z 0-9	j 12-11-1	z 11-0-9
1011	B	VT 12-9-8-3	ESC 0-9-7	+ 12-8-6	; 11-8-6	K 11-2	[12-8-2	k 12-11-2	{ 12-0
1100	C	FF 12-9-8-4	FS 11-9-8-4	, 0-8-3	< 12-8-4	L 11-3	\ 0-8-2	l 12-11-3	| 12-11
1101	D	CR 12-9-8-5	GS 11-9-8-5	- 11	= 8-6	M 11-4	] 11-8-2	m 12-11-4	} 11-0
1110	E	SO 12-9-8-6	RS 11-9-8-6	. 12-8-3	> 0-8-6	N 11-5	^ 11-8-7	n 12-11-5	~ 11-0-1
1111	F	SI 12-9-8-7	US 11-9-8-7	/ 0-1	? 0-8-7	O 11-6	_ 0-8-5	o 12-11-6	DEL 12-9-7

Some positions in the ASCII code chart may have a different graphic representation on various devices as:

ASCII	IBM 029
!	¦
[	¢
]	!
^	>

Control Characters:

NUL	–	Null	DC3	–	Device Control 3
SOH	–	Start of Heading (CC)	DC4	–	Device Control 4 (stop)
STX	–	Start of Text (CC)	NAK	–	Negative Acknowledge (CC)
ETX	–	End of Text (CC)	SYN	–	Synchronous Idle (CC)
EOT	–	End of Transmission (CC)	ETB	–	End of Transmission Block (CC)
ENQ	–	Enquiry (CC)	CAN	–	Cancel
ACK	–	Acknowledge (CC)	EM	–	End of Medium
BEL	–	Bell (audible or attention signal)	SS	–	Start of Special Sequence
BS	–	Backspace (FE)	ESC	–	Escape
HT	–	Horizontal Tabulation (punch card skip) (FE)	FS	–	File Separator (IS)
LF	–	Line Feed (FE)	GS	–	Group Separator (IS)
VT	–	Vertical Tabulation (FE)	RS	–	Record Separator (IS)
FF	–	Form Feed (FE)	US	–	Unit Separator (IS)
CR	–	Carriage Return (FE)	DEL	–	Delete
SO	–	Shift Out	SP	–	Space (normally nonprinting)
SI	–	Shift In	(CC)	–	Communication Control
DLE	–	Data Link Escape (CC)	(FE)	–	Format Effector
DC1	–	Device Control 1	(IS)	–	Information Separator
DC2	–	Device Control 2			

APPENDIX G

IOP PANEL COMMANDS

AS (CR)	Clear address stop
AS=xxxxxxxx (CR)	Set address stop
CRMD=xxxxxxxxxxxx (CR)	Load CRAM with xxxxxxxxxxxx
=xxxxxxxxxxxx (CR)	Load CRAM with data and increment address
CS (CR)	Read control switches
CS=xxxxxxxx (CR)	Set control switches
EA (CR)	Read effective address
EXEC (CR)	Execute CRAM
GPR (CR)	Read general purpose registers
HALT (CR)	Halt
IPL (CR)	IPL from default address
IPL=xxxx (CR)	IPL from xxxx
IS (CR)	Clear instruction stop
IS=xxxxxxxx (CR)	Set instruction stop
MA=xxxxxx (CR)	Read physical memory address location
(CR)	Increment and read memory address
MAV=xxxxxx (CR)	Read virtual memory address location
(CR)	Increment and read memory address
MD=xxxxxxxx (CR)	Write memory data
=xxxxxxxx (CR)	Increment and write memory data
= (CR)	Increment and write previous data
MSGE (CR)	Message
OVR (CR)	Toggle clock override
PRIP (CR)	Set primary panel (master only)
PSD (CR)	Read Program Status Doubleword (1 and 2)
PSD=xxxxxxxx (CR)	Write Program Status Word (2)
PSW (CR)	Read Program Status Word (1)
PSW=xxxxxxxx (CR)	Write Program Status Word (1)
REGA=xxxxxxxx (CR)	Write General Purpose Register A
RS (CR)	Clear read operand stop
RS=xxxxxxxx (CR)	Set read operand stop
RST (CR)	Reset
RUN (CR)	Run
SECP (CR)	Set secondary panel (master and slave)
STEP (CR)	Instruction step
(CR)	Instruction step
WS (CR)	Clear write operand stop

WS=xxxxxxxx (CR)	Set write operand stop
@@C	Enter console mode
@@P	Enter panel mode
(LF)	Repeat command

Notes:

1 (CR) denotes Carriage Return after each command.

2 LOCK ON and LOCK OFF are not supported by the CRT panel.

Console Mode

To change from SCP mode to console mode, it is necessary for an operator to input @@C (CR).

Upon receipt of the CR command following the @@C command, the firmware moves the cursor on the CRT to the extreme left margin of the next line.

To return to the control panel mode, enter @@P (CR). When the control panel mode is selected, // is used as the prompt.

Glossary

Address Space	The range of addresses described by a given configuration of the SELMAP. Normally the address range is from 0 to some map block boundary. Also referred to as "logical address space". When the CPU is unmapped, the only address space is the range of physical memory configured.
Batch	Batch processing is the sequential processing of multiple jobs, with the operating system's Job Control program providing automatic job-to-job transition. Job Control and batch are often used synonymously.
Indirectly-Connected Interrupt	A hardware interrupt for which a special control block is defined at SYSGEN. The control block is connected to an active software task via the M.CONN service or OPCOM CONNECT command so that when the hardware interrupt occurs, the task is resumed. Resumption is based on the software priority of the user task.
Console	The teletypewriter or CRT terminal which is used by the operator to control the operation of MPX-32. Also called "operator's console", "OPCOM console", or "system console".
Context Area	Task program Status Doubleword (PSD) and registers stored in the Task Service Area.
Context Switch	The process of one task relinquishing CPU control and another task gaining CPU control. Software: based on availability of resources and software priorities. Hardware: based on hardware interrupt and trap priorities.

Controller Definition Table (CDT)	The CDT contains an entry for each controller described to SYSGEN. This entry describes the channel number, device type code, the number of units on the controller, Unit Definition Table (UDT) address, etc. IOCS uses the CDT entry to record controller status and control I/O queue information.
DATAPOOL	Datapool is a memory partition which can be defined at SYSGEN or via the FILEMGR. DATAPOOL structure is defined by a datapool dictionary created and maintained via the Datapool Editor utility (DPEDIT). Elements of DATAPOOL storage are referenced symbolically rather than by their address.
DATAPOOL Dictionary	The DATAPOOL dictionary is a map of elements in the DATAPOOL. The dictionary equates a logical location in the DATAPOOL to a symbolic value.
DATAPOOL Editor	The DATAPOOL Editor (DPEDIT) is a utility used to maintain the DATAPOOL dictionary. Definitions of elements in the DATAPOOL can be modified, deleted, or added to the dictionary via the DATAPOOL Editor.
Device Handlers	Device handlers execute queued I/O commands, process service interrupts, and perform device testing functions. A handler is provided for each standard peripheral device.
Directly-Connected Interrupt	A user module that is incorporated with the resident MPX-32 operating system at SYSGEN and associated with a specific hardware interrupt level; hardware context switch on interrupt goes directly to this task without routing through MPX-32.
Disc Allocation Unit	A disc allocation unit is a number of records, the size of which is disc-dependent. There are currently between one and four 192-word blocks per disc allocation unit.
Dispatch Queue	A list of tasks that have been activated, their priorities, and other pertinent information. Referenced by the Execution Scheduler for software context switches.
Fast File	The file definition is retrieved from the System Master Directory (SMD) with one disc access.

File	A collection of related records on a recording medium. MPX-32 input files may be on disc, tape, paper tape, or cards, or input can come from a terminal. Output files can be directed to the above as well as line printers. "File" in MPX-32 normally denotes disc files (as opposed to files on a device medium).
File Assignment Table (FAT)	A FAT describes the channel and the device to which a file is assigned. There is a FAT entry for each logical file code referenced by a user program.
File Code	A three-character field in a File Control Block (FCB) that contains a mnemonic that refers to the file. The mnemonic can be used to refer to the FCB in Job Control statements, TSM and OPCOM commands, and system services such as $ASSIGNx. Synonymous with logical file code.
File Control Block (FCB)	An FCB contains parameters which describe an I/O operation and identify the file code on which the operation is to be performed. User tasks and system programs define the contents of their related FCB's.
File Manager	The File Manager (FILEMGR) is a utility program used to maintain permanent disc files. It provides a set of directives for creating, deleting, saving, and restoring files.
File Name	A file name is the one- to eight-ASCII character name of a permanent file. It is established when the file is created.
File Pointer Table (FPT)	The FPT provides the link between a user's FCB identifying the file and the FAT entry describing the device to which the user's file code is assigned. Static contents of the FPT are built by the MPX-32 resource allocation service; they are updated dynamically by the device allocation and deallocation system services. The FPT table is maintained in the task's TSA built by MPX-32.
File System Executive (FISE)	The File System Executive provides the interface between user tasks and available file space. It performs dynamic allocation and deallocation of temporary file space, locates or adds permanent file definitions in the System Master Directory, and maintains the disc allocation map.

Global Common	Global common is a set of memory partitions defined via SYSGEN or the FILEMGR which multiple tasks can access as a global resource. Definitions of Global Common items are determined by their relative position within the common partition.
I/O Control System	The Input/Output Control System (IOCS) processes I/O requests by the task and queues these requests by the software priority of the task.
Impure Data	Tables, arrays, etc. that are an integral part of a task, yet can be modified during execution. If a task is sectioned, impure data are located in DSECT and reside in read/write memory separate from code and pure data, which are located in CSECT and reside in read-only memory.
Independent	A task that runs independent of an interactive or batch environment, normally a real-time task.
Interactive	Of or pertaining to terminals. An interactive task is one doing I/O to a terminal.
Interrupt	An internal or external event that requires rapid service by special software (or firmware) routines. The CPU preserves its current state and transfers control to the required interrupt handler.
Interrupt Control Block (ICB)	A series of memory locations included in a software interrupt handler that are pointed to by the IVL. This is where the CPU preserves its current state and transfers control to a software handler.
Interrupt Vector Location (IVL)	Dedicated memory location for connecting an interrupt to connecting an ICB.
Job	A job is a sequential set of tasks whose batched execution is scheduled by the Job Control program based on Job Control statements.

Job Control Program	A program (i.e., a collection of tasks) which provides batch processing services, including: job-to-job transition, management of spooled I/O, run-time I/O device assignments. Job Control processing is controlled through Job Control Statements. One copy of the Job Control program is used for each job processed concurrently in the system. The number of Job Control programs that can run concurrently is specified at SYSGEN time.
Job Name	A job name is the one- to eight-character job name from the $JOB statement which defines the beginning of the job. An owner name is also specified for the job.
Job Queue	The single ordered set of all SYC files containing spooled jobs waiting to be initiated by Job Control programs.
Job Sequence Number	A job sequence number is a decimal number which is assigned sequentially to a job when it is spooled to SYC disc space by the System Input program.
Job Stream	The sequence of jobs processed by a particular Job Control program. The number of job streams processed by MPX-32 is the number of Job Control programs which may run concurrently, which is established by SYSGEN. Synonymous with batchstream.
Key	A one-to-eight character code associated with an ownername/username to provide a degree of protection for system access and user files.
Load Module	A cataloged task that may be activated in one of three task activation environments: independent, interactive, or batch.
Load Module Information Table	A part of a load module produced by the Cataloger that contains special indicators for the task, a relocation matrix, and resource requirements.
Logical File Code (lfc)	Synonymous with file code (q.v.).
Macro Library Editor	The Macro Library Editor (MACLIBR) is a utility program used to create disc and tape resident macro libraries for use with the Macro Assembler.

Map	Synonymous with SELMAP (q.v.).
Map Block	The set of memory locations whose addresses are calculated from a single map register. All map blocks are the same size and begin on addresses which are multiples of their size. Map block size is determined by hardware map granularity. On the 32/7x, a map block is 8KW and on the 32/27, a map block is 2KW. In general, a map block is the smallest mappable quantum of physical memory.
Media Conversion	The Media Conversion utility (MEDIA) is used for media to media conversion, media copying, and media verification. It provides functions ranging from file duplication to merging media inputs into single or multiple media outputs.
MPX-32 Executive	The MPX-32 Executive schedules CPU processing for all tasks running under software priority levels. MPX-32 stands for Mapped Programming Executive.
Object Module	An object module is the smallest unit of a task that is output on the SGO by the Assembler or Compiler and is identified by name. It consists of relocatable object code to be processed by the Cataloger.
On-Line or Online	Synomymous with INTERACTIVE (q.v.).
Operator Communications	Operator Communications (OPCOM) commands allow the user to exercise system control through a set of commands entered via the console or a terminal.
Owner	One who owns a task. See owner name.
Owner Name	A one to eight ASCII character, left justified, blank filled name, maintained in the M.KEY file, if it exists, and associated with task activation.
Priority	Relative eligibility. Hardware priority refers to the priority scheme of external interrupts and service interrupts. Software priority refers to the priority scheme used by MPX-32 to resolve conflicting requests for resources. Priority for a task activated independent of the

batch or interactive environment is the priority established for the task when it is cataloged. The priority for a task activated interactively (via a RUN command) is the terminal priority established at SYSGEN. The priority for a task activated as part of a batch job is 64 unless modified via the OPCOM command URGENT.

Privileged

Privilege is a state of processing that allows access to a set of otherwise unexecutable "privileged" instructions, and to a privileged collection of system services. The privileged/ unprivileged state of a task is indicated when it is cataloged.

Procedure

Synonymous with 'code'.

Processor

A SYSTEMS program that a user can run to perform a related set of operations such as file maintenance, media conversion, FORTRAN compilation, etc. Also used synonymously with utility. Language processors are compilers, assemblers, or interpreters. Command processors are Job Control, TSM, and OPCOM.

Program

A program is a part of a task, a task, or a set of related tasks.

Program Status Doubleword (PSD)

Hardware registers that define the state of the CPU at any given time.

Protection

Memory protection: Write by unprivileged programs is not allowed. File protection: Use of passwords etc. that allow the user who creates a file to protect it, if desired.

Protection Granule

The smallest unit of memory within a map block that can be individually write locked. On the 32/7x, a protection granule is 512 contiguous words beginning on a 512-word boundary.

Pseudonym

A one-to-eight character name which provides more information about a task's environment than the task number/ownername, etc. For example, TSM uses the pseudonym TSM*terminal-number to identify a task by the terminal it is running on and Job Control uses the pseudonym .Odevmnc to identify a spooling task's target device.

Pure Data	Tables, arrays, etc. that are an integral part of a task and do not change during execution. Located with code. If a task is sectioned, pure data are included with code in CSECT and reside in read-only memory.
Real-Time	Any user task which is not activated in an interactive or batch environment is usually considered a real-time task. Real-time tasks are typically designed to respond to external stimuli (such as external interrupts) or to be executed periodically, as with a timer.
Reel Identifier	A 1-4 character name output with a magnetic tape mount message and also written to a tape along with the volume number if using multivolume tape.
Reentrancy	The logical property of procedural code which allows it to be executed completely asynchronously by multiple concurrent tasks.
Resident	In MPX-32, user's code (e.g., an interrupt handler) that is incorporated with the MPX-32 operating system at SYSGEN and is mapped into every user's address space with the operating system.
Resident (Locked in Memory)	A task that is non-swappable. It remains in memory until it exits or aborts.
Resource Allocator	The part of the resident system whose primary function is to allocate memory and peripherals.
Resourcemark	A numeric value used cooperatively by tasks to synchronize access to a common resource such as a disc file or sharable device.
Resource Requirement Summary (RRS)	A part of the Load Module Information Table that defines the devices, files, etc. required for the task.
Scheduler	The CPU scheduler dispatches CPU control based upon system events and task priorities. Synonymous with MPX-32 Executive.

Slow File	On MPX-32, a slow file is one in which the scatter- storage mechanism used to build SMD entries may use a back-up algorithm if the file name maps into an existing active file (collision mapping). If the back-up algorithm is used, it may require additional disc access.
Source Update	The Source Update (UPDATE) utility is used to create and update user and system source files.
Subroutine Library Editor	The Subroutine Library Editor (LIBED) is a utility program for creating and updating subroutine libraries.
Supervisor Call (SVC)	Supervisor Call is a trap that provides user interface to system services. Also the name of the software instruction that causes the trap.
System General Object (SGO) File	SGO files are used for the accumulation of object code within batch jobs. A separate SGO file is allocated for each job and exists for the duration of the job.
System Binary Output (SBO) File	An SBO file is a temporary file used for punched output. SBO files generated by real-time tasks are output to destination peripheral devices when the files are deallocated. SBO files generated by a batch job are output to destination peripheral devices upon job completion.
System Control File (SYC)	An SYC file is a disc file that provides intermediate storage for Job Control statements, object code, and data for a batch job. A separate job file is dynamically created on the SYC for each task initiated in a user's job file, and when the last task in the job completes execution, the job's SYC is deleted.
System Generation (SYSGEN)	The System Generation program is used to tailor the MPX-32 operating system to the hardware and software requirements of an installation.
System Input	The System Input task transfers batch input from devices and dynamically linked temporary and permanent disc files to intermediate storage on System Control (SYC) files.

System Listed Output	An SLO file is a temporary file used for listed output. SLO files generated by real-time tasks are output to destination peripheral devices when the files are deallocated. SLO files generated by a batch job are output to destination peripheral devices upon job completion.
System Loader	The System Loader is the part of the Allocator that loads any cataloged load module into memory upon request; it performs all necessary biasing of relocatable data.
System Master Directory (SMD)	The SMD is created at SYSGEN and describes the location, length, name, and related information about each file or memory partition in the system. Entries are made in the SMD for load modules as a part of the cataloging process.
System Output	The System Output task outputs the print (SLO) and punch (SBO) data collected for each task from temporary disc files to destination listed and punched output devices.
System Service	Equivalent to the term "Monitor Service" in RTM.
Task	A task is a body of code which is scheduled for CPU time as a single entity. It has one Dispatch Queue Entry (DQE) and one Task Service Area (TSA). A task may be activated as real-time, batch, or on-line.
Task Identifier	Tasks are identified by attributes. Attributes of a task include its unique task number (see below), the name of its owner, its load module name, its job sequence number (if batch), and an optional pseudonym to use in intertask communication. The task number is the only attribute that can be used to abort or to communicate with a task that can be multicopied. It may be obtained by the user by specifying one or a combination of task attributes.
Task Number	A task number is a sequential 32-bit number which is assigned to the task when it is activated and identifies the task uniquely over time (until system restart). Task attributes known by the user can be used to obtain the task number. The operator must refer to the task only by task number when attempting to, for example, abort.
Task Scheduler	See Scheduler.

Terminal	An I/O device featuring a keyboard for input and either a CRT or a printer for output. The asynchronous communication channels for terminals are managed by the MPX-32 Terminal Services Manager (TSM).
Terminal Services Manager (TSM)	A collection of tasks and service routines which control the MPX-32 interactive environment.
Timer	An optional task scheduling mechanism provided for each task by MPX-32. Timers are managed using the same real-time clock which is used to maintain the time-of-day.
Timer Unit	A timer unit is a number of real-time clock interrupts selected by the user at SYSGEN to represent a logical unit of time.
Transfer Control Word (TCW)	A TCW contains byte count and data address information used to define and control an I/O operation involving data transfer.
Trap Processors	Most Trap Processors are firmware and/or software routines that are entered when any exceptional condition trap occurs; they then perform the appropriate processing. The SVC trap is a special case used to transfer control from a task to the MPX-32 operating system.
Unit Definition Table (UDT)	The UDT describes each peripheral device in the system. The entry describes the device subaddress and channel number, the device type code, unit status, number of allocation units, the physical characteristics of the device, etc.
Unprivileged	An unprivileged task is one that does not execute "privileged" instructions. The privileged/unprivileged state of a task is established when it is cataloged.
User	See User Name.
User Name	One to eight ASCII character, left justified, blank filled name, maintained in the M.KEY file, if it exists, and associated with all user files in the System Master Directory (SMD). The absence of a user name indicates a system file.

INDEX

This index is for use with the MPX-32 Reference Manual, Volumes I, II and III. Its format is section number (volume number) where the information can be found.